UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

T.128

(02/98)

SÉRIE T: TERMINAUX DES SERVICES TÉLÉMATIQUES

Partage d'application multipoint

Recommandation UIT-T T.128

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE T

TERMINAUX DES SERVICES TÉLÉMATIQUES

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

RECOMMANDATION UIT-T T.128

PARTAGE D'APPLICATION MULTIPOINT

Résumé

La présente Recommandation définit un protocole supportant le partage d'application multipoint.

Le protocole T.128 supporte le partage d'applications informatiques multipoints en autorisant qu'une vue sur une application informatique s'exécutant sur un site donnée soit présentée sous la forme d'une session sur d'autres sites. Chaque site peut, selon des conditions spécifiées, prendre le contrôle de l'application informatique partagée en envoyant de l'information à l'aide d'un clavier et d'un dispositif de pointage distants. Ce style de partage d'application ne requiert pas et ne produit pas de disposition particulière pour la synchronisation d'instances multiples de la même application informatique se déroulant sur plusieurs sites. Au contraire, il permet la visualisation et le contrôle distants d'une instance d'application unique en donnant l'impression que l'application se déroule localement.

La présente Recommandation utilise les services fournis par les Recommandations T.122 (MCS) et T.124 (GCC).

Source

La Recommandation UIT-T T.128, élaborée par la Commission d'études 16 (1997-2000) de l'UIT-T, a été approuvée le 6 février 1998 selon la procédure définie dans la Résolution n° 1 de la CMNT.

AVANT-PROPOS

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

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

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

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

NOTE

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

DROITS DE PROPRIÉTÉ INTELLECTUELLE

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

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Page

Recommandation T.128

PARTAGE D'APPLICATION MULTIPOINT

(Genève, 1998)

1 Domaine

La présente Recommandation définit un protocole supportant le partage d'application multipoint. Il utilise les services fournis par les Recommandations T.122 (MCS) et T.124 (GCC).

Les détails de la communication avec les dispositifs d'entrée et de sortie et les interfaces utilisateurs sur le terminal hôte sont considérés comme étant hors du domaine de la présente Recommandation et sont laissés à la discrétion de l'implémenteur. Ainsi, la présente Recommandation ne fait aucune hypothèse sur l'appartenance des dispositifs d'entrée et de sortie ainsi que des interfaces utilisateurs à une quelconque architecture spécifique.

La Figure 1 présente un aperçu général au niveau d'un même nœud du domaine de la présente Recommandation et de ses relations aux autres éléments de la série T.120.

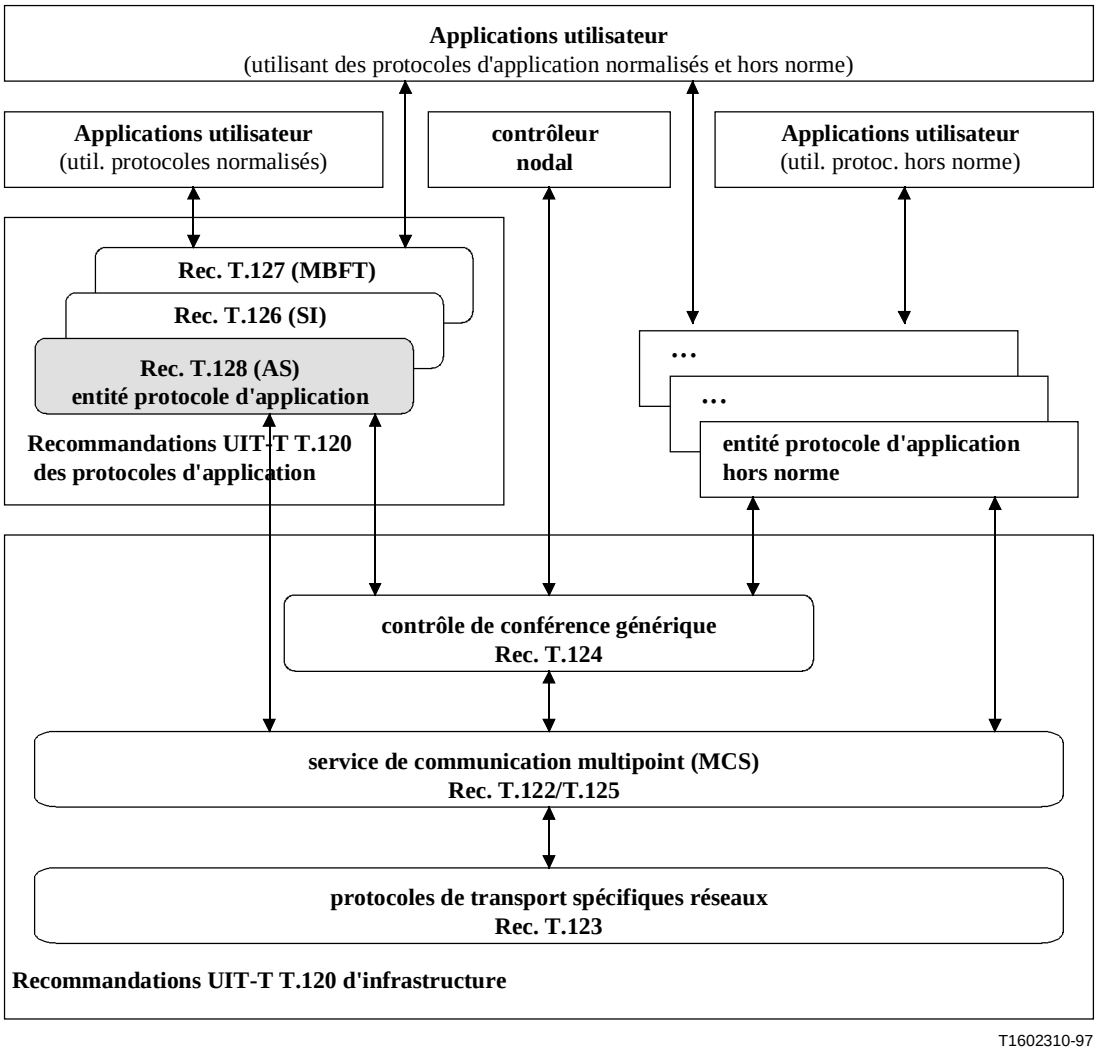


Figure 1/T.128 – Domaine de la Recommandation T.128

2 Références Normatives

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

– Recommandation F.710 du CCITT (1991), *Principes généraux applicables au service de conférence audiographique*.

– Recommandation UIT-T H.221 (1997), *Structure de trame pour canal d'un débit de 64 à 1920 kbit/s pour les téléservices audiovisuels*.

– Recommandation T.35 du CCITT (1991), *Procédure d'attribution des codes définis par le CCITT dans le cas de moyens non normalisés*.

– Recommandation T.50 du CCITT (1992), *Alphabet international de référence (ancien alphabet international n° 5 ou AI5) – Technologies de l'information – Jeux de caractères codés à 7 bits pour l'échange d'informations*.

– Recommandation UIT-T T.120 (1996), *Protocoles de données pour conférence multimédia*.

– Recommandation UIT-T T.121 (1996), *Modèle générique d'application*.

– Recommandation UIT-T T.122 (1993), *Service de communication multipoint pour la définition des services de conférence audiographique et conférence audiovisuelle*.

– Recommandation UIT-T T.123 (1996), *Piles protocolaires de données propres au réseau pour conférences multimédias*.

– Recommandation UIT-T T.124 (1995), *Commande de conférence générique*.

– Recommandation UIT-T T.125 (1994), *Spécification de protocole du service de communication multipoint*.

– Recommandation V.42 *bis* du CCITT (1990), *Procédures de compression des données pour les équipements de terminaison du circuit de données (ETCD) utilisant des procédures de correction d'erreur*.

– Recommandation UIT-T X.509 (1997) | ISO/CEI 9594-8:1997, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: cadre d'authentification*.

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

– Recommandation UIT-T X.691 (1997) | ISO/CEI 8825-2[1], *Technologies de l'information – Règles de codage ASN.1: spécification des règles de codage compact*.

– ISO/CEI 8859-1:1998, *Technologies de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1: Alphabet latin n° 1*.

– ISO/CEI 10646-1:1993, *Technologies de l'information – Jeu universel de caractères codés à plusieurs octets – Partie 1: Architecture et table multilingue*.

[1] A publier.

3 Définitions

La présente Recommandation définit les termes suivants:

3.1 partage d'application (AS, a*pplication sharing)*: processus par lequel deux terminaux ou plus coopèrent pour partager les sorties d'applications fonctionnant sur un ou plusieurs terminaux avec d'autres terminaux et pour fournir des entrées aux autres applications.

3.2 entité de conférence de partage d'application (ASCE, *application sharing conference entity)*: entité de protocole d'application interagissant avec une application utilisateur au-dessus, et, en dessous, avec des fournisseurs locaux de MCS (service de communication multipoint) et de contrôle de conférence générique (GCC, *generic conference control*), pour implémenter un partage d'application. Les données sont échangées entre des ASCE homologues utilisant des unités de données de protocole de partage d'application (ASPDU, *application sharing protocol data unit*).

3.3 phototrame: zone rectangulaire décrite par un tableau de points à deux dimensions. Ces points peuvent être codés selon une variété de méthodes de codage.

3.4 table de couleurs: ensemble fini de couleurs défini par au moins trois couleurs primaires linéairement indépendantes. C'est un synonyme de palette (voir plus bas), mais on l'utilise dans la présente Recommandation pour désigner la palette particulière d'antémémoire associée aux données de phototrame mises en antémémoire.

3.5 moniteur: zone d'affichage physique ou logique d'un terminal particulier ou d'un gestionnaire de fenêtre adressée par l'ASCE dans les capacités de l'AS. Une ASCE peut choisir d'adresser une taille correspondant à la zone d'affichage réelle du terminal ou tout autre zone d'affichage logique.

3.6 poignée: nombre unique pour toute la session AS utilisé pour identifier un élément adressable.

3.7 capacité hors norme: capacité sortant du domaine de la présente Recommandation mais adoptée à la suite d'une négociation reconnue par tous les participants de la session.

3.8 palette: ensemble fini de couleurs défini par au moins trois couleurs primaires linéairement indépendantes.

3.9 palettisé: terme utilisé pour décrire les éléments de protocoles (tels que des données phototramées) constitués de points palettisés. La couleur du point palettisé est spécifiée par le numéro de la couleur correspondant à la position dans la palette référencée par la valeur de point.

3.10 curseur: phototrame pouvant être déplacée sur le moniteur virtuel et utilisée comme indicateur de position.

3.11 capacité normalisée: capacité définie dans le domaine de la présente Recommandation mais non requise par toutes les implémentations d'ASCE. Il faut noter que toutes les capacités normalisées doivent être négociées avant chaque utilisation.

3.12 unicode: format de chaîne de caractères multilingue défini par l'ISO/CEI 10646-1.

3.13 moniteur virtuel: moniteur logique correspondant au plus grand des moniteurs des ASCEs hôtes.

3.14 fenêtre: zone rectangulaire du moniteur correspondant à une zone d'affichage de l'interface utilisateur gérée par le gestionnaire de fenêtre du terminal.

3.15 gestionnaire de fenêtre: programme s'exécutant sur le terminal, chargé de la gestion d'un ensemble de fenêtres d'interface d'utilisateur sur le moniteur du terminal.

4 Abréviations

La présente Recommandation utilise les abréviations suivantes.

AS partage d'application (*application sharing*)

ASCE entité de conférence de partage d'application (*application sharing conferencing entity*)

ASPDU unité de donnée de protocole de partage d'application (*application sharing protocol data unit*)

CEI Commission électrotechnique internationale

GCC contrôle de conférence générique (*generic conference control*)

ISO Organisation internationale de normalisation (*International organization for standardization*)

MCS service de communication multipoint (*multipoint communication service*)

UIT Union internationale des télécommunications

5 Aperçu général

Le protocole AS permet le partage d'applications informatiques multipoints en autorisant qu'une vue sur une application informatique s'exécutant sur un site donnée soit présentée sous la forme d'une session sur d'autres sites. Chaque site peut, selon des conditions spécifiées, prendre le contrôle de l'application informatique partagée en envoyant de l'information à l'aide d'un clavier et d'un dispositif de pointage distants. Ce style de partage d'application ne requiert pas et ne produit pas de disposition particulière pour la synchronisation d'instances multiples de la même application informatique se déroulant sur plusieurs sites. Au contraire, il permet la visualisation et le contrôle distants d'une instance d'application unique en donnant l'impression que l'application se déroule localement.

Une session AS comprend une ou plusieurs ASCE coopérant via le protocole AS pour partager une application ou plus à l'intérieur de la même session. Le protocole AS définit des interactions entre ASCE. Il ne définit pas des interactions entre une ASCE et le système d'exploitation ou les dispositifs d'entrée et sortie du terminal local.

5.1 Mode de base et mode hérité

Le protocole AS supporte deux modes opératoires. Le mode hérité est fourni pour l'interopérabilité avec une base installée existante d'équipements de terminaux d'utilisateurs. Le mode de base fournit des attributs supplémentaires en vue d'une amélioration future du protocole AS.

Toutes les ASCE se conformant à la présente Recommandation implémenteront les deux modes de base et hérité. L'usage du protocole AS du GCC (voir le paragraphe 7) est défini de telle sorte que lorsque toutes les ASCE d'une même conférence supportent le protocole de base (à savoir, il y a seulement des terminaux sans mode hérité préexistant dans la conférence), alors toutes les ASCE utiliseront le mode de base. L'UIT a l'intention de mettre toutes les améliorations à venir du protocole AS dans le mode de base.

Une proportion significative de la présente Recommandation est commune aux deux modes, avec les différences entre les deux modes portant sur:

- la manière dont les capacités sont échangées et négociées: voir le paragraphe 8.2;
- la manière dont les ASCE sont activées et désactivées: voir le paragraphe 8.4;
- la définition et le codage des ASPDU: voir le paragraphe 9.

Les différences entre les modes de base et légal du protocole AS sont précisées dans le texte.

5.2 Concepts du partage d'application

5.2.1 Modèles de fenêtre et de moniteur

La présente Recommandation ne fait aucune hypothèse et ne requiert aucun équipement local de terminal ou d'environnement de terminal particulier. Elle ne suppose pas non plus ni ne requiert de gestionnaire local de fenêtre de terminal ou de modèle de fenêtrage particulier. Par exemple, elle ne suppose pas:

- que le terminal est un PC ou une station de travail – ce peut être un terminal dédié;
- que les fenêtres du protocole AS correspondent aux fenêtres gérées par un gestionnaire de fenêtre de terminal – elles peuvent mapper des zones arbitraires de l'écran du terminal local;
- que les moniteurs du protocole AS correspondent aux écrans des terminaux locaux – ils peuvent mapper des fenêtres de navigateurs ou d'afficheurs locaux, ou à des zones d'affichage de terminaux dédiés.

Puisque la présente Recommandation ne suppose pas ni ne requiert aucun équipement de terminal local ou d'environnement de terminal particulier, le protocole AS définit avant tout un moniteur logique et un modèle de fenêtres formé d'une collection de fenêtres sur un moniteur. Ceci requiert que chaque ASCE active dans une session AS soit capable de mapper au modèle de fenêtre et de moniteur du protocole AS les concepts d'environnement de terminal local et vice versa. Le modèle de fenêtre et de moniteur du protocole AS supporte les concepts clés suivants:

- moniteur — le moniteur d'AS est un rectangle défini selon des coordonnées de moniteur virtuel. Une ASCE devrait fournir le mappage approprié entre un moniteur d'AS et un concept de terminal local;
- moniteur virtuel — le moniteur virtuel est l'union des tailles des moniteurs des ASCE hôtes (à savoir les ASCE qui sont des fenêtres hôtes – voir ci-dessous). L'origine du moniteur virtuel (à savoir le pixel 0,0) est défini en haut à gauche;
- fenêtre — une fenêtre AS est un rectangle défini dans des coordonnées de moniteur virtuel. Les fenêtres AS peuvent être totalement à l'intérieur, partiellement à l'intérieur ou totalement à l'extérieur du moniteur virtuel. Une ASCE devrait fournir le mappage approprié entre les fenêtres AS et le concept de fenêtre de terminal local;
- ordre d'empilement — l'ordre d'empilement (ou ordre en Z) d'une fenêtre AS définit un ordre de profondeur de fenêtre entre fenêtres sur le moniteur virtuel, de sorte que, si l'on prend deux fenêtres, la fenêtre la plus élevée dans l'ordre en Z est en avant et peut cacher l'autre;
- tout en haut — la fenêtre AS tout en haut dans l'ordre en Z est au premier plan et peut cacher toutes les autres dans l'ordre en Z;
- tout en bas — la fenêtre AS tout en bas dans l'ordre en Z est à l'arrière plan et peut être cachée par toutes les autres dans l'ordre en Z.

D'autres concepts du protocole AS sont expliqués ci-dessous ou dans les paragraphes appropriés de description du protocole.

La Figure 2 montre un exemple de collection d'ASCE dans une session AS. Cette figure est utilisée tout au long du présent sous-paragraphe pour illustrer les concepts clés du protocole AS.

- L'ASCE A est une fonction complète, affichante et hébergeante – à savoir qu'elle partage à la fois des fenêtres dans la session AS et affiche des fenêtres reflets partagées avec d'autres ASCE. Elle est exécutée sur un PC généraliste, où le moniteur de l'AS et les fenêtres de l'AS sont mappés directement sur le moniteur du gestionnaire de fenêtre du terminal.
- L'ASCE B est une fonction visualisante seulement – à savoir qu'elle affiche des fenêtres reflets partagées avec d'autres ASCE mais ne partage pas elle-même des fenêtres dans la session AS. Elle est exécutée sur un terminal d'affichage spécialisé ou le moniteur de l'AS et les fenêtres de l'AS sont mappées sur une fenêtre visionneur de taille configurable d'une manière indépendante, qui est actuellement plus petite que le moniteur de l'AS et donc défilante pour pouvoir afficher le moniteur de l'AS dans son intégralité.
- L'ASCE C est une fonction hébergeante seulement – à savoir qu'elle partage des fenêtres dans la session AS, mais n'affiche pas de fenêtres reflets. Elle est exécutée sur une station haut de gamme non surveillée, où le moniteur de l'AS et les fenêtres de l'AS sont mappées directement sur le moniteur du gestionnaire de fenêtre du terminal.

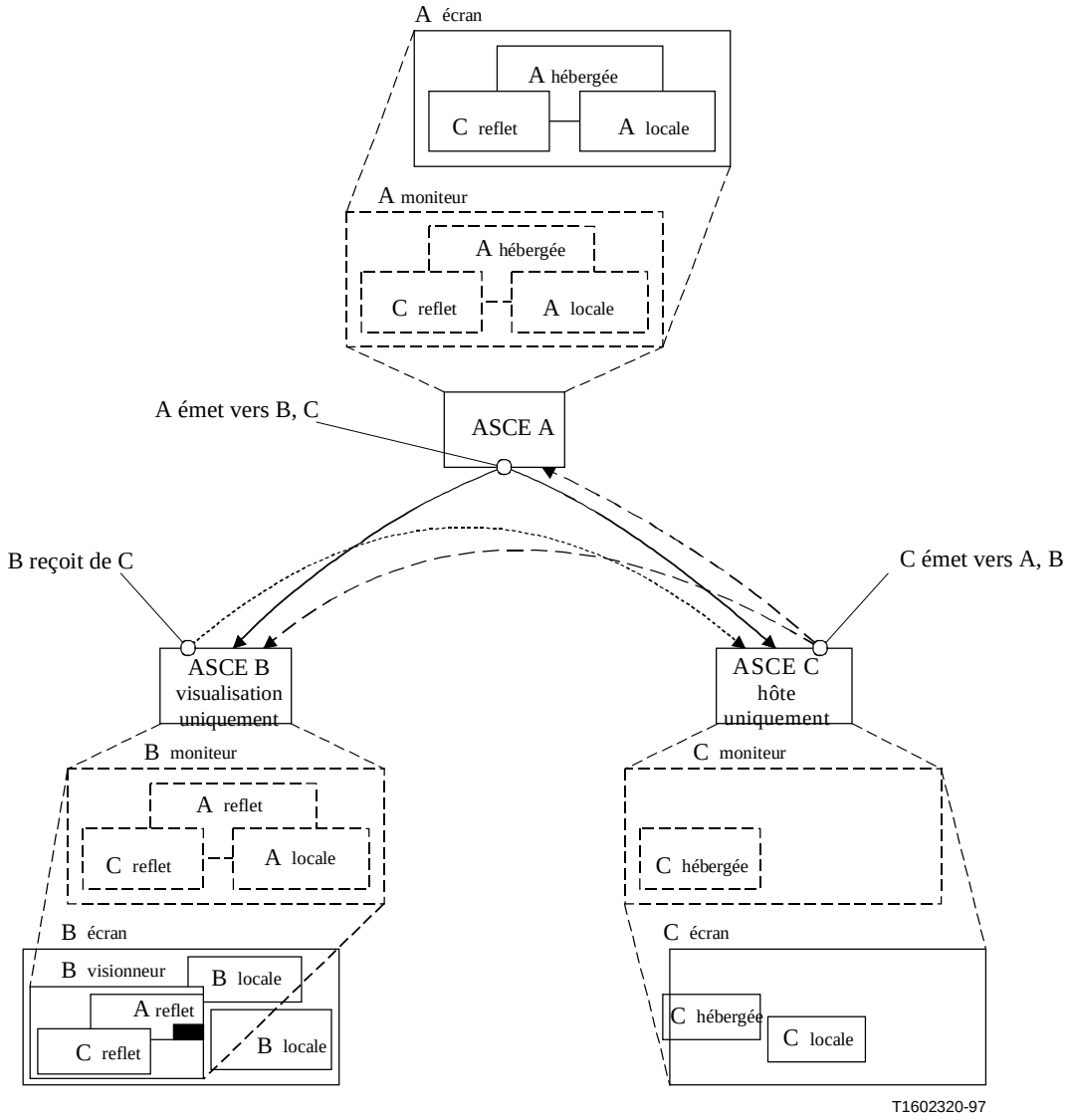


Figure 2/T.128 – Exemple de collection d'ASCE dans une session AS

Dans une session AS, les fenêtres sont des types suivants:

- hébergée: les fenêtres hébergées sont possédées par une application s'exécutant sur le terminal local et sont partagées dans la session AS. Pour chaque fenêtre hébergée il y aura une fenêtre reflet correspondante sur une ASCE homologue (sauf dans le cas où une ASCE est seulement hôte);
- reflet: les fenêtres reflets sont tracées par l'ASCE et correspondent à une fenêtre hébergée sur une ASCE homologue particulière;
- locale: les fenêtres locales ne sont pas partagées. Leurs sorties sont visibles seulement sur le terminal local. Une ASCE a seulement besoin de surveiller les fenêtres locales pour les superviser lorsqu'elles masquent une fenêtre hébergée et lorsque ce masquage empêche l'ASCE d'obtenir une information de traçage valide pour la fenêtre hébergée.

La Figure 2 montre (pour chaque ASCE) à la fois les fenêtres locales sur l'écran du terminal réel et les fenêtre de l'AS sur le moniteur de l'AS qui sont gérées par l'ASCE via le protocole AS.

- L'ASCE A gère trois fenêtres:
  - une fenêtre reflet correspondant à la fenêtre hébergée sur l'ASCE C;
  - une fenêtre hébergée partagée dans la session de l'AS;
  - une fenêtre locale – car elle masque la fenêtre hébergée et sur le terminal local qui l'empêche d'obtenir une information de traçage valide pour cette fenêtre hébergée.

  La fenêtre reflet C est tracée sur l'écran du terminal local par l'ASCE – et les deux autres fenêtres sont affichées par le terminal local.
- L'ASCE B gère aussi trois fenêtres:
  - une fenêtre reflet correspondant à la fenêtre hébergée sur l'ASCE A;
  - une fenêtre reflet correspondant à la fenêtre hébergée sur l'ASCE C;
  - la fenêtre locale de l'ASCE A – car elle masque la fenêtre hébergée sur l'ASCE A.
  - l'ASCE B ne supervise pas les fenêtres locales sur l'écran de son terminal car elle ne présente pas de fenêtre.

  Les fenêtres reflets A et C sont tracées dans une fenêtre de visionneur sur l'écran du terminal local par l'ASCE. La fenêtre locale n'est pas tracée, mais est plutôt utilisée pour calculer et tracer une zone masquée, masquant partiellement la fenêtre reflet A (montrée comme une zone noire en bas à droite de la fenêtre reflet A).
- L'ASCE C gère une fenêtre:
  - une fenêtre hébergée partagée dans la session de l'AS;
  - l'ASCE C ne supervise pas la fenêtre reflet correspondant à la fenêtre hébergée sur l'ASCE A comme elle ne visualise pas les fenêtres reflets;
  - l'ASCE C ne supervise pas sa fenêtre locale, puisque la fenêtre locale masque la fenêtre hébergée, le terminal local lui permet d'obtenir une information de traçage valide pour cette fenêtre hébergée;
  - la fenêtre hébergée est affichée par le terminal local.

Voir 8.10 pour plus d'information sur la gestion des fenêtres et de la liste de fenêtres.

5.2.2 Sortie

Quand une ASCE héberge une fenêtre, elle est responsable de la construction d'un flux de sortie pour cette fenêtre qui permettra aux autres ASCE de tracer fidèlement les fenêtres reflets correspondantes.

Le flux du protocole de sortie de l'AS est multidiffusé via le AS-CHANNEL vers toutes les autres ASCE actives à l'intérieur de la conférence. Si une ASCE ne fait qu'héberger (comme c'est le cas dans la Figure 2), elle peut alors ignorer tout ou partie du flux de sortie. D'une manière identique, une ASCE réceptrice peut décider de tracer seulement un sous-ensemble du flux de sortie sur sa source (à savoir tracer seulement les fenêtres d'une ASCE particulière).

Une ASCE détermine le répertoire de sortie autorisé basé sur les capacités actuellement négociées. Ceci requiert que l'ASCE soit capable d'ajuster sa stratégie de sortie car des ASCE munies de capacités plus basses ou plus élevées peuvent rejoindre ou quitter la conférence. Cependant à l'intérieur du répertoire autorisé en cours et en relation avec les contraintes inhérentes au protocole AS, une ASCE a une liberté considérable pour construire un flux de sortie.

Le flux de sortie est composé:

- d'une information d'état – telle qu'une information sur une liste de fenêtres (voir 8.10);
- d'une information de couleur – telle qu'une palette et une information de table de couleurs cachée (voir 8.15 et 8.16.8);
- de données de phototrames et d'ordres (voir 8.16 et 8.17).

Le protocole AS supporte les ordres primaires suivants:

- Blt destination;
- Blt motif;
- Blt écran;
- Blt mémoire;
- Blt opération point-à-point ternaire;
- texte;
- texte étendu;
- trame;
- rectangle;
- ligne;
- rectangle opaque;
- sauvegarde moniteur;
- origine moniteur.

Dans des scénarios typiques de partage d'application, les ordres forment une proportion très élevée du trafic ASPDU. Les ordres peuvent alors être codés de manière différentielle pour en réduire le volume dans le flux de sortie. Ce processus sera appelé codage d'ordre.

Le protocole AS supporte un format de données de phototrame unique, pouvant être compressé en utilisant un algorithme de compression de phototrame au fil de l'eau à deux dimensions.

Comme elle dépend de l'activité de traçage de l'application locale dans les fenêtres hébergées et du contrôle de flux (voir 8.5), une ASCE présentatrice peut utiliser une gamme de stratégies pour réduire le volume du flux de sortie et/ou retenir la capacité à répondre du traçage distant. Les stratégies possibles peuvent être décrites comme suit:

- commuter entre les ordres d'envoi et les données de phototrame:

  là où une application locale est en train de tracer de manière très active dans une fenêtre hébergée et là où l'ASCE effectue une contrepression du contrôle de flux, celle-ci peut préférer accumuler les informations liées à l'activité de traçage plutôt que d'envoyer les ordres, et alors par conséquent d'envoyer des données de phototrame pour les liaisons

accumulées. Ceci peut réduire la fréquence des mises à jour sur les ASCE distantes, mais permettra de minimiser les données en route et assurer que les ASCE distantes suivent l'activité de traçage. Les points de commutation entre les ordres et les phototrames et vice versa dépendront du comportement de traçage de l'application, du contrôle de flux et de l'implémentation de l'ASCE particulière;

- combiner les mises à jour qui se recouvrent – ce processus est nommé *spoiling*:

  là où l'ASCE subit une contrepression du contrôle de flux et là où l'application locale trace une séquence de mises à jour d'ordres ou de données de phototrame se recouvrant ou se masquant, alors la séquence peut (dans certains cas) être réduite en une séquence plus petite ou en une mise à jour unique. Là aussi, ceci peut réduire la fréquence des mises à jour sur les ASCE distantes, et permettra de minimiser les données en route et assurer que les ASCE distantes suivent l'activité de traçage.

Il est demandé à une ASCE hôte de s'assurer que le flux de sortie construit final permet aux ASCE réceptrices de tracer correctement les fenêtres reflets correspondantes. D'une manière identique, là où le flux de sortie s'appuie sur l'information de contrôle (telle qu'une information de palette ou de liste de fenêtres) qui a changé sur le terminal hôte, il appartient à l'ASCE hôte de s'assurer que l'information de contrôle soit envoyée avant l'information de sortie. Toutes les ASPDU de sortie et de contrôle dépendantes d'ordres sont envoyées avec une priorité minimale de sorte que l'ASCE émettrice puisse ordonner ces sorties en toute fiabilité.

5.2.3 Contrôle et entrée

Dans une session AS, le contrôle est géré grâce à une combinaison de président, du protocole de contrôle central et (lorsqu'il est négocié) du protocole de contrôle négocié – Voir 8.19, 8.12 et 8.13 respectivement. Le président a la préférence, en ce sens que lorsqu'une conférence est en mode dirigé, les protocoles de contrôle spécifiques de l'AS ne sont pas utilisés (autrement que pour renforcer le président). Là où le président n'est pas en position de force, alors le contrôle est géré grâce à la combinaison du protocole de contrôle central et (lorsqu'il est négocié) du protocole de contrôle négocié).

Un concept clé commun à ces schémas de contrôle est que, à tout moment dans une session AS, une ASCE a le contrôle et possède le droit de fournir des entrées aux fenêtres hébergées et reflets[2].

Là où une ASCE a le contrôle et est en position de fournir des entrées aux fenêtres hébergées et reflets (ceci dépend si elle-même ou d'autres ASCE sont détachées ou non), alors, lorsqu'un événement d'entrée local apparaît, l'ASCE détermine, sur la base de la structure de sa fenêtre locale et de la structure de la fenêtre partagée sur son moniteur logique, si cet événement est pour une fenêtre locale, hébergée ou reflet. Là où l'événement est pour une fenêtre locale ou hébergée, alors l'ASCE peut différer le traitement de l'événement sur le terminal local. Là où l'événement est pour une fenêtre reflet, alors l'ASCE détermine quelle ASCE homologue possède la fenêtre hébergée correspondante et construit un flux d'entrée approprié pour celle-ci.

Le flux d'entrée est composé d'événements de dispositifs de pointage et de clavier intercalés. Les événements de clavier peuvent être représentés soit comme des points de code ou des codes de clés virtuels. Les événements de dispositif de pointage sont représentés comme un dispositif de pointage logique à trois boutons. Voir 8.18 pour plus d'information.

Là où une ASCE effectue une contrepression du contrôle de flux (voir 8.5) et qu'il existe un niveau élevé d'activité dans les dispositifs de pointage des utilisateurs, l'ASCE peut réduire une séquence

---

[2] La présente Recommandation ne spécifie pas le comportement de contrôle pour ce qui concerne les fenêtres de l'ASCE locales.

d'événements de déplacements de dispositifs de pointage en un seul déplacement. On appelle ce procédé le *spoiling*. Le *spoiling* en entrée peut aussi être appliqué à l'ASCE hôte, là où une séquence d'événements de dispositifs de pointage est mise en file d'attente dans l'ASCE attendant de la réinjecter dans l'environnement du terminal local. Le *spoiling* d'entrées reçues et émises peut réduire la fréquence des mises à jour des déplacements des dispositifs de pointage présentés aux fenêtres hébergées et ainsi, dégrader la fluidité du mouvement du dispositif de pointage. Toutefois, cela minimisera les données en route et permettra aux ASCE de suivre l'activité du dispositif de pointage.

5.2.4 Couleur

Le protocole AS transporte des informations de couleur pour le traitement des ordres et des données de phototrame.

- Pour les ordres (à l'exception des ordres *Memory Blt* et *Memory Three-Ways Blt* – Voir ci-dessous) les informations de couleurs sont encapsulées dans l'ordre lui-même, bien qu'elles puissent être absentes dans les ordres individuels comme résultat du codage d'un ordre.
- Les données de phototrame sont toujours palettisées et interprétées en référence à une palette envoyée précédemment contenant les informations de couleurs.
- Pour les *Memory Blt* et *Memory Three-Ways Blt*, l'ordre contient quelques unes des informations de couleur requises, mais il fait aussi référence aux données de phototrame cachées (comme la source de l'opération). Ces données de phototrame cachées (voir 8.16.7) sont toujours palettisées et interprétées en référence à une table de couleur précédemment cachée (voir 8.16.8) contenant les informations de couleurs.

Dans le mode hérité du protocole AS, les informations de couleurs sont exprimées seulement comme des RVB. Dans le mode de base, les informations de couleurs sont exprimées comme des RVB avec des informations de précision de couleur optionnelle.

Les ASCE peuvent fournir des informations d'espaces de couleurs:

- à l'intérieur de palettes: là où l'espace de couleurs s'applique seulement à cette palette;
- à l'intérieur de tables de couleurs dans les ordres *Cache ColorTable*: là où l'espace de couleurs s'applique seulement à cette table de couleurs mise en antémémoire;
- Dans les ordres *Color Space*: là où l'espace de couleurs s'applique aux ordres qui suivent.

Le protocole AS ne donne pas de mandat pour un mappage de couleur particulière entre l'ASCE et le terminal local, bien que des ASCE pourraient tenter de maintenir la fidélité des couleurs (à l'intérieur des contraintes des capacités négociées) lorsqu'elles construisent et interprètent des informations relatives aux couleurs dans le flux de sortie.

5.2.5 Coordonnées et troncature

Toutes les coordonnées dans le protocole AS sont en coordonnées de moniteur virtuel. Le moniteur virtuel est l'union des tailles des moniteurs des ASCE hôte (à savoir les ASCE qui hébergent des fenêtres) et est vu comme la zone de tracé commune utilisée par les ASCE hôtes. L'origine du moniteur virtuel (à savoir le pixel 0,0) est définie au coin haut gauche.

Une ASCE tronquera toutes les coordonnées et rectangles du protocole AS sur le moniteur virtuel. Le protocole AS ne recommande pas une stratégie de troncature particulière en relation avec la zone d'affichage du terminal local.

Tous les rectangles du protocole AS sont inclus, c'est-à-dire que les coordonnées du pixel le plus à droite et le plus bas représentent des coordonnées qui sont à l'intérieur du rectangle. Par exemple, un rectangle avec les coordonnées (10,10,19,19) a pour taille en pixels 10×10.

6 Utilisation de MCS

Toute la communication T.128 se fera à travers le MCS comme spécifié dans la Recommandation T.122. Le présent paragraphe détaille une utilisation spécifique des services MCS, l'allocation de canal et les priorités de données. La présente Recommandation est conforme aux mécanismes décrits dans la Recommandation T.121 pour ce qui est des opérations propres aux sessions de base dans la norme, aux sessions de base hors norme et à la session d'enregistrement. Tous les autres types de session sont optionnels.

Une ASCE utilise les primitives du service MCS décrites dans le Tableau 6-1 pour attacher et détacher d'un domaine, rejoindre et quitter le canal de l'AS, et envoyer et recevoir des ASPDU.

Tableau 6-1/T.128 – Primitives MCS requises par une ASCE

Primitive MCS	Description
MCS-ATTACH-USER	crée un attachement MCS (à travers un SAP MCS) à un domaine hébergé par le fournisseur MCS. Une confirmation de résultat est renvoyée au demandeur. Si la demande est acceptée, un ID utilisateur est assigné.
MCS-DETACH-USER	détruit un attachement MCS préalablement créé par l'invocation de MCS-Attach-User. Cette primitive peut être demandée par un utilisateur ou initialisée par un fournisseur. Elle délivre une indication à chaque attachement MCS du même domaine. Dans le cas d'une initialisation par le fournisseur, une indication est aussi délivrée à l'attachement détruit.
MCS-CHANNEL-JOIN	utilisé par une application cliente pour rejoindre un canal approprié dont l'utilisation est définie par l'application. Ceci est une condition préalable pour recevoir des données envoyées au canal.
MCS-CHANNEL-LEAVE	utilisé par une application cliente pour quitter un canal rejoint précédemment et ainsi arrêter la réception des données envoyées à ce canal. La primitive peut être initialisée à l'initiative de l'utilisateur (demande seulement) ou à celle du fournisseur (indication à l'utilisateur assigné uniquement).
MCS-SEND-DATA	utilisé pour transmettre des données aux autres membres d'un domaine. Si l'émetteur est un membre du canal de destination, il ne recevra pas ses propres indications de données. Cependant, il recevra des indications de données des autres sources adressées sur ce canal.

Les primitives de requêtes MCS sont dirigées de l'ASCE vers le fournisseur MCS, tandis que les primitives d'indication le sont du fournisseur en direction de l'ASCE. Des détails supplémentaires sur les primitives MCS peuvent être trouvés dans la Recommandation T.122: service de communication multipoint – Définition du Service.

6.1 Utilisation de canal MCS

Le Tableau 6-2 décrit l'usage du canal MCS pour les sessions d'ASCE des types définis dans la Recommandation T.121. Dans le cadre d'une session de base normalisée (voir la Recommandation T.121) utilisant le mode de base du protocole AS, on utilisera les *Channel IDs* indiqués dans le Tableau 6-2 (IDs symboliques décrites). Pour tous les autres types de session, les *Application Registry Resource IDs* montrés dans le tableau seront utilisés pour l'allocation des

canaux dynamiques. Les *Resources IDs* donnés seront codés sous la forme de chaînes de caractères à trois octets T.50 en utilisant les caractères montrés entre quotes dans le Tableau 6-2.

Tableau 6-2/T.128 – Description des canaux AS

Mnémonique	Channel IDs pour canal statique	Application Registry Resource IDs pour canal dynamique	Description
AS-{MCS-USER-ID}-CHANNEL	–	–	certaines ASPDU sont envoyées directement aux ASCE de manière individuelle. Pour faire ceci, on utilise les canaux MCS-USER-ID individuels des ASCE homologues dans le domaine MCS.
AS-CHANNEL	AS-CHANNEL-0	"421"	ce canal garantit à toutes les ASPDU d'être diffusées aux ASCE homologues dans un même domaine.

6.2 Utilisation des services de données MCS

Le Tableau 6-3 liste l'utilisation du service de données MCS MCS-SEND-DATA pour chaque ASPDU (avec des ASPDU spécifiques de mode indiquées lorsque nécessaire). Les ASCE n'utilisent pas le service de données MCS MCS-UNIFORM-SEND-DATA. Ce tableau inclut le canal au travers duquel les données sont envoyées et la priorité de donnée avec laquelle celle-ci est envoyée.

Le protocole AS utilise les trois priorités MCS comme un moyen de regrouper les ASPDU selon le contrôle de flux. Le contrôle de flux est appliqué selon le type de canal et de priorité (voir 8.5). Un tel regroupement d'ASPDU par priorité permet l'application d'un régime de contrôle de flux applicable au groupe comme suit:

- une haute priorité a un contrôle de flux et est utilisée dans le cas des ASPDU d'activation, de contrôle de flux et d'entrée;
- une priorité moyenne n'a pas de contrôle de flux et est utilisée pour les ASPDU de contrôle;
- une basse priorité a un contrôle de flux et est utilisée pour les ASPDU d'information de fenêtre et de sortie.

Toutes les ASPDU spécifiées dans la présente Recommandation sont placées dans le paramètre *Data* de la primitive MCS-SEND-DATA. Les ASPDU sont mises en paquets dans la séquence d'octets que constitue le paramètre *Data* de telle sorte que le bit de tête soit placé dans le bit le plus significatif de chaque octet, et rempli jusqu'au bit le moins significatif de chaque octet.

Tableau 6-3/T.128 – Utilisation des primitives des données MCS pour les ASPDU

ASPDU	Canal	Modes	Priorité
ApplicationPDU	AS-CHANNEL ou AS-{MCS-USER-ID}-CHANNEL (Note 1)	les deux	moyenne
ConfirmActivePDU	AS-CHANNEL	mode hérité	toutes (Note 2)
ControlPDU	AS-CHANNEL	les deux	moyenne
DeactivateAllPDU	AS-CHANNEL	mode hérité	haute
DeactivateOtherPDU	AS-{MCS-USER-ID}-CHANNEL	mode hérité	haute
DeactivateSelfPDU	AS-CHANNEL	mode hérité	haute
DemandActivePDU	AS-CHANNEL	mode hérité	haute
FlowResponsePDU	AS-{MCS-USER-ID}-CHANNEL	les deux	haute
FlowStopPDU	AS-CHANNEL	les deux	haute
FlowTestPDU	AS-CHANNEL	les deux	haute, moyenne ou basse (Note 3)
FontPDU	AS-CHANNEL	les deux	moyenne
InputPDU	AS-CHANNEL	les deux	haute
MediatedControlPDU	AS-CHANNEL ou AS-{MCS-USER-ID}-CHANNEL (Note 4)	les deux	moyenne
PointerPDU	AS-CHANNEL	les deux	moyenne
RemoteSharePDU	AS-{MCS-USER-ID}-CHANNEL	les deux	moyenne
RequestActivePDU	AS-{MCS-USER-ID}-CHANNEL	mode hérité	haute
SynchronizePDU	AS-CHANNEL	les deux	toutes (Note 5)
UpdateCapabilityPDU	AS-{MCS-USER-ID}-CHANNEL	mode hérité	moyenne
UpdatePDU	AS-CHANNEL	les deux	basse
WindowActivationPDU	AS-CHANNEL ou AS-{MCS-USER-ID}-CHANNEL (Note 6)	les deux	basse
WindowListPDU	AS-CHANNEL	les deux	basse

NOTE 1 – *ApplicationPDU* (*NotifyHostedApplications*) est envoyé sur le AS-CHANNEL et *ApplicationPDU* (*UnhostApplication*) est envoyé sur un AS-{MCS-USER-ID}-CHANNEL particulier. Voir 8.9 pour plus d'information.

NOTE 2 – *ConfirmActivePDU* est envoyé avec toutes les priorités. Voir 8.4.1 pour plus d'information.

NOTE 3 – *FlowTestPDU* est envoyé avec la priorité appliquée au contrôle de flux. Voir 8.5 pour plus d'information.

NOTE 4 – *MediatedControlPDU* est envoyé soit sur le AS-CHANNEL ou sur un AS-{MCS-USER-ID}-CHANNEL particulier. Voir 8.13 pour plus d'information.

NOTE 5 – *SynchronizePDU* est envoyé avec les priorités Haute, Moyenne ou Basse selon la demande. Voir 8.6 pour plus d'information.

NOTE 6 – Les indications *WindowActivationPDU* sont envoyées sur le AS-CHANNEL et les requêtes *WindowActivationPDU* sont envoyées sur un AS-{MCS-USER-ID}-CHANNEL particulier. Voir 8.11 pour plus d'information.

7 Utilisation de GCC

Le mode hérité du protocole AS utilisera les procédures d'une Session de Base Hors Norme de la manière spécifiée dans la Recommandation T.121 et utilisera la Clé de Protocole d'Application définie dans l'Annexe B. Le mode de base du protocole AS peut utiliser les procédures définies pour une Session d'Enregistrement, une Session de Base Normalisée, une Session Publique ou une Session Privée de la manière spécifiée dans la Recommandation T.121 et utilisera comme Clé de Protocole d'Application l'Identificateur d'Objet défini dans l'Annexe C.

Toutes les ASCE conformes à la présente Recommandation seront tout d'abord enregistrées de manière active ou inactive dans la Session de Base Hors Norme (pour signaler qu'elles supportent le mode hérité du protocole AS), en utilisant dans les deux cas les procédures définies dans la Recommandation T.121 et resteront enregistrées dans les deux sessions aussi longtemps qu'elles indiqueront qu'elles supportent le protocole AS.

Un équipement terminal préexistant supportant seulement le mode hérité sera enregistré de manière active ou inactive dans la Session de Base Hors Norme en utilisant les procédures définies par la Recommandation T.121 et restera enregistré dans cette session aussi longtemps qu'il donnera cette indication.

Une ASCE ne sera pas enregistrée dans la Session de Base Normalisée du protocole AS à moins que pour chaque nœud d'une conférence il y ait le même nombre d'ASCE enregistrées, soit de manière active ou inactive, dans la Session de Base Hors Norme que dans la Session d'Inscription.

Si la Session de Base Hors Norme est en cours (à savoir, toutes les ASCE sont actives dans le mode hérité), sur réception de l'indication *GCC-conference-roster-report*, lorsque le nombre d'ASCE enregistrées dans la Session d'Inscription est égal au nombre d'ASCE enregistrées dans la Session de Base Hors Norme de la conférence (à savoir toutes les ASCE de la conférence sont conformes), toutes les ASCE deviendront inactives dans le mode hérité, seront enregistrées dans la Session de Base Normalisée et deviendront actives dans le mode de base. Voir 8.4 pour plus d'information sur l'activation d'ASCE.

Si la Session de Base Normalisée est en cours (à savoir, toutes les ASCE sont actives dans le mode de base), sur réception de l'indication *GCC-conference-roster-report*, lorsque le nombre d'ASCE enregistrées dans la Session d'Inscription est inférieur au nombre d'ASCE enregistrées dans la Session de Base Hors Norme de la conférence (à savoir une ou plusieurs ASCE de la conférence ne sont pas conformes à l'équipement terminal préexistant), toutes les ASCE enregistrées dans la Session de Base Normalisée deviendront inactives dans le mode de base, ne seront plus enregistrées dans la Session de Base Normalisée et deviendront actives dans le mode hérité. Voir 8.4 pour plus d'information sur l'activation d'ASCE.

Les règles ci-dessus décrivant comment commuter entre les modes de base et hérité signifient que le mode hérité sera effectif seulement lorsque la conférence contiendra un équipement terminal préexistant ne supportant que le mode hérité du protocole AS. Lorsque toutes les ASCE de la conférence sont conformes à la présente Recommandation, c'est le mode de base qui sera appliqué. Il en résulte que le passage du mode de base au mode hérité se produit lorsque le premier nœud à mode uniquement hérité rejoint une conférence composée entièrement d'ASCE conformes à la présente Recommandation. Réciproquement, le passage du mode hérité au mode de base se produit lorsque le dernier nœud de mode uniquement hérité quitte une conférence qui se trouve ensuite entièrement composée d'ASCE conformes à la présente Recommandation.

Les ASCE supportant le mode de base du protocole AS peuvent être, selon leur choix, enregistrées dans une Session Publique ou une Session Privée en utilisant les procédures définies dans la Recommandation T.121.

8 Spécification de protocole

8.1 Sessions AS

Une session AS est formée d'une ou plusieurs ASCE enregistrées dans une conférence comme décrit dans le paragraphe 7. Les ASCE peuvent rejoindre ou quitter la session AS à tout moment.

8.2 Capacités

Les capacités AS sont collectées dans des jeux de capacités, où chaque jeu est formé de capacités individuelles parentes. Les jeux de capacités sont décrits dans les Tableaux 8-3 à 8-20. Là où ci-après la présente Recommandation fait référence à une capacité individuelle, elle utilise la notation **capability_set.capability**. Par exemple Bitmap.desktopWidth fait référence à la capacité *desktopWidth* dans le jeu de capacité *Bitmap*.

La liste complète des jeux de capacités est appelée capacités combinées d'une ASCE. La liste des capacités combinées contient une copie de chacun des jeux de capacités correspondants dans un ordre quelconque:

- jeu de capacités général: voir 8.2.3;
- jeu de capacités de phototrame: voir 8.2.4;
- jeu de capacités d'ordres: voir 8.2.5;
- jeu de capacités d'antimémoire de phototrame: voir 8.2.7;
- jeu de capacités d'antimémoire de table de couleurs: voir 8.2.8;
- jeu de capacités activation de fenêtres: voir 8.2.9;
- jeu de capacités de contrôle: voir 8.2.10;
- jeu de capacités de curseurs: voir 8.2.11;
- jeu de capacités de partage[3]: voir 8.2.12.

Dans le mode hérité du protocole AS, une ASCE peut définir des extensions de capacités hors normes en ajoutant des jeux de capacités privées et/ou en ajoutant des capacités privées à la fin des jeux de capacités définis. L'interprétation de telles capacités n'est pas couverte pas la présente Recommandation. Lors de l'utilisation du mode hérité du protocole AS, une ASCE devrait ignorer les jeux de capacités hors normes non reconnus ainsi que les extensions de jeux de capacités non reconnues. De la même manière, une ASCE traitera toute capacité privée non fournie par d'autres ASCE comme égale à zéro (pour une valeur entière) ou FALSE (pour une valeur logique ou un indicateur binaire).

Dans le mode de base du protocole AS, une ASCE peut définir des extensions de capacités hors normes en ajoutant un ou plusieurs jeux de capacités hors normes (voir 8.2.13) à ses propres capacités combinées. L'interprétation de telles capacités n'est pas couverte par la présente Recommandation.

---

[3] Le jeu *Share capability set* est défini pour le mode hérité du protocole AS seulement. Il ne fait pas partie des capacités combinées d'une ASCE dans le mode de base du protocole AS.

8.2.1 Distribution des capacités

Pour le mode hérité du protocole AS, une ASCE avertit ses capacités combinées pendant l'activation d'une ASCE sur les ASPDU suivantes:

- *DemandActivePDU*: voir 8.4.1;
- *RequestActivePDU*: voir 8.4.1;
- *ConfirmActivePDU*: voir 8.4.1.

Pour le mode hérité du protocole AS, une ASCE avertit un changement dans un jeu de capacités particulier en utilisant *UpdateCapabilityPDU* (voir 8.2.14).

Pour ce qui concerne le mode de base du protocole AS, l'échange de capacités sera effectué selon la Recommandation T.121, via le mécanisme d'inscription d'application. Une ASCE utilisera les requêtes *GCC-Application-Enroll* pour s'inscrire et annoncer ses capacités combinées en cours. Là où ses capacités changent lors d'une inscription, une ASCE se réinscrira en produisant une requête GCC-Application-Enroll avec l'indicateur *Enroll/Unenroll* mis à *Enroll* incluant les capacités combinées révisées. Les ASCE reçoivent des capacités d'applications réduites ou non, générées à partir des capacités combinées fournies par toutes les ASCE inscrites, via les indications *GCC-Application-Roster-Report*. Une ASCE peut demander de traiter les indications GCC-Application-Roster-Report de nombreuses fois dans une session AS et adhérera aux limites imposées par les capacités rapportées de cette façon.

8.2.2 Négociation de capacités

Dans le mode hérité du protocole AS, les échanges d'ASPDU d'activation d'ASCE (voir 8.4.1) assurent qu'une ASCE possède une copie des capacités de chacune des autres ASCE actives. De plus, une ASCE est responsable du traitement de la négociation de toutes les capacités (à savoir, le GCC n'est pas impliqué dans la distribution ou la négociation des capacités).

Dans le mode de base du protocole AS, les capacités sont distribuées par le GCC. Il existe un petit nombre de capacités devant être annoncées via la liste des capacités non réductibles dans le répertoire – elles sont spécifiées comme telles dans la description de la capacité dans le jeu de capacités du mode de base approprié. Cependant en général, une capacité particulière peut être annoncée via soit la liste des capacités non réductibles ou celles réductibles dans le répertoire. Là où une ASCE annonce une capacité particulière via une liste, elle n'annoncera pas la même capacité via une autre liste. Cependant, on ne force pas à une ASCE à annoncer une capacité – elle peut choisir de ne pas annoncer une capacité particulière dans une liste et faire confiance au comportement de comptage du répertoire GCC pour forcer l'utilisation d'une valeur par défaut (voir ci-dessous).

Cette approche hybride de la distribution des capacités du mode de base est mise en œuvre de telle sorte que les ASCE peuvent annoncer des capacités via les capacités réductibles, ceci ayant pour effet de réduire le trafic GCC relatif au répertoire, mais peuvent aussi annoncer des capacités via les capacités non réductibles lorsque cela est plus approprié. On recommande aux ASCE d'utiliser de préférence les capacités réductibles lorsque c'est possible.

Dans le mode de base du protocole AS, lorsqu'une ASCE annonce une capacité particulière via la liste des capacités non réductibles dans le répertoire, l'ASCE codera la valeur de capacité en utilisant les règles de codage définies au 9.4.

La négociation des capacités d'ASCE pour une capacité particulière dépend de la règle de négociation de capacité définie pour cette capacité, du mode de protocole (à savoir si l'ASCE utilise le mode de base ou légal) et si (dans le mode de base) la capacité est annoncée via les capacités réductibles ou non.

8.2.2.1 Règles de capacités *one* et *info*

Là où la règle de négociation de capacité définie est *one* ou *info*, alors une ASCE traitera les capacités annoncées comme suit.

Dans le mode de base, là où une capacité est négociée en utilisant les règles *one* ou *info*, la capacité sera annoncée en utilisant les capacités non réductibles.

Règle	Traitement d'ASCE requis
one	la valeur de capacité négociée est la valeur annoncée par une ASCE homologue particulière.
info	les capacités peuvent aussi être classées comme informatives – Elles sont fournies seulement à des fins d'information et de diagnostic et ne sont pas négociées.

8.2.2.2 Règles de capacités *max* et *min*

Là où la règle de négociation de capacité définie est *min* ou *max*, alors une ASCE traitera les capacités annoncées sur la base de leurs valeurs candidates comme suit:

- pour le mode hérité du protocole AS, les valeurs candidates correspondent aux valeurs annoncées fournies par chaque ASCE durant l'activation d'ASCE, mais en excluant la valeur annoncée par l'ASCE négociante. S'il existe N ASCE actives, alors il existe N−1 valeurs candidates;
- pour le mode de base du protocole AS, les valeurs candidates sont composées des valeurs de capacités réduites dans le répertoire et de toutes les valeurs non réductibles, mais en excluant la valeur non réductible annoncée par l'ASCE négociante. Le répertoire indique combien de valeurs de capacités d'ASCE annoncées contribuent aux valeurs réduites, ce qui permet à une ASCE de déterminer si toutes les ASCE annoncent une valeur. Aussi, s'il existe N ASCE dans le répertoire, C valeurs réductibles d'ASCE annoncées et NC valeurs non réductibles d'ASCE annoncées, alors:
  - si N = C, toutes les ASCE ont annoncé des valeurs réductibles pour la capacité particulière: il y a une valeur candidate;
  - si N = NC, toutes les ASCE ont annoncé des valeurs non réductibles pour la capacité particulière: il y a NC−1 valeurs candidates – la valeur non réductible annoncée par l'ASCE négociante étant exclue;
  - si N = C + NC, toutes les ASCE ont annoncé des valeurs non réductibles ou réductibles pour la capacité particulière: il y a
    - C + NC−1 valeurs candidates – la valeur non réductible annoncée par l'ASCE négociante étant exclue;
    - C + NC valeurs candidates – incluant la valeur réductible annoncée par l'ASCE négociante.
  - si N > C + NC, les ASCE n'ont pas toutes annoncé une valeur: la valeur de capacité négociée est la valeur par défaut (voir plus bas).

Dans le mode de base, si une ASCE détermine que toutes les ASCE ont annoncé des valeurs réductibles (à savoir le cas N = C ci-dessus), alors la valeur de capacité négociée est une valeur réduite dans le répertoire.

Dans le mode de base, si une ASCE détermine que les ASCE n'ont pas toutes annoncé une valeur (à savoir le cas où N > C + NC ci-dessus), alors la valeur de capacité négociée est la valeur par défaut. Pour des capacités logiques, la valeur par défaut est FALSE (ou non établie). Pour des capacités

entières, la valeur par défaut est spécifiée dans la description de la capacité dans le jeu de capacités du mode de base approprié.

Pour le mode hérité, ou le mode de base dans lequel toutes les valeurs annoncées d'ASCE et le nombre de valeurs candidates est supérieur à 1 (à savoir toutes les ASCE ou certaines ont annoncé des valeurs non réductibles (cas N = NC et N = C + NC ci-dessus), alors l'ASCE applique la règle de négociation de capacité *min* ou *max*, comme suit:

Règle	Traitement d'ASCE requis
min	la valeur de capacité négociée est le minimum des valeurs candidates • pour les valeurs entières, le minimum est la valeur entière la plus basse. • pour les valeurs logiques, le minimum est FALSE. • pour les indicateurs binaires (dans le mode hérité uniquement), chaque indicateur binaire est négocié de manière indépendante et le minimum n'est pas attribué.
max	la valeur de capacité négociée est le maximum des valeurs candidates • pour les valeurs entières, le maximum est la valeur entière la plus élevée. • pour les valeurs logiques, le maximum est TRUE. • pour les indicateurs binaires (dans le mode hérité uniquement), chaque indicateur binaire est négocié de manière indépendante et le maximum est attribué.

8.2.2.3 Règles de capacité de groupe

La négociation de capacités d'ASCE est généralement effectuée sur des valeurs de capacités uniques. Cependant, il y a des cas où un nombre de capacités est négocié en groupe – par exemple, la négociation des capacités relatives au nombre de bits par pixel du jeu de capacités phototrame (voir 8.2.4). Lorsque c'est le cas, une capacité individuelle dans le groupe peut être négociée soit en utilisant l'une des règles de négociation de capacités précédemment décrites – telles que *min* ou *max* – avec la valeur négociée injectée dans la négociation de groupe, ou en injectant sa valeur non négociée directement dans la négociation de groupe.

Là où les capacités sont négociées par groupe, celles-ci sont identifiées comme telles dans le jeu de capacités particulier avec une référence à la règle spéciale utilisée pour négocier le groupe.

Le Tableau 8-1 résume la notation utilisée dans le reste du présent sous-paragraphe pour la description de capacités.

Tableau 8-1/T.128 – Notation de capacités – Classes

Classe	Description
L	logique: une valeur logique. En mode hérité la valeur est TRUE ou FALSE. En mode de base, si la capacité est fournie, la valeur est TRUE. En mode de base, la valeur par défaut est FALSE.
F	indicateur binaire: une collection de valeurs binaires dont chacune est TRUE ou FALSE. Cette classe est autorisée seulement dans les jeux de capacités du mode hérité.
N	entier: une valeur entière signée ou non. Dans le mode de base, la valeur par défaut est spécifiée dans la description du jeu de capacités.
S	chaîne: une chaîne de texte T.50 terminée par NULL. Cette classe est autorisée seulement dans les jeux de capacités du mode hérité.

Le Tableau 8-2 illustre un exemple de négociations de capacités. Chaque rangée montre, pour chaque ASCE (ASCE 1 et 2) la valeur de capacité (pour chaque classe) affectée par l'ASCE lors de l'annonce de ses capacités, les valeurs résultantes pour la règle *one* en relation avec les autres ASCE et la valeur négociée pour les règles *min* et *max* dans les cas où toutes les ASCE annoncent toutes qu'elles utilisent des capacités réductibles d'une part et des capacités non réductibles d'autre part [ce qui donne quatre combinaisons: *min* (C) est la règle *min* et toutes annoncent utiliser des capacités réductibles, *min* (NC) est la règle *min* et toutes annoncent utiliser des capacités non réductibles, etc...][4]. Par exemple:

- ASCE 1 a annoncé la valeur logique TRUE, son indicateur de bit vaut 0x0001 et sa valeur entière vaut 100;
- ASCE 2 a annoncé la valeur logique FALSE, son indicateur de bit vaut 0x0003 et sa valeur entière vaut 300;
- lorsque l'ASCE 1 applique la règle *one* envers l'ASCE 2, elle génère les valeurs annoncées de l'ASCE 2 – à savoir respectivement FALSE, 0x0003 et 300;
- lorsque l'ASCE 1 applique la règle *max* dans le cas où les deux ASCE ont annoncé des capacités réductibles, elle génère le maximum de toutes les valeurs annoncées – à savoir respectivement TRUE et 300;
- lorsque l'ASCE 1 applique la règle *max* dans le cas où les deux ASCE ont annoncé des capacités non réductibles, elle génère le maximum de toutes les valeurs annoncées à l'exception de la sienne – à savoir respectivement TRUE, 0x0003 et 300;
- lorsque l'ASCE 2 applique la règle *one* envers l'ASCE 1, elle génère les valeurs annoncées de l'ASCE 1 – à savoir respectivement TRUE, 0x0001 et 100.

Cet exemple n'est pas exhaustif. Les cas énumérés couvrent quatre des dix possibilités de l'exemple. De plus, il existe des combinaisons ultérieures où une ASCE peut fournir des capacités réductibles ou non.

Tableau 8-2/T.128 – Exemple de négociation de capacités

	ASCE 1						**ASCE 2**					
Classe	**Valeur**	**one ⇒ 2**	**min (C)**	**min (NC)**	**max (C)**	**max (NC)**	**Valeur**	**1⇐ one**	**min (C)**	**min (NC)**	**max (C)**	**max (NC)**
B	T	F	F	F	T	F	F	T	F	T	T	T
F	x0001	x0003	s/o	x0003	s/o	x0003	x0003	x0001	s/o	x0001	s/o	x0001
N	100	300	100	300	300	300	300	100	100	100	300	300
s/o: sans objet												

8.2.3 Jeu de capacités général

Le jeu de capacités général fournit des capacités pour les caractéristiques générales de l'ASCE distributrice. Voir les Tableaux 8-3 et 8-4.

---

[4] La classe indicateur binaire est autorisée seulement dans le mode hérité, où la distribution des capacités via les ASPDU d'activation d'ASCE est équivalente à annoncer des capacités non réductibles. Ainsi, les valeurs résultantes de la classe indicateur binaire pour les cas de capacités réductibles *min* et *max* ne sont pas décrites.

Tableau 8-3/T.128 – Jeu de capacités général (mode hérité)

Capacité	Description	Classe	Règle
OSMajorType	cette capacité indique le type principal du système d'exploitation. Les valeurs autorisées sont les suivantes: • non spécifié • Windows • OS/2 • Macintosh • UNIX/X cette capacité est donnée à des fins de diagnostic et d'information seulement.	N	*info*
OSMinorType	cette capacité indique le type secondaire du système d'exploitation. Les valeurs dépendent de la capacité *OSMajorType* (voir ci-dessus). Les valeurs autorisées sont les suivantes: *OSMajorType* — *OSMinorType* Windows — non spécifié, Windows 3.1x, Windows 95, Windows NT OS/2 — non spécifié, OS/2 Warp (Intel x86), PowerPC Macintosh — non spécifié, Macintosh, PowerPC UNIX/X — non spécifié, Native Server, Pseudo Server non spécifié — non spécifié cette capacité est donnée à des fins de diagnostic et d'information seulement.	N	*info*
protocolVersion	cette capacité spécifie le niveau de version de protocole. La valeur autorisée est 0x200 (indiquant respectivement les versions principales et secondaires 2 et 0)	N	*info*

Tableau 8-3/T.128 – Jeu de capacités général (mode hérité) *(fin)*

Capacité	Description	Classe	Règle
generalCompressionTypes	cette capacité est un ensemble d'indicateurs binaires listant (s'ils existent) les schémas de compression généraux qui sont supportés par cette ASCE. L'interprétation de champ dépend de la capacité négociée *generalCompressionLevel* (voir ci-dessous) comme suit: • si la capacité *generalCompressionLevel* négociée vaut zéro, alors seul le dernier indicateur binaire significatif de ce champ est valable (à savoir le bit 0). • si la capacité *generalCompressionLevel* négociée est supérieure à zéro, alors tous les indicateurs binaires de ce champ sont valables. Voir 8.3.2 pour plus d'information sur la compression générale hors norme.	F	*min*
updateCapabilityFlag	cette capacité indique si cette ASCE peut recevoir le *UpdateCapabilityPDU*. Une valeur à TRUE indique qu'elle peut le recevoir et à FALSE qu'elle ne peut pas. Voir 8.2.14 pour plus d'information.	L	*one*
remoteUnshareFlag	cette capacité indique si cette ASCE peut recevoir un *ApplicationPDU* avec l'action *UnhostApplication*. Une valeur à TRUE indique qu'elle peut le recevoir et à FALSE qu'elle ne peut pas. Voir 8.9 pour plus d'information.	L	*one*
generalCompressionLevel	cette capacité indique quel niveau de traitement du schéma de compression général est supporté par cette ASCE. Voir 8.3.2 pour plus d'information sur la compression générale hors norme.	N	*min*

Tableau 8-4/T.128 – Jeu de capacités général (mode de base)

Capacité (valeur par défaut)	**Description**	**ID**	**Classe**	**Règle**
remoteUnshareFlag	cette capacité indique si cette ASCE peut recevoir un *ApplicationPDU* avec l'action *UnhostApplication*. Voir 8.9 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	1	L	*one*
v42bisCompressionFlag	cette capacité indique si cette ASCE peut recevoir des ASPDU utilisant la compression V.42 *bis*. Voir 8.3.2 pour plus d'information.	2	L	*min*
v42bisNumberCodeWords (défaut: 512)	cette capacité spécifie le nombre total de mots de code devant être utilisés par l'algorithme de compression V.42 *bis*. C'est une limite supérieure pour le paramètre P1 de V.42 *bis*. La Rec. V.42 *bis* n'impose pas de limite supérieure sur sa valeur. Voir 8.3.2 pour plus d'information.	3	N	*min* (Note)
v42bisMaxStringLength (défaut: 6)	cette capacité spécifie la longueur maximale de chaîne donnée en entrée du codeur V.42 *bis*. C'est une limite supérieure pour le paramètre P2 de V.42 *bis*. Pour plus d'information, voir 8.3.2.	4	N	*min* (Note)
NOTE – Les capacités v42bisNumberCodeWords et v42bisMaxStringLength dépendent de la capacité v42bisCompressionFlag. Si le paramètre respectif numberOfEntities renvoyé par l'indicateur GCC-Application-Roster-Report pour les capacités v42bisNumberCodeWords ou v42bisMaxStringLength est égal au paramètre numberOfEntities pour la capacité v42bisCompressionFlag, alors la valeur respective négociée est mise en œuvre, sinon la valeur par défaut de la capacité est mise en œuvre.				

8.2.4 Jeu de capacités de phototrame

Le jeu de capacités de phototrame fournit des capacités pour les caractéristiques orientées phototrame de l'ASCE distributrice. Voir les Tableaux 8-5 et 8-6.

Tableau 8-5/T.128 – Jeu de capacités de phototrame (mode hérité)

Capacité	Description	Classe	Règle
preferredBitsPerPixel	cette capacité indique le format préféré de cette ASCE pour les données phototrame. Les valeurs autorisées sont 1, 4 et 8.	N	groupe (Note 1)
receive1BitPerPixelFlag	cette capacité indique si cette ASCE peut recevoir des données de phototrame à 1 bit par pixel. On demande à une ASCE de pouvoir recevoir des phototrames à un bit par pixel et de mettre ce paramètre à TRUE.	L	groupe (Note 1)
receive4BitsPerPixelFlag	cette capacité indique si cette ASCE peut recevoir des phototrames à 4 bits par pixel. Une valeur à TRUE indique qu'elle peut en recevoir et une valeur à FALSE qu'elle ne peut pas. Là où une ASCE spécifie qu'elle peut en recevoir, alors elle doit aussi recevoir des phototrames à 1 bit par pixel.	L	groupe (Note 1)
receive8BitsPerPixelFlag	cette capacité indique si cette ASCE peut recevoir des phototrames à 8 bits par pixel. Une valeur à TRUE indique qu'elle peut en recevoir et une valeur à FALSE qu'elle ne peut pas. Là où une ASCE spécifie qu'elle peut en recevoir, alors elle doit aussi recevoir des phototrames à 1 et 4 bits par pixel.	L	groupe (Note 1)
desktopWidth	cette capacité spécifie la largeur du moniteur de cette ASCE en pixels.	N	groupe (Note 2)
desktopHeight	cette capacité spécifie la hauteur du moniteur de cette ASCE en pixels.	N	groupe (Note 2)
desktopResizeFlag	cette capacité indique si cette ASCE peut recevoir des *UpdateCapabilityPDUs* contenant un jeu de capacités de phototrame comme résultat d'une mise à jour de la taille de moniteur d'une ASCE homologue. Une valeur à TRUE indique qu'elle peut en recevoir et une valeur à FALSE qu'elle ne peut pas. Voir 8.2.14 pour plus d'information.	L	*one*
bitmapCompressionFlag	cette capacité indique si cette ASCE peut recevoir des données de phototrame compressées dans les ordres *UpdatePDU* (Phototrame) et *CacheBitmap*. Voir 8.17 pour plus d'information sur la compression de phototrame.	L	*min*

NOTE 1 – Les capacités relatives aux bits par pixel sont négociées comme un groupe pour déterminer le *sendingBitsPerPixel* en utilisant l'algorithme décrit au 8.2.4.1.

NOTE 2 – Les capacités de hauteur et largeur de moniteur sont négociées comme un groupe pour déterminer (entre autres) la taille du moniteur virtuel. Voir 8.2.4.2.

Tableau 8-6/T.128 – Jeu de capacités de phototrame (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
preferredBitsPerPixel (défaut: 8)	cette capacité indique le format préféré de cette ASCE pour les données de phototrame. Les valeurs autorisées sont 1, 4 et 8. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	10	N	groupe (Note 1)
receive4BitsPerPixelFlag	cette capacité indique si cette ASCE peut recevoir des phototrames à 4 bits par pixel. Là où une ASCE spécifie qu'elle peut en recevoir, alors elle doit aussi recevoir des bitmaps à 1 bit par pixel.	11	L	groupe (Note 1)
receive8BitsPerPixelFlag	cette capacité indique si cette ASCE peut recevoir des phototrames à 8 bits par pixel. Là où une ASCE spécifie qu'elle peut en recevoir, alors elle doit aussi recevoir des bitmaps à 1 et 4 bits par pixel.	12	L	groupe (Note 1)
desktopWidth (défaut: 640)	cette capacité spécifie la largeur du moniteur de cette ASCE en pixels. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	13	N	groupe (Note 2)
desktopHeight (défaut: 480)	cette capacité spécifie la hauteur du moniteur de cette ASCE en pixels. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	14	N	groupe (Note 2)
bitmapCompressionFlag	cette capacité indique si cette ASCE peut recevoir des données de phototrame compressées dans les ordres *UpdatePDU* (Phototrame) et *CacheBitmap*. Voir 8.17 pour plus d'information sur la compression de phototrame.	15	L	*min*

NOTE 1 – Les capacités relatives aux bits par pixel sont négociées comme un groupe pour déterminer le *sendingBitsPerPixel* en utilisant l'algorithme décrit au 8.2.4.1.

NOTE 2 – Les capacités de hauteur et largeur de moniteur sont négociées comme un groupe pour déterminer (entre autres) la taille du moniteur virtuel. Voir 8.2.4.2.

8.2.4.1 Emission de négociation de capacité de bit par pixel

Les jeux de capacités de phototrame *preferredBitsPerPixel*, *receive1BitPerPixelFlag* (en mode hérité), *receive4BitsPerPixelFlag* et *receive8BitsPerPixelFlag* sont négociés comme un groupe pour déterminer le *sendingBitsPerPixel* utilisé par chaque ASCE lors de l'émission de données de phototrame, de palettes et de tables de couleurs.

Certains types de terminaux peuvent acquérir et restituer des données de phototrame dans une variété de profondeurs de couleurs, mais peuvent avoir une profondeur de couleur préférée (qui normalement correspond à la profondeur de couleur de l'écran du terminal réel) dans laquelle l'acquisition et la restitution sont soit plus efficaces ou peuvent être achevées selon une meilleure une fidélité de couleur. Alors le jeu de capacité Phototrame fournit une variété de capacités permettant à une ASCE d'exprimer:

- sa profondeur de couleur préférée pour la réception de données de phototrame, de palettes et de tables de couleurs;
- d'autres profondeurs de couleur dans lesquelles elle peut recevoir des données de phototrame, des palettes et des tables de couleurs.

L'algorithme de négociation des capacités pour ce groupe détermine un *sendingBitsPerPixel* pour une ASCE particulière utilisant les valeurs des capacités *preferredBitsPerPixel*, *receive4BitsPerPixelFlag* et *receive8BitsPerPixelFlag* annoncées comme suit:

- met *combinedBitsPerPixel* à la valeur minimale de *preferredBitsPerPixel* de cette ASCE et à la valeur maximale de toutes les valeurs des *preferredBitsPerPixel* des autres ASCE;
- si *combinedBitsPerPixel* vaut 1 alors met *sendingBitsPerPixel* à 1;
- sinon si *combinedBitsPerPixel* est inférieur ou égal à 4 et la valeur *receive4BitsPerPixelFlag* négociée à TRUE alors met *sendingBitsPerPixel* à 4;
- sinon si la valeur *receive8BitsPerPixelFlag* négociée est à TRUE alors met *sendingBitsPerPixel* à 8;
- sinon si la valeur *receive4BitsPerPixelFlag* négociée est à TRUE alors met *sendingBitsPerPixel* à 4;
- sinon met *sendingBitsPerPixel* à 1.

Dans le mode hérité du protocole AS, le jeu de capacités Phototrame contient une capacité *receive1BitPerPixelFlag*, mais celle-ci ne joue aucun rôle dans la négociation, comme la contrainte de supporter des profondeurs de couleur moins élevées (à savoir on demande à une ASCE de supporter au moins 1 et 4 bits par pixel si elle en supporte 8) signifie que toutes les ASCE doivent supporter 1 bit par pixel. Dans le mode de base du protocole AS, le jeu de capacités Phototrame ne contient pas de capacité *receive1BitPerPixelFlag*.

La contrainte pour une ASCE de supporter des profondeurs de couleur moins élevées signifie qu'une ASCE conforme à la présente Recommandation ne provoquera pas d'échec dans une négociation de capacités mise en œuvre pour trouver une valeur adaptée.

Une ASCE peut utiliser un algorithme privé alternatif dans certaines circonstances. Par exemple, certains terminaux à 8 bits par pixel ne génèrent pas de données de phototrame à 4 bits par pixel de manière fiable. Là où c'est le cas et que la valeur *receive8BitsPerPixel* vaut TRUE, l'ASCE peut mettre *sendingBitsPerPixel* à 8, même si la valeur *combinedBitsPerPixel* vaut 4. Cependant, là où une ASCE utilise réellement un algorithme privé, elle générera encore une valeur *sendingBitsPerPixel* cohérente avec les valeurs négociées des capacités *receive4BitsPerPixelFlag* et *receive8BitsPerPixelFlag*.

8.2.4.2 Négociation des capacités de taille de moniteur

Les hauteur et largeur de moniteur du jeu des capacités Phototrame sont négociées indépendamment pour déterminer la taille du moniteur virtuel. Voir 5.2.5 pour plus d'information sur le moniteur virtuel.

L'algorithme de négociation de capacités utilise essentiellement la règle *max* (voir 8.2.2), mais les valeurs candidates sont les valeurs de capacité annoncées par toutes les ASCE actives pouvant inclure ou non l'ASCE déterminante.

Par exemple, dans une conférence avec quatre ASCE actives: A, B, C et D où:

- l'ASCE A annonce une largeur *Bitmap.desktopWidth* et une hauteur *Bitmap.desktopHeight* respectivement de 800 et 600;
- l'ASCE B annonce une largeur *Bitmap.desktopWidth* et une hauteur *Bitmap.desktopHeight* respectivement de 1024 et 768;
- l'ASCE C annonce une largeur *Bitmap.desktopWidth* et une hauteur *Bitmap.desktopHeight* respectivement de 1600 et 1200;

- l'ASCE D annonce une largeur *Bitmap.desktopWidth* et une hauteur *Bitmap.desktopHeight* respectivement de 640 et 480;
- alors si les ASCE A et C sont hôtes, les valeurs de capacités de moniteur virtuel négociées sont de 1600 par 1200;
- tandis que si les ASCE B et D sont hôtes, les valeurs de capacités de moniteur virtuel négociées sont de 1024 par 768.

8.2.5 Jeu de capacités d'ordres

Le jeu de capacités d'ordres fournit des capacités pour les caractéristiques des ordres de l'ASCE distributrice.

Voir l'Appendice I pour les valeurs informatives des capacités *Desktop Save* (à savoir *desktopSaveXGranularity*, *desktopSaveYGranularity* and *desktopSaveSize*). Voir les Tableaux 8-7 et 8-8.

Tableau 8-7/T.128 – Jeu de capacités d'ordres (mode hérité)

Capacité	Description	Classe	Règle
terminalDescriptor	cette capacité est une chaîne de texte T.50 terminée par le caractère NULL pouvant être utilisée pour identifier les caractéristiques du terminal local à des fins de diagnostic et d'information.	S	*info*
desktopSaveXGranularity	cette capacité spécifie la granularité en X minimale en pixels pour cette ASCE lors de la réception des ordres de *Desktop Save*. Voir 8.16.17 pour plus d'information.	N	*max* (Note 1)
desktopSaveYGranularity	cette capacité spécifie la granularité en Y minimale en pixels pour cette ASCE lors de la réception des ordres *Desktop Save*. Voir 8.16.17 pour plus d'information.	N	*max* (Note 1)
maximumOrderLevel	cette capacité spécifie le niveau d'ordre maximal supporté dans la capacité *OrderSupport* (voir ci-dessous). Voir 8.2.6 pour plus d'information sur les niveaux d'ordre.	N	*info*
numberFonts	cette capacité est le nombre maximal de polices mappables de cette ASCE, dont les détails sont fournis dans *FontPDU*. Voir 8.8 pour plus d'information sur *FontPDU*.	N	*info* (Note 2)
orderFlags	cette capacité est un ensemble d'indicateurs binaires indiquant le support d'ordre fourni par cette ASCE. Les valeurs d'indicateurs binaires sont définies comme suit: • *Negotiate order support*: cet indicateur doit obligatoirement être positionné. • *Cannot receive orders*: si cet indicateur est positionné, il indique que cette ASCE ne peut recevoir d'ordres. une ASCE devra toujours positionner l'indicateur *Negotiate order support*. Voir 8.16.3 pour plus d'information sur le codage d'ordre.	F	*min* (Note 3)

Tableau 8-7/T.128 – Jeu de capacités d'ordres (mode hérité) *(suite)*

Capacité	Description	Classe	Règle
orderSupport	cette capacité est un tableau de 32 niveaux d'ordre indexé par type d'ordre. Les indices de tableau autorisés sont décrits ci-après. Toutes les autres valeurs de tableau seront mises à zéro. Voir ci-dessous pour plus d'information sur les niveaux d'ordre. **Ordre** — **Index** *Destination Blt Support* — 0 — voir 8.16.4. *Pattern Blt Support* — 1 — voir 8.16.5. *Screen Blt Support* — 2 — voir 8.16.6. *Memory Blt Support* — 3 — voir 8.16.9. *Memory Three Way Blt Support* — 4 — voir 8.16.10. *Text Support* — 5 — voir 8.16.11. *Extended Text Support* — 6 — voir 8.16.12. *Rectangle Support* — 7 — voir 8.16.14. *Line Support* — 8 — voir 8.16.16. *Frame Support* — 9 — voir 8.16.13. *Opaque Rectangle Support* — 10 — voir 8.16.15. *Desktop Save Support* — 11 — voir 8.16.17.	N	*min*
textFlags	cette capacité est un ensemble d'indicateurs binaires indiquant la correspondance de police et les options de texte supportées par cette ASCE. Les valeurs d'indicateurs binaires sont définies comme suit: • *Check font aspect*: si cet indicateur est positionné, il indique que cette ASCE supporte la vérification des aspects horizontaux et verticaux de police pendant la correspondance de police. • *Check font signatures*: si cet indicateur est positionné, il indique que cette ASCE supporte la vérification des signatures de police pendant la correspondance de police. • *DeltaX simulation*: si cet indicateur est positionné, il indique que cette ASCE permet les approximations en Delta X pendant la correspondance de police.	F	*min*

Tableau 8-7/T.128 – Jeu de capacités d'ordres (mode hérité) *(fin)*

Capacité	Description	Classe	Règle
	• *Baseline Start*: si cet indicateur est positionné, il indique que cette ASCE peut recevoir des ordres *Text* et *Extended Text* où la position de départ du texte est spécifiée en relation avec la ligne de base des caractères. voir 8.8 pour plus d'information sur la correspondance. Voir 8.16.11 et 8.16.12 pour plus d'information sur les ordres *Text* et *Extended Text*.		
desktopSaveSize	cette capacité spécifie pour chaque ASCE la taille totale en pixels de l'antémémoire de moniteur de cette ASCE. Voir 8.16.17 pour plus d'information.	N	*min* (Note 1)

NOTE 1 – Comme résultat d'une négociation de capacités, une ASCE doit être capable de recevoir des ordres Desktop Save élaborés en utilisant des granularités en X et Y plus grandes que, ou n'étant pas des multiples exacts de ses capacités *desktopSaveXGranularity* et *desktopSaveYGranularity* annoncées. De la même manière, une ASCE doit être capable d'envoyer des ordres *Desktop Save* là où la taille *desktopSaveSize* négociée n'est pas un multiple exact de ses capacités *desktopSaveXGranularity* et *desktopSaveYGranularity* annoncées.

NOTE 2 – Une ASCE réceptrice peut utiliser la capacité *numberFonts* pour déterminer si elle a suffisamment de place pour stocker les polices de cette ASCE dans les *FontPDUs* associés. La capacité *numberFonts* est une limite supérieure et les *FontPDUs* associés peuvent contenir un nombre d'attributs de polices plus petit ou égal à la capacité *numberFonts*. Voir 8.8 pour plus d'information sur *FontPDU*.

NOTE 3 – Il est obligatoire que l'indicateur binaire de support de l'ordre *Negotiate* soit positionné. Les indicateurs binaires d'ordres *Cannot receive* autorisent une ASCE à annoncer si elle peut ou non recevoir des ordres. Lorsque l'indicateur binaire d'ordre *Cannot receive* n'est pas positionné, il indique que l'ASCE peut recevoir des ordres. Lorsque l'indicateur binaire d'ordre *Cannot receive* est positionné, il indique que l'ASCE ne peut pas recevoir d'ordre, ce qui oblige les ASCE homologues à désactiver complètement l'envoi de mises à jour d'ordre et à envoyer seulement des mises à jour de phototrame.

Tableau 8-8/T.128 – Jeu de capacités d'ordres (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
desktopSaveXGranularity (défaut: 1)	cette capacité spécifie la granularité minimale en X en pixels pour cette ASCE lorsqu'elle reçoit des ordres *Desktop Save*. Voir 8.16.17 pour plus d'information.	20	N	*max* (Note)
desktopSaveYGranularity (défaut: 1)	cette capacité spécifie la granularité minimale en Y en pixels pour cette ASCE lorsqu'elle reçoit des ordres *Desktop Save*. Voir 8.16.17 pour plus d'information.	21	N	*max* (Note)

Tableau 8-8/T.128 – Jeu de capacités d'ordres (mode de base) *(suite)*

Capacité (valeur par défaut)	**Description**	**ID**	**Classe**	**Règle**
desktopSaveSize (défaut: 160,000)	cette capacité indique la taille totale en pixels de l'antémémoire du moniteur de cette ASCE. Voir 8.16.17 pour plus d'information.	22	N	*min* (Note)
checkFontAspectFlag	cette capacité indique si cette ASCE supporte la vérification des aspects horizontaux et verticaux des polices durant la correspondance de police. Voir 8.8 pour plus d'information sur la correspondance de police.	23	L	*min*
checkFontSignaturesFlag	cette capacité indique si cette ASCE supporte la vérification des signatures de polices durant la correspondance de police. Voir 8.8 pour plus d'information sur la correspondance de police.	24	L	*min*
allowDeltaXFlag	cette capacité indique si cette ASCE autorise les approximations en Delta X durant la correspondance de police. Voir 8.8 pour plus d'information sur la correspondance de police.	25	L	*min*
baselineStartFlag	cette capacité indique si cette ASCE peut recevoir des ordres *Text* et *Extended Text* là où la position de départ est spécifiée en relation avec la ligne de base des caractères. Voir 8.16.11 et 8.16.12 pour plus d'information sur les ordres *Text* et *Extended Text*.	26	L	*min*
receiveOrdersFlag	cette capacité indique si cette ASCE peut recevoir des ASPDU (ordres) *UpdatePDU*, à savoir si elle peut recevoir des ordres ou non. Voir 8.16 pour plus d'information sur les ordres.	27	L	*min*
DesktopSaveLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Save Desktop* supporté par cette ASCE. Voir 8.16.17 pour plus d'information sur l'ordre *Desktop Save*.	30	N	*min*
DestinationBltLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Destination Blt* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.14 pour plus d'information sur l'ordre *Destination Blt*.	31	N	*min*
ExtendedTextLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Extented Text* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.12 pour plus d'information sur l'ordre *Extended Text*.	32	N	*min*
FrameLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Frame* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.13 pour plus d'information sur l'ordre *Frame*.	33	N	*min*
LineLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Line* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.16 pour plus d'information sur l'ordre *Line*.	34	N	*min*

Tableau 8-8/T.128 – Jeu de capacités d'ordres (mode de base) *(suite)*

Capacité (valeur par défaut)	**Description**	**ID**	**Classe**	**Règle**
MemoryBltLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Memory Blt* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.9 pour plus d'information sur l'ordre *Memory Blt*.	35	N	*min*
MemoryThreeWayBltLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Memory Three Way Blt* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.10 pour plus d'information sur l'ordre *Memory Three Way Blt*.	36	N	*min*
OpaqueRectangleLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Opaque Rectangle* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.15 pour plus d'information sur l'ordre *Opaque Rectangle*.	37	N	*min*
PatternBltLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Pattern Blt* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.5 pour plus d'information sur l'ordre *Pattern Blt*.	38	N	*min*
RectangleLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Rectangle* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.14 pour plus d'information sur l'ordre *Rectangle*.	39	N	*min*
ScreenBltLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Screen Blt* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.6 pour plus d'information sur l'ordre *Screen Blt*.	40	N	*min*
TextLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Text* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.11 pour plus d'information sur l'ordre *Text*.	41	N	*min*
DesktopOriginLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Desktop Origin* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.18 pour plus d'information sur l'ordre *Desktop Origin*.	42	N	*min*
CacheBitmapLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Cache Bitmap* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.7 pour plus d'information sur l'ordre *Cache Bitmap*.	43	N	*min*
CacheColorTableLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Cache Color Table* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.8 pour plus d'information sur l'ordre *Cache ColorTable*.	44	N	*min*

Tableau 8-8/T.128 – Jeu de capacités d'ordres (mode de base) *(fin)*

Capacité (valeur par défaut)	**Description**	**ID**	**Classe**	**Règle**
ColorSpaceLevel (défaut: 1)	cette capacité indique le niveau d'ordre *Color Space* supporté par cette ASCE. Voir 8.2.6 pour plus d'information sur les niveaux d'ordre et 8.16.19 pour plus d'information sur l'ordre *Color Space*.	45	N	*min*
NOTE – Comme résultat de négociation de capacité, une ASCE doit être capable de recevoir des ordres *Desktop Save* construits en utilisant des granularités en X et Y plus grandes que, ou n'étant pas des multiples exacts de, ses capacités *desktopSaveXGranularity* et *desktopSaveYGranularity* annoncées. D'une manière identique, une ASCE doit être capable d'envoyer des ordres *Desktop Save* là où le *desktopSaveSize* négocié n'est pas un multiple exact de ses capacités *desktopSaveXGranularity* et *desktopSaveYGranularity* annoncées.				

8.2.6 Niveaux d'ordre

Le protocole AS permet des améliorations ultérieures du support d'ordre en utilisant les niveaux d'ordre. Les valeurs de niveau d'ordre sont dans l'intervalle 0..255, avec la valeur 0 indiquant que l'ordre indiqué n'est pas supporté.

Les valeurs de niveau d'ordre dans la capacité *Order.orderSupport* (pour le mode hérité du protocole AS) et dans les paramètres de capacité de niveau d'ordre pour chaque ordre (dans le mode de base du protocole AS) indiquent le niveau d'ordre maximal que l'ASCE distributrice peut recevoir sur la base de chaque ordre. Si une ASCE indique qu'elle peut recevoir le niveau d'ordre N pour un ordre particulier, alors elle sera capable de recevoir des ordres dans l'intervalle 1..N. Il est envisagé que d'autres demandes de partage d'application apparaissent (à savoir reflétant un usage changeant d'ordre dans les terminaux cibles), alors les ordres AS existants pourront être améliorés. Là où c'est le cas, une ASCE peut annoncer un support pour une série d'ordres améliorés et là où dans une conférence avec un groupe d'ASCE actives celles-ci les supportent aussi, les ordres améliorés peuvent être utilisés lors de la construction des ASPDU *UpdatePDU* (Ordres). En résumé, ceci signifie que:

- là où une ASCE supporte un ordre de niveau d'ordre N, elle devra être prête à recevoir des ordres de niveau 1.. N;
- là où une ASCE supporte un ordre de niveau d'ordre N et que le niveau d'ordre négocié est supérieur ou égal à N, alors l'ASCE peut envoyer un ordre de niveau 1.. N;
- là où une ASCE supporte un ordre de niveau d'ordre N et que le niveau d'ordre négocié est inférieur à N, alors l'ASCE peut envoyer un ordre de niveau allant de 1 à la valeur négociée – mais peut ne pas envoyer des ordres de niveau supérieur à la valeur négociée.

Pour le mode hérité du protocole AS, le support de niveau d'ordre n'est pas défini pour les ordres *Desktop Origin*, *Cache Bitmap* et *Cache ColorTable* (voir respectivement 8.16.18, 8.16.7 et 8.16.8). En l'absence de support de niveau d'ordre pour ces ordres:

- si une ASCE supporte l'ordre *Screen Blt* au niveau d'ordre 1 ou au dessus, elle supportera aussi l'ordre *Desktop Origin*;
- si une ASCE supporte les ordres *Memory Blt* ou *Memory Three Way Blt* à un niveau 1 ou au dessus, elle supportera aussi les ordres *Cache Bitmap* et *Cache ColorTable* et s'assurera qu'elle annonce des valeurs valides pour les jeux de capacités *Bitmap Cache* et *ColorTable Cache* (voir 8.2.7 et 8.2.8 pour plus d'information sur les jeux de capacités *Bitmap Cache* et *ColorTable Cache*).

8.2.7 Jeu de capacités d'antémémoire de phototrame (*Bitmap Cache*)

Le jeu de capacités *Bitmap Cache* fournit des capacités pour les caractéristiques d'antémémoire de phototrame de l'ASCE distributrice. Ces capacités sont utilisées pour négocier les valeurs utilisées pour construire les ordres *Cache Bitmap* dans les *UpdatePDUs*. Voir 8.16.7 pour plus d'information sur les ordres *Cache Bitmap*.

Si une ASCE supporte les ordres *Memory Blt* ou *Memory Three Way Blt* au niveau d'ordre 1 ou au dessus, elle supportera l'ordre *Cache Bitmap* (en mode de base au niveau 1 et au dessus) et s'assurera qu'il annonce des valeurs de jeu de capacités *Bitmap Cache* avec:

- des valeurs différentes de zéro pour les capacités *cache1Entries*, *cache2 Entries* et *cache3Entries*;
- des valeurs autorisées pour les capacités *cache1MaximumCellSize*, *cache2MaximumCellSize* et *cache3MaximumCellSize* (comme spécifiées dans les Tableaux 8-9 et 8-10);
- *cache3MaximumCellSize* > = *cache2MaximumCellSize* > = *cache1MaximumCellSize*.

La où une ASCE permet de mettre en antémémoire des phototrames, ces capacités indiquent les tailles de cache de phototrame pour chaque autre ASCE hôte, c'est-à-dire qu'en annonçant ces capacités, l'ASCE a l'obligation de fournir un jeu d'antémémoires de phototrames correspondants aux tailles annoncées par chaque ASCE active dans la conférence hébergeant une fenêtre.

Voir l'Appendice I pour les valeurs informatives des capacités *Bitmap Cache*.

Tableau 8-9/T.128 – Jeu de capacités Bitmap Cache (mode hérité)

Capacité	Description	Classe	Règle
cache1Entries	cette capacité est le nombre exact d'entrées d'antémémoire dans la première zone d'antémémoire.	N	*min*
cache1MaximumCellSize	cette capacité est la taille maximale de cellule en octets de la première zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	N	*min*
cache2Entries	cette capacité est le nombre exact d'entrées d'antémémoire dans la deuxième zone d'antémémoire.	N	*min*
cache2MaximumCellSize	cette capacité est la taille maximale de cellule en octets de la deuxième zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	N	*min*
cache3Entries	cette capacité est le nombre exact d'entrées d'antémémoire dans la troisième zone de cache.	N	*min*
cache3MaximumCellSize	cette capacité est la taille maximale de cellule en octets de la troisième zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	N	*min*

Tableau 8-10/T.128 – Jeu de capacités Bitmap Cache (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
cache1Entries (défaut: 600)	cette capacité est le nombre exact d'entrées d'antémémoire dans la première zone d'antémémoire.	80	N	*min*
cache1MaximumCellSize (défaut: 496)	cette capacité est la taille maximale de cellule en octets de la première zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	81	N	*min*
cache2Entries (défaut: 300)	cette capacité est le nombre exact d'entrées d'antémémoire dans la deuxième zone d'antémémoire.	82	N	*min*
cache2MaximumCellSize (défaut: 2032)	cette capacité est la taille maximale de cellule en octets de la deuxième zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	83	N	*min*
cache3Entries (défaut: 150)	cette capacité est le nombre exact d'entrées d'antémémoire dans la troisième zone d'antémémoire.	84	N	*min*
cache3MaximumCellSize (défaut: 4080)	cette capacité est la taille maximale de cellule en octets de la troisième zone d'antémémoire. La valeur de cette capacité est dans l'intervalle 256..16384.	85	N	*min*

8.2.8 Jeu de capacités d'antémémoire de table de couleurs (*ColorTable Cache*)

Le jeu de capacités *ColorTable Cache* fournit des capacités pour les caractéristiques d'antémémoire de tables de couleurs de l'ASCE distributrice. Ces capacités sont utilisées pour négocier les valeurs utilisées pour construire les ordres *Cache ColorTable* dans les *UpdatePDUs*. Voir 8.16.8 pour plus d'information sur les ordres *Cache ColorTable*, ainsi que les Tableaux 8-11 et 8-12.

Si une ASCE supporte les ordres *Memory Blt* ou *Memory Three Way Blt* au niveau d'ordre 1 ou au dessus, elle supportera l'ordre *Cache ColorTable* (en mode de base au niveau 1 et au dessus) et s'assurera qu'il annonce des valeurs de jeu de capacités *ColorTable Cache* contenant une valeur différente de zéro pour la capacité *colorTableCacheSize*.

Là où une ASCE permet de mettre en antémémoire des tables de couleurs, ces capacités indiquent les tailles d'antémémoire de table de couleurs pour chaque autre ASCE hôte, c'est-à-dire qu'en annonçant ces capacités, l'ASCE a l'obligation de fournir un jeu d'antémémoires de tables de couleurs correspondants aux tailles annoncées par chaque ASCE active dans la conférence hébergeant une fenêtre.

Voir l'Appendice I pour les valeurs informatives des valeurs de jeu de capacités *ColorTable Cache*.

Tableau 8-11/T.128 – Jeu de capacités ColorTable Cache (mode hérité)

Capacité	Description	Classe	Règle
colorTableCacheSize	cette capacité spécifie le nombre d'entrées de table de couleurs dans le cache de table de couleurs de cette ASCE réceptrice. Là où une ASCE supporte de mettre en antémémoire une table de couleurs, les valeurs autorisées sont comprises dans l'intervalle 1..255 (zéro n'est pas autorisé). Voir 8.16.8 pour plus d'information sur la gestion d'antémémoire.	N	*min*

Tableau 8-12/T.128 – Jeu de capacités ColorTable Cache (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
colorTableCacheSize (défaut: 6)	cette capacité spécifie le nombre d'entrées de table de couleurs dans le cache de table de couleurs de cette ASCE réceptrice. Là où une ASCE supporte de mettre en antémémoire une table de couleurs, les valeurs autorisées sont comprises dans l'intervalle 1..255 (zéro n'est pas autorisé). Voir 8.16.8 pour plus d'information sur la gestion d'antémémoire.	90	N	min

8.2.9 Jeu de capacités activation de fenêtres (*Window Activation*)

Le jeu de capacités *Window Activation* fournit des capacités pour les caractéristiques d'activation de fenêtre de l'ASCE distributrice, et en particulier sur la nature de son support pour les messages d'activation *WindowActivationPDU* spécifiques. Voir 8.11 pour plus d'information sur les *WindowActivationPDUs*, ainsi que les Tableaux 8-13 et 8-14.

Tableau 8-13/T.128 – Jeu de capacités Window Activation (mode hérité)

Capacité	Description	Classe	Règle
helpKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpKey*. Une valeur à TRUE indique qu'elle peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpKey* et une valeur à FALSE qu'elle ne le peut pas. Voir 8.9 pour plus d'information.	L	*one*
helpIndexKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpIndexKey*. Une valeur à TRUE indique qu'elle peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpIndexKey* et une valeur à FALSE qu'elle ne le peut pas. Voir 8.9 pour plus d'information.	L	*one*
helpExtendedKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpExtendedKey*. Une valeur à TRUE indique qu'elle peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpExtendedKey* et une valeur à FALSE qu'elle ne le peut pas. Voir 8.9 pour plus d'information.	L	*one*
windowManagerMenuFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *WindowManagerMenu*. Une valeur à TRUE indique qu'elle peut recevoir des *WindowActivationPDUs* contenant l'action *WindowManagerMenu* et une valeur à FALSE qu'elle ne le peut pas. Voir 8.9 pour plus d'information.	L	*one*

Tableau 8-14/T.128 – Jeu de capacités Window Activation (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
helpKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpKey*. Voir 8.9 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	100	L	*one*
helpIndexKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpIndexKey*. Voir 8.9 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	101	L	*one*
helpExtendedKeyFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *ActivationHelpExtendedKey*. Voir 8.9 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	102	L	*one*
windowManagerMenuFlag	cette capacité indique si cette ASCE peut recevoir des *WindowActivationPDUs* contenant l'action *WindowManagerMenu*. Voir 8.9 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	103	L	*one*

8.2.10 Jeu de capacités de contrôle (*Control*)

Le jeu de capacités *Control* fournit des capacités pour les caractéristiques de contrôle de l'ASCE distributrice. Ces capacités sont utilisées pour négocier les valeurs utilisées entre les ASCE dans la gestion du contrôle et de l'état détaché. Voir 8.12 pour plus d'information sur les caractéristiques de contrôle, ainsi que les Tableaux 8-15 et 8-16.

Tableau 8-15/T.128 – Jeu de capacités Control (mode hérité)

Capacité	Description	Classe	Règle
controlFlags	cette capacité est un ensemble d'indicateurs binaires indiquant les options supportées par cette ASCE. Les valeurs d'indicateur binaire sont définies comme suit: • *Allow Mediated Control* si cette ASCE ne positionne pas la capacité *Allow Mediated Control*, l'ASCE ne supporte pas le protocole *control mediated* de l'AS. Là où c'est le cas, les capacités *remoteDetachFlag*, *controlInterest* et *detachInterest* ont besoin d'être négociées et le *MediatedControlPDU* n'est pas valide. Voir 8.13 pour plus d'information.	F	*min*
remoteDetachFlag	cette capacité indique si cette ASCE autorise les autres ASCE à la forcer dans le mode *detached control*. Une valeur à TRUE indique que cette ASCE l'autorise et à FALSE qu'elle ne l'autorise pas. Voir 8.13 pour plus d'information.	L	*one*
controlInterest	cette capacité indique le comportement de cette ASCE pour ce qui concerne les changements de contrôle. Les valeurs autorisées sont: • *Always* cette ASCE permet toujours les changements de contrôle; • *Confirm* cette ASCE demande de confirmer les changements de contrôle; • *Never* cette ASCE n'autorise jamais les changements de contrôle. Voir 8.13 pour plus d'information.	N	*max* (Note)
detachInterest	cette capacité indique le comportement de cette ASCE pour ce qui concerne les changements vers l'état *Detach*. Les valeurs autorisées sont: • *Always* cette ASCE permet toujours les changements dans l'état *detach*; • *Confirm* cette ASCE demande toujours de confirmer les changements dans l'état *detach*; • *Never* cette ASCE ne permet jamais les changements dans l'état *detach*. Voir 8.13 pour plus d'information.	N	*max* (Note)
NOTE – Les capacités *controlInterest* et *detachInterest* sont négociées en utilisant des valeurs telles que *Never* soit plus grande que *Confirm* elle-même plus grande que *Always*.			

Tableau 8-16/T.128 – Jeu de capacités Control (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
mediatedControlFlag	cette capacité indique si l'ASCE supporte le protocole *Control Mediated*. Là où c'est le cas, les capacités *remoteDetachFlag*, *controlInterest* et *detachInterest* ont besoin d'être négociées et le *MediatedControlPDU* n'est pas valide. Voir 8.13 pour plus d'information.	110	L	*min*
remoteDetachFlag	cette capacité indique si cette ASCE autorise les autres ASCE à la forcer dans le mode *detached control*. Voir 8.13 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	111	L	*one*
controlInterest (défaut: *always*)	cette capacité indique le comportement de cette ASCE pour ce qui concerne les changements de contrôle. Les valeurs autorisées sont: • *Always* cette ASCE permet toujours les changements de contrôle; • *Confirm* cette ASCE demande de confirmer les changements de contrôle;	112	N	*max* (Note)
	• *Never* cette ASCE n'autorise jamais les changements de contrôle. Voir 8.13 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.			
detachInterest (défaut: *always*)	cette capacité indique le comportement de cette ASCE pour ce qui concerne les changements vers l'état *detach*. Les valeurs autorisées sont: • *Always* cette ASCE permet toujours les changements dans l'état *detach*; • *Confirm* cette ASCE demande toujours de confirmer les changements dans l'état *detach*; • *Never* cette ASCE ne permet jamais les changements dans l'état *detach*. voir 8.13 pour plus d'information. Cette capacité doit être annoncée en utilisant les capacités non réductibles.	113	N	*max* (Note)
NOTE – Les capacités *controlInterest* et *detachInterest* sont négociées en utilisant des valeurs telles que *Never* soit plus grande que *Confirm* elle-même plus grande que *Always*.				

8.2.11 Jeu de capacités de curseurs (*Pointer*)

Le jeu de capacités *Pointer* fournit des capacités pour les caractéristiques des curseurs de l'ASCE distributrice. Ces capacités sont utilisées pour négocier les valeurs utilisées pour construire les *PointersPDUs*. Voir 8.14 pour plus d'information sur les curseurs ainsi que les Tableaux 8-17 et 8-18.

Là où une ASCE ne supporte pas les curseurs de couleur, ou là où elle les supporte mais ne souhaite pas supporter la mise en antémémoire de curseur de couleur, alors elle devrait annoncer la règle *one* dans son *Pointer.colorPointercacheSize*. Ceci sera interprété comme indiquant que l'ASCE annonçante ne peut que se rappeler du dernier curseur de couleur ou monochrome (ceci dépendant de

la valeur négociée du *Pointer.colorPointerFlag*) – à savoir qu'elle ne supporte pas la mise en antémémoire de curseur.

Là où une ASCE supporte les curseurs de couleur ainsi que leur mise en antémémoire, ces capacités indiquent la taille de l'antémémoire de curseur de couleur pour chaque ASCE hôte. Il en résulte qu'en annonçant ces capacités, l'ASCE s'oblige à fournir une antémémoire de curseur de couleur de la taille annoncée pour chaque ASCE active dans la conférence hébergeant des fenêtres.

Voir l'Appendice I pour les valeurs informatives des capacités *Pointer*.

Tableau 8-17/T.128 – Jeu de capacités Pointer (mode hérité)

Capacité	Description	Classe	Règle
colorPointerFlag	cette capacité indique si cette ASCE supporte les curseurs de couleur. Une valeur à TRUE indique qu'elle les supporte et une valeur à FALSE indique que non. Voir 8.14 pour plus d'information.	L	*min*
colorPointerCacheSize	cette capacité spécifie le nombre d'entrées dans l'antémémoire de curseurs de couleur en réception de cette ASCE. Les valeurs autorisées sont comprises dans l'intervalle 1..500. Là où une ASCE ne supporte pas les curseurs de couleur, elle positionnera cette capacité à 1 (la valeur zéro n'est pas autorisée). Voir 8.14 pour plus d'information sur la mise en antémémoire de curseur.	N	*min*

Tableau 8-18/T.128 – Jeu de capacités Pointer (mode de base)

Capacité (valeur par défaut)	Description	ID	Classe	Règle
colorPointerFlag	cette capacité indique si cette ASCE supporte les curseurs de couleur. Voir 8.14 pour plus d'information.	120	L	*min*
colorPointerCacheSize (défaut: 25)	cette capacité spécifie le nombre d'entrées dans l'antémémoire de curseurs de couleur en réception de cette ASCE. Les valeurs autorisées sont comprises dans l'intervalle 1..500. Là où une ASCE ne supporte pas les curseurs de couleur, elle positionnera cette capacité à 1 (la valeur zéro n'est pas autorisée). Voir 8.14 pour plus d'information sur la mise en antémémoire de curseur.	121	N	*min*

8.2.12 Jeu de capacités de partage (*Share*)

Le jeu de capacités *Share* fournit des informations concernant le nœud sur lequel s'exécute l'ASCE. Ce jeu de capacités est seulement supporté dans le mode hérité du protocole AS.

Le *GCC Node ID* peut être utilisé par les ASCE réceptrices pour corréler les ASCE distributrices avec leurs enregistrements d'application dans le répertoire *GCC Application Roster*. Voir le Tableau 8-19.

Tableau 8-19/T.128 – Jeu de capacités Share (mode hérité)

Capacité	Description	Classe	Règle
nodeID	cette capacité est le *GCC node ID* du nœud de l'ASCE annonçante.	N	*info*

8.2.13 Jeu de capacités hors norme

Le jeu de capacités hors normes est utilisé pour négocier des fonctions hors normes. N'importe quel numéro de capacités hors normes peut apparaître dans le jeu de capacités hors normes tant qu'ils ont chacun un identificateur *Non-Standard-Identifier* unique. L'interprétation de ces capacités n'est pas couverte par la présente Recommandation.

Le jeu de capacités hors normes est supporté seulement dans le mode de base du protocole AS. Il n'y a aucun mécanisme dans le mode hérité pour négocier des capacités hors normes. Voir le Tableau 8-20.

Tableau 8-20/T.128 – Jeu de capacités hors normes (mode de base)

Capacité	Description	ID	Classe	Règle
nonStandardCapability	cette capacité est utilisée pour négocier des fonctions hors normes. N'importe quel nombre de capacités hors normes peut apparaître dans le jeu de capacités tant qu'ils ont chacun un identificateur *Non-Standard-Identifier* unique. L'interprétation de ces capacités n'est pas couverte par la présente Recommandation.	identificateur hors norme	–	–

8.2.14 Mise à jour de capacités

Dans le mode de base du protocole, là où les capacités de l'ASCE changent alors qu'elles sont inscrites, cette dernière s'inscrira à nouveau en distribuant une demande *GCC-Application-Enroll* pour s'inscrire dans la session *Standard Base Session* avec l'indicateur *Enroll/Unenroll* mis à *Enroll*, incluant les capacités combinées révisées. Voir le paragraphe 7 pour plus d'information sur l'inscription de session et 8.2.1 pour plus d'information sur la distribution des capacités.

Pour le mode hérité du protocole AS, une ASCE peut annoncer des changements dans un jeu de capacités en envoyant un *UpdateCapabilityPDU* aux autres ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *UpdateCapabilityPDU* est montré dans le Tableau 8-21.

Une ASCE enverra un *UpdateCapabilityPDU* seulement là où l'indicateur *General.updateCapabilityFlag* et les capacités *Bitmap.desktopResizeFlag* négociés sont tous à TRUE.

Le seul jeu de capacités autorisé pouvant être annoncé dans un *UpdateCapabilityPDU* est le jeu de capacités *Bitmap* du mode hérité (voir 8.2.4).

Tableau 8-21/T.128 – UpdateCapabilityPDU

Paramètre	Description
ShareData Header	le *ShareData Header* est décrit au 8.3.
updateCapabilitySet	ce paramètre sera un jeu de capacité de phototrame. Voir 8.2.4 pour une description du jeu de capacités *Bitmap*.

Quand une ASCE détermine qu'une ou plusieurs capacités ont été mises à jour, elle traitera n'importe quelle négociation de capacité demandée (voir 8.2.2) et traitera la synchronisation d'hôte (si elle est hôte) et de reflet (voir 8.6).

8.3 Formats ASPDU

Dans le mode hérité du protocole AS, toutes les ASPDU autres que les ASPDU de contrôle de flux suivantes contiennent un *ShareControl Header*. Le Tableau 8-22 décrit le *ShareControl Header* pour le mode hérité du protocole AS. Dans le mode de base du protocole AS, les ASPDU ne contiennent pas de *ShareControl Headers*.

- *FlowTestPDU*: voir 8.5.
- *FlowResponsePDU*: voir 8.5.
- *FlowStopPDU*: voir 8.5.

Tableau 8-22/T.128 – ShareControl Header (mode hérité)

Paramètre	Description
totalLength	c'est la longueur totale en octets de l'ASPDU à l'intérieur de laquelle cet en-tête est inclus. Ce paramètre est demandé puisque les implémentations MCS peuvent segmenter les ASPDU en transmission et ne sont pas obligées de les recombiner lors de la livraison. Ce paramètre permet aux ASCE réceptrices de traiter efficacement la recombinaison lors d'une segmentation de MCS.
protocolVersion	ce paramètre identifie la version de protocole supportée par l'ASCE distributrice. La valeur autorisée est 1.
PDUSource	ce paramètre est le *MCS User ID* de l'ASCE envoyant l'ASPDU contenant ce *ShareControl Header*.

Toutes les ASPDU autres que les ASPDU d'activation d'ASCE et de contrôle de flux contiennent un *ShareData Header* et sont référencées comme des ASPDU de données.

- *DemandActivePDU*: voir 8.4.1.
- *ConfirmActivePDU*: voir 8.4.1.
- *RequestActivePDU*: voir 8.4.1.
- *DeactivateOtherPDU*: voir 8.4.1.
- *DeactivateSelfPDU*: voir 8.4.1.
- *DeactivateAllPDU*: voir 8.4.1.
- *FlowTestPDU*: voir 8.5.
- *FlowResponsePDU*: voir 8.5.
- *FlowStopPDU*: voir 8.5.

Les Tableaux 8-23 et 8-24 décrivent le *ShareData Header* pour les modes héritées et de base du protocole AS.

Tableau 8-23/T.128 – ShareData Header (mode hérité)

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre identifie de manière unique la session ASCE dans laquelle cette ASPDU est distribuée. Voir 8.4.2 pour une description des identificateurs partagés.
streamID	ce paramètre identifie le flux pour cette ASPDU. Voir 8.3.1 pour une description des flux.
uncompressedLength	ce paramètre est la longueur en octets des données ASPDU non compressées à partir du paramètre *generalCompressedType* inclus. Ce paramètre peut être utilisé comme une vérification sur la décompression. Voir 8.3.2 pour plus d'information sur la compression générale hors norme.
generalCompressedType	ce paramètre indique si l'ASCE distributrice a compressé entièrement l'ASPDU contenant ce *ShareData Header* et dans l'affirmative, indique quel schéma de compression général a été appliqué. L'interprétation de ce champ dépend de la capacité *General.generalCompressionLevel* négociée comme suit: • si la capacité *General.generalCompressionLevel* négociée vaut zéro, alors l'ASCE référencera uniquement le bit le plus significatif de ce champ; • si la capacité *General.generalCompressionLevel* négociée est supérieure à zéro, alors l'ASCE référencera tous les bits de ce champ. une valeur à zéro indique qu'aucune compression générale n'a été appliquée. Une valeur différente de zéro spécifie le type de compression particulier qui a été appliqué. Voir 8.3.2 pour plus d'information sur la compression générale.
generalCompressedLength	là où le *generalCompressedType* est différent de zéro, ce paramètre est la longueur en octets des données compressées, à partir du paramètre *uncompressedLength*. Là où le *generalCompressedType* vaut zéro, indiquant qu'aucune compression générale n'a été appliquée, ce paramètre sera égal à zéro. Voir 8.3.2 pour plus d'information sur la compression générale.

Tableau 8-24/T.128 – ShareData Header (mode de base)

Paramètre	Description
shareID	ce paramètre est l'instance de répertoire de la session ASCE à l'intérieur de laquelle cette ASPDU est distribuée. Voir 8.4.2 pour une description des identificateurs de partage.
generalCompressionSpecifier	ce paramètre optionnel identifie le schéma de compression générale et tous les paramètres associés applicables à cette ASPDU. Voir 8.3.2 pour plus d'information.

8.3.1 Flux (*Streams*)

Dans le mode hérité du protocole AS, on demande à une ASCE d'envoyer des ASPDU de données via un ensemble de flux logiques. Les flux sont utilisés pour représenter des priorités de données ASCE et correspondent à des priorités MCS comme décrit dans le Tableau 8-25. Les identificateurs de flux sont présents dans toutes les ASPDU de données du mode hérité. Les flux ne sont pas supportés dans le mode de base du protocole AS.

Tableau 8-25/T.128 – Identificateurs de flux (mode hérité)

Stream ID	MCS Priority
4	haute
2	moyenne
1	basse

8.3.2 Compression générale

La présente Recommandation spécifie les schémas de compression générale pouvant être appliqués aux ordres (appelés codage d'ordre – Voir 8.16.3) et aux données de phototrame (appelés compression de phototrame – Voir 8.17) dans les *UpdatePDU*. Ces deux schémas de compression délivrent de bons rapports de compression sur des données de sortie d'AS typiques pour des charges de traitement relativement faibles.

Cependant, là où une ASCE effectue une compression locale de contrôle de flux (voir 8.5) – Par exemple lors de l'envoi via des liens plus lents comme des lignes RTPC, ou lors de l'envoi via liens congestionnés à plus hauts débits – L'expérience montre que des améliorations significatives des performances peuvent être accomplies en appliquant aux ASPDU sélectionnées un schéma de compression à base de dictionnaire – Incluant les données déjà compressées dans les ASPDU *d'UpdatePDU* (Ordres) et *UpdatePDU* (Phototrame). Cette approche est dénommée compression générale.

8.3.2.1 Compression générale en mode hérité

La présente Recommandation ne spécifie pas un mécanisme de compression générale pour le mode hérité, mais fournit plutôt un mécanisme indépendant du schéma de compression dans lequel les ASCE peuvent négocier et utiliser des schémas de compression générale mutuellement acceptés, comme suit.

Une ASCE négociera le schéma de compression générale en utilisant les capacités *General.generalCompressionLevel* et *General.generalCompressionTypes*. Voir 8.2.3 pour plus d'information sur le jeu de capacités *General*. La capacité *GeneralCompressionLevel* indique quel niveau de compression générale est supporté par une ASCE tandis que *General.generalCompressionTypes* est un jeu d'indicateurs binaires permettant à une ASCE d'indiquer si elle supporte zéro, un ou une sélection de schémas de compression générale.

Il faut noter que l'interprétation de la capacité *General.generalCompressedTypes* dépend de la capacité *General.generalCompressionLevel* négociée. Là où la capacité négociée *General.generalCompressionLevel* vaut zéro, alors l'indicateur binaire le moins significatif de la capacité *General.generalCompressedTypes* sera pris en compte. Là où la capacité négociée *generalCompressionLevel* est supérieure à zéro, alors tous les indicateurs binaires à l'intérieur de la capacité *General.generalCompressedTypes* seront pris en compte.

Là où une ASCE détermine que d'autres ASCE homologues peuvent recevoir un schéma de compression générale particulier, celle-ci peut appliquer sélectivement une compression générale aux ASPDUs de données, c'est-à-dire qu'une ASCE peut appliquer une compression générale à zéro, une ou toutes les ASPDUs. Les critères par lesquels une ASCE détermine si une compression générale est appropriée pour une ASPDU de données particulière peuvent inclure l'état du contrôle de flux (voir 8.5) et la taille de l'ASPDU particulière. La présente Recommandation ne préconise pas d'approche particulière pour l'application de la compression générale; il s'agit d'un choix du ressort local effectué par chaque ASCE.

Là où une ASCE n'applique pas une compression générale à une ASPDU de données, soit parce que aucun schéma de compression générale adéquat n'est disponible à la suite de la négociation de capacités ou bien, là où un schéma de compression générale adéquat a été négocié mais que l'ASCE émettrice détermine que la compression générale n'est pas appropriée, l'ASCE affectera *generalCompressedType* à zéro pour indiquer que l'ASPDU n'a pas subi de compression générale, le paramètre *uncompressedLength* à la longueur d'octets de l'ASPDU et le paramètre *generalCompressedLength* à zéro. Là où une ASCE applique une compression générale à une ASPDU de données, il affectera le paramètre *generalCompressedType* à une valeur de schéma de compression générale négociée pour indiquer que l'ASPDU a subi une compression générale, le paramètre *uncompressedLength* à la longueur en octets avant compression de l'ASPDU et le paramètre *generalCompressedLength* à la longueur en octets après compression de l'ASPDU. Le fait d'affecter à la fois les paramètres *uncompressedLength* et *generalCompressedLength* permet à une ASCE réceptrice de vérifier que la longueur de l'ASPDU après décompression est cohérente avec sa longueur non compressée d'origine.

Il faut noter que l'interprétation du paramètre *generalCompressedType* dans le *ShareData Header* dépend de la capacité négociée *General.generalCompressionLevel*. Là où la capacité négociée *General.generalCompressionLevel* vaut zéro, alors seul le bit le plus significatif du paramètre *General.generalCompressedType* sera pris en compte. Là où la capacité négociée *General.generalCompressionLevel* est supérieure à zéro, alors tous les bits du paramètre *General.generalCompressedType* seront pris en compte.

Voir l'Appendice I pour les valeurs informatives et des informations sur la compression générale en mode hérité.

8.3.2.2 Compression générale en mode de base

La présente Recommandation spécifie l'utilisation optionnelle de V.42 *bis* pour la compression générale en mode de base.

Là où une ASCE détermine que d'autres ASCE homologues peuvent recevoir des ASPDU ayant subi une compression générale selon V.24 *bis*, il peut appliquer sélectivement une compression générale V.42 *bis* aux ASPDU de données, c'est-à-dire qu'une ASCE peut appliquer la compression générale V.42 *bis* à zéro, tout ou partie des ASPDU. Les critères par lesquels une ASCE détermine si une compression générale V.42 *bis* est appropriée pour une ASPDU de données particulière peuvent inclure l'état du contrôle de flux (voir 8.5) et la taille de l'ASPDU particulière. La présente Recommandation ne recommande pas une approche particulière pour l'application de la compression générale V.42 *bis* – Ceci est laissé localement à l'initiative de chaque ASCE.

Là où une compression générale V.42 *bis* est appliquée, le codeur est initialisé pendant la synchronisation d'hôte de l'émetteur et le décodeur est initialisé pendant la synchronisation reflet des récepteurs (voir 8.6.2 et 8.6.3).

Là où une ASCE applique une compression générale V.42 *bis* à une ASPDU de données particulière, elle affectera le paramètre *generalCompressionSpecifier* pour indiquer que l'ASPDU a subi une compression générale V.42 *bis*. Si l'ASPDU est la première à avoir subi une compression générale

V.42 *bis* à la suite de l'initialisation du codeur V.42 *bis*, alors l'ASCE pourra aussi inclure les paramètres V.42 *bis* P1 et P2 optionnels dans le paramètre *CompressionSpecifier* général pour permettre aux ASCE homologues d'optimiser les ressources relatives à la compression générale.

8.4 Activation d'ASCE

A l'intérieur d'une session, une ASCE peut être dans l'un des trois états d'activation suivants:

- une ASCE inactive ne participe pas au partage de fenêtres (à savoir, elle n'héberge pas de fenêtre, ne trace pas de fenêtre reflet ou n'envoie ni ne reçoit d'entrée AS). Lorsqu'une ASCE est inactive, elle n'a pas besoin de retenir des ressources locales requises pour l'hébergement et/ou le partage;
- une ASCE active en instance est en phase de devenir active et ne participe pas dans le partage des fenêtres (à savoir, elle n'héberge pas de fenêtre, ne trace pas de fenêtre reflet ou n'envoie ni ne reçoit d'entrée);
- une ASCE active coopère avec les autres ASCE actives en partageant des fenêtres.

Les sous-paragraphes suivants décrivent l'activation d'ASCE et la manipulation des identificateurs de partage pour les modes de base et hérités.

8.4.1 Activation d'ASCE (mode hérité)

Dans le mode hérité de l'activation d'ASCE et de partage du protocole AS, la manipulation d'identificateur est grandement indépendante de GCC et est effectuée en utilisant les éléments de protocole AS spécifiques.

Une ASCE rejoint la session AS de base hors norme (voir le paragraphe 7) en utilisant une requête *GCC-Application-Enroll* avec l'indicateur *Active/Inactive* mis à *Active* et avec des capacités limitées. Ceci ne rend pas l'ASCE active (en relation avec les états de la session AS décrits ci-dessus) mais assure réellement à l'ASCE une visibilité des autres ASCE dans la conférence.

Une ASCE peut rester inscrite dans la session de base hors norme durant une certaine période tout en passant plusieurs fois de l'état actif à l'état inactif et vice versa. Cette distinction entre inscription et activation permet à une ASCE de différer la mise en œuvre de ressources spécifiques de partage d'application à ces moments là dans la conférence où le partage d'application a réellement lieu.

Une ASCE désirant devenir active enverra soit un *DemandActivePDU* ou un *RequestActivePDU* à toutes les ASCE dans la conférence selon la manière indiquée dans le Tableau 6-3. Le contenu de *DemandActivePDU* est décrit dans le Tableau 8-26 et le contenu *RequestActivePDU* dans le Tableau 8-27.

Il est recommandé qu'une ASCE envoie initialement un *RequestActivePDU* à toutes les ASCE dans la conférence pour déterminer s'il existe d'autres ASCE actives dans la session AS. Si aucune ne répond au *RequestActivePDU*, alors à un moment ultérieur, lorsque l'ASCE désirera commencer le partage, elle devrait envoyer un *DemandActivePDU* à toutes les ASCE dans la conférence pour initialiser l'activation.

Le fait d'envoyer un *DemandActivePDU* requiert que l'ASCE émettrice ait généré un identificateur de partage unique, tandis qu'un *RequestActivePDU* ne le requiert pas. Le fait d'envoyer un *DemandActivePDU* peut obliger à un changement dans l'identificateur de partage accepté en cours. Voir 8.4.2 pour plus d'information sur les identificateurs de partage.

Sur réception d'un *DemandActivePDU*, conditionné à une négociation réussie de capacités (voir 8.2), une ASCE peut envoyer des *ConfirmActivePDUs* (voir ci-dessous) à toutes les ASCE dans la conférence. On ne demande pas à une ASCE inactive de répondre à un *DemandActivePDU* – Elle

peut choisir de rester inactive. Si une ASCE inactive répond à un *DemandActivePDU*, alors elle rentre dans l'état actif.

Si une ASCE est déjà active, alors sur réception d'un *RequestActivePDU* et conditionnée à une négociation réussie de capacité (voir 8.2), elle enverra des *ConfirmActivePDUs* (voir ci-dessous) à toutes les ASCE dans la conférence.

Dans les deux cas, le *ConfirmActivePDU* sera envoyé à toutes les ASCE dans la conférence sur tous les canaux de priorités – Haute, moyenne et basse – de la manière indiquée dans le Tableau 6-3. Le contenu de *ConfirmActivePDU* est montré dans le Tableau 8-28. Ceci assure que n'importe quelle donnée sur n'importe quel canal peut être reçue après le *ConfirmActivePDU*. Le premier *ConfirmActivePDU* reçu sur n'importe quel canal provoque l'entrée dans l'état actif de l'ASCE réceptrice inactive en instance. Les *ConfirmActivePDUs* ultérieurs seront ignorés.

La Figure 3 illustre l'utilisation des *RequestActivePDU*, *DemandActivePDU* et *ConfirmActivePDU* dans l'activation de trois ASCE, où initialement aucune d'entre elles n'était active et où l'ASCE 1 utilise un *RequestActivePDU* pour déterminer l'état d'activation des autres ASCE à l'intérieur de la conférence. ASCE 1 commence ensuite la partage et utilise un *DemandActivePDU* pour initialiser l'activation des autres ASCE à l'intérieur de la conférence.

Voir la Figure 5 pour un résumé du diagramme de phase des transitions autorisées entre les états d'ASCE.

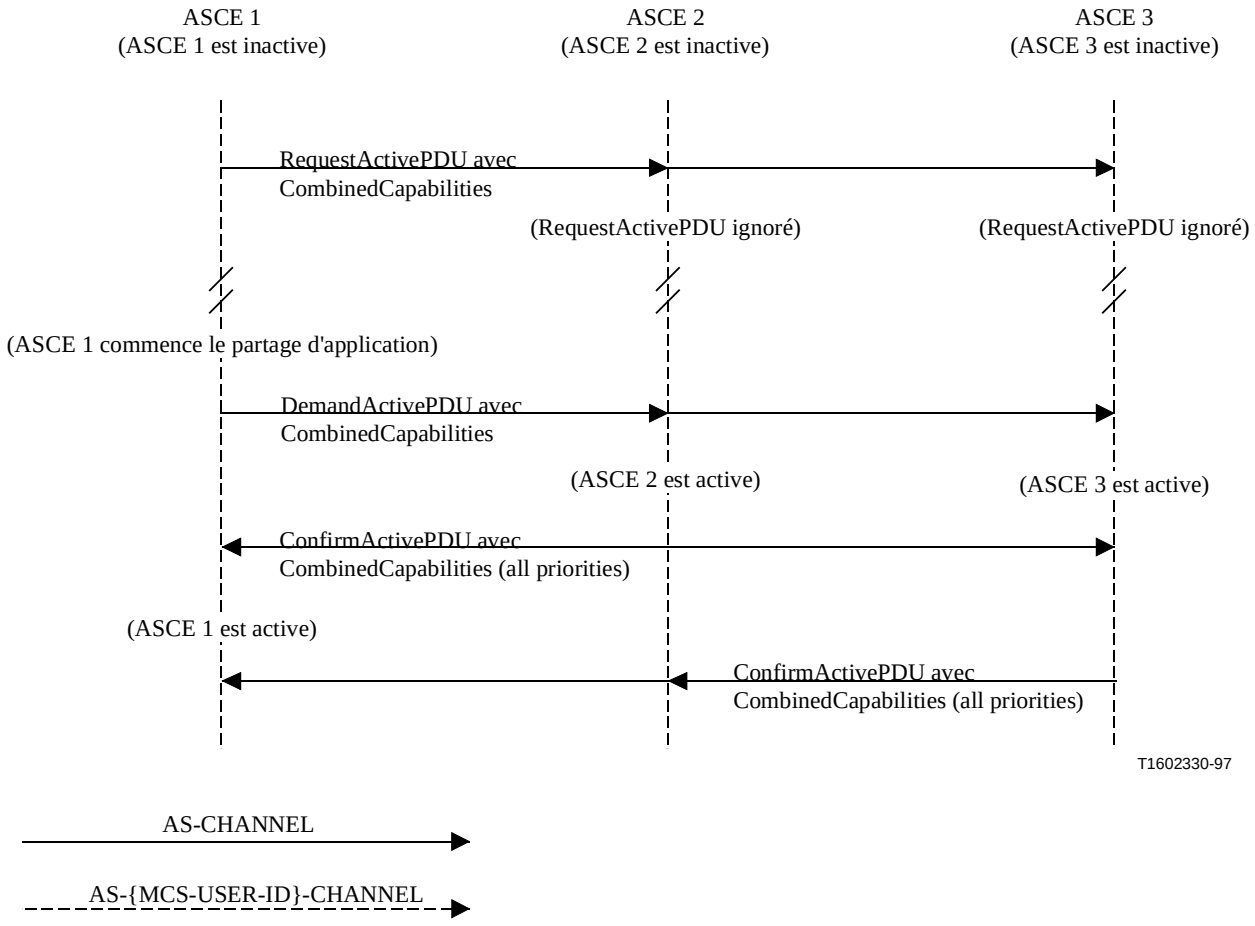


Figure 3/T.128 – Activation réussie de trois ASCE (mode hérité)

Tableau 8-26/T.128 – DemandActivePDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre est l'identificateur de partage proposé à l'intérieur de la session AS. Voir 8.4.2 pour la description de ce paramètre.
sourceDescriptor	ce paramètre est une chaîne de caractères T.50 terminée par NULL identifiant l'ASCE adaptée à l'affichage vers un utilisateur final. La présente Recommandation n'impose aucune interprétation sur les contenus de chaînes.
combinedCapabilities	ce paramètre est une liste des capacités combinées de cette ASCE, contenant une copie de chacun des jeux de capacités du mode hérité dans n'importe quel ordre. Voir 8.2 pour plus d'information sur les capacités.

Tableau 8-27/T.128 – RequestActivePDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
sourceDescriptor	ce paramètre est une chaîne de caractères T.50 terminée par NULL identifiant l'ASCE adaptée à l'affichage vers un utilisateur final. La présente Recommandation n'impose aucune interprétation sur les contenus de chaînes.
combinedCapabilities	ce paramètre est une liste des capacités combinées de cette ASCE, contenant une copie de chacun des jeux de capacités du mode hérité dans n'importe quel ordre. Voir 8.2 pour plus d'information sur les capacités.

Tableau 8-28/T.128 – ConfirmActivePDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre est l'identificateur de partage proposé à l'intérieur de la session AS. Voir 8.4.2 pour la description de ce paramètre.
originatorID	ce paramètre est le *MCS User ID* de l'ASCE ayant distribué le *demandActivePDU* ou le *RequestActivePDU* pour laquelle ceci est une réponse. Ceci permet à une ASCE de recevoir ce *ConfirmActivePDU* pour le corréler avec un *DemandActivePDU* ou *RequestActivePDU* précédents.
sourceDescriptor	ce paramètre est une chaîne de caractères T.50 terminée par NULL identifiant l'ASCE adapté à l'affichage vers un utilisateur final. La présente Recommandation n'impose aucune interprétation sur les contenus de chaînes.
combinedCapabilities	ce paramètre est une liste des capacités combinées de cette ASCE, contenant une copie de chacun des jeux de capacités du mode hérité dans n'importe quel ordre. Voir 8.2 pour plus d'information sur les capacités.

Pour désactiver une autre ASCE, une ASCE enverra un *DeactivateOtherPDU* à l'ASCE devant être désactivée de la manière décrite dans le Tableau 6-3. Le contenu de *DeactivateOtherPDU* est montré dans le Tableau 8-29. Sur réception d'un *DeactivateOtherPDU*, une ASCE deviendra inactive. Il est recommandé qu'une ASCE ne puisse publier qu'un *DeactivateOtherPDU* lorsqu'elle a reçu un

DemandActivePDU, un *RequestActivePDU* ou un *ConfirmActivePDU* avec des capacités qui pourraient affecter sérieusement le déroulement du partage d'application dans la conférence. Cette situation ne devrait pas apparaître lorsque les autres ASCE fournissent des capacités conformes au mode hérité du protocole AS.

Une ASCE peut désactiver toutes les ASCE actives en envoyant un *DeactivateAllPDU* à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *DeactivateAllPDU* est montré dans le Tableau 8-31. Sur réception d'un *DeactivateAllPDU*, une ASCE devient inactive. Il est recommandé que les ASCE ne distribuent des *DeactivateAllPDU* que dans des circonstances exceptionnelles.

Lorsqu'une ASCE détermine qu'il n'y a pas d'autres ASCE actives dans la session AS, elle devient inactive.

Lorsqu'une ASCE devient inactive, elle envoie un *DeactivateSelfPDU* à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *DeactivateSelfPDU* est montré dans le Tableau 8-30.

La Figure 4 illustre l'utilisation de *DeactivateOtherPDU* et *DeactivateSelfPDU* là où l'ASCE 1 désactive l'ASCE 3. Sur réception d'un *DeactivateOtherPDU*, l'ASCE 3 envoie un *DeactivateSelfPDU* aux ASCE 1 et 2, par lequel toutes les deux reçoivent la notification que l'ASCE 3 est devenu inactive.

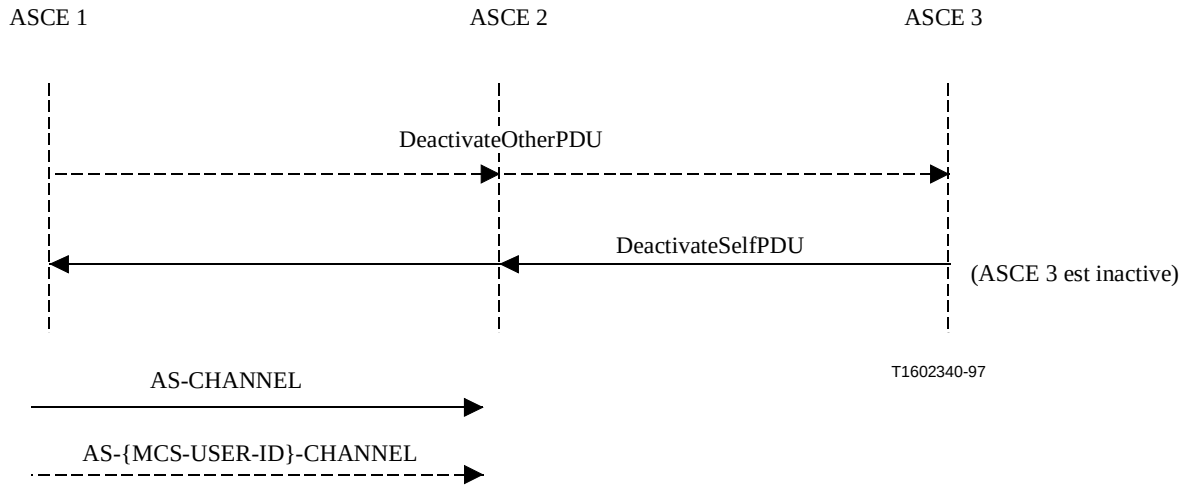


Figure 4/T.128 – Désactivation d'une autre ASCE

Tableau 8-29/T.128 – DeactivateOtherPDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre est l'identificateur de partage courant utilisé dans la session AS. Voir 8.4.2 pour la description de ce paramètre.
deactivateID	ce paramètre est le *MCS User ID* de l'ASCE à désactiver.
sourceDescriptor	ce paramètre est une chaîne de texte T.50 terminée par NULL identifiant l'ASCE adaptée à l'affichage vers un utilisateur final. La présente Recommandation n'impose aucune interprétation des contenus de la chaîne.

Tableau 8-30/T.128 – DeactivateSelfPDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre est l'identificateur de partage courant utilisé dans la session AS. Voir 8.4.2 pour la description de ce paramètre.

Tableau 8-31/T.128 – DeactivateAllPDU

Paramètre	Description
ShareControl Header	le *ShareControl Header* est décrit au 8.3.
shareID	ce paramètre est l'identificateur de partage courant utilisé dans la session AS. Voir 8.4.2 pour la description de ce paramètre.
sourceDescriptor	ce paramètre est une chaîne de texte T.50 terminée par NULL identifiant l'ASCE adaptée à l'affichage vers un utilisateur final. La présente Recommandation n'impose aucune interprétation des contenus de la chaîne.

8.4.2 Identificateurs de partage (mode hérité)

Un identificateur de partage est une poignée 32-bits unique présent dans (presque) toutes les ASPDU que les ASCE utilisent pour détecter et ignorer des données générées tardivement ayant un rapport avec les capacités et les activations précédentes dans la session AS. Dans le mode hérité du protocole AS, les identificateurs de partage sont construits localement par chaque ASCE et comprennent le *MCS User ID* de l'ASCE (dans les 16 bits les plus significatifs) avec un compteur 2^{16} cyclique non signé s'incrémentant de manière monotone (dans les 16 bits les moins significatifs).

Dans le mode hérité du protocole AS, les ASPDU suivantes ne contiennent pas d'identificateur de partage. Toutes les autres ASPDU du mode hérité en contiennent.

- *RequestActivePDU*: voir 8.4.1.
- *FlowTestPDU*: voir 8.5.
- *FlowResponsePDU*: voir 8.5.
- *FlowStopPDU*: voir 8.5.

Chaque ASCE dans une session AS maintiendra un identificateur de partage local, étant sa propre vue de l'identificateur de partage courant dans la session, comme suit:

- quand une ASCE sera inactive ou aura distribué un *RequestActivePDU*, elle mettra dans son identificateur de partage local la valeur spéciale *invalid*;
- une ASCE générera une proposition de nouvel identificateur de partage (en incrémentant la partie compteur de l'identificateur de partage précédent) lors de la distribution d'un *DemandActivePDU* et mettra ce nouvel identificateur dans son identificateur de partage local;
- une ASCE vérifiera l'identificateur de partage dans des *DemandActivePDU* ou *ConfirmActivePDU* entrants par rapport à son identificateur de partage local comme suit:
  - si son identificateur de partage local est *invalid*, elle mettra son identificateur de partage local dans l'identificateur de partage de l'ASPDU entrante;

– si son identificateur de partage local est *valid*, elle mettra son identificateur de partage local à la valeur la plus élevée des deux identificateurs de partage. Il faut noter que, puisque les identificateurs de partage sont constitués d'une combinaison de *MCS User ID* et de compteur local, les valeurs d'identificateur de partage ne peuvent pas être égales.

- sur réception d'un *RequestActivePDU* (si elle est active), une ASCE affectera la valeur d'identificateur de partage à l'identificateur de partage local dans son *ConfirmActivePDU* ultérieur;
- sur distribution de *DeactivateSelfPDU* ou de *DeactivateAllPDU*, ou sur réception de *DeactivateOtherPDU* ou de *DeactivateAllPDU*, ou lorsqu'il n'y a pas d'autre ASCE active dans la session, une ASCE mettra dans son identificateur de partage local la valeur spéciale *invalid*;
- sur réception de toutes les autres ASPDU contenant un identificateur de partage (voir ci-dessus), une ASCE écartera l'ASPDU dont la valeur d'identificateur de partage dans l'ASPDU ne correspond pas à son identificateur de partage local.

Le bénéfice de ce qui est ci-dessus est que les ASCE, utilisant le mode hérité du protocole AS dans une session AS, coopèrent pour maintenir un identificateur de partage courant accepté par toutes les ASCE actives, avec cet identificateur de partage pouvant changer lorsque de nouvelles ASCE deviennent actives.

La Figure 5 présente un diagramme de phase résumant les transitions autorisées entre les états d'ASCE pour ce qui concerne la valeur d'identificateur de partage. Ceci montre deux sous états de l'état actif en instance, dépendant du fait que l'ASCE possède ou non un identificateur de partage *invalid* ou *proposed*. Ceci ne montre pas des ASPDU et des combinaisons d'identificateurs de partage ou d'état qui ne provoquent pas de transitions d'états et ne montre pas non plus d'actions d'ASCE associées à des transitions d'état.

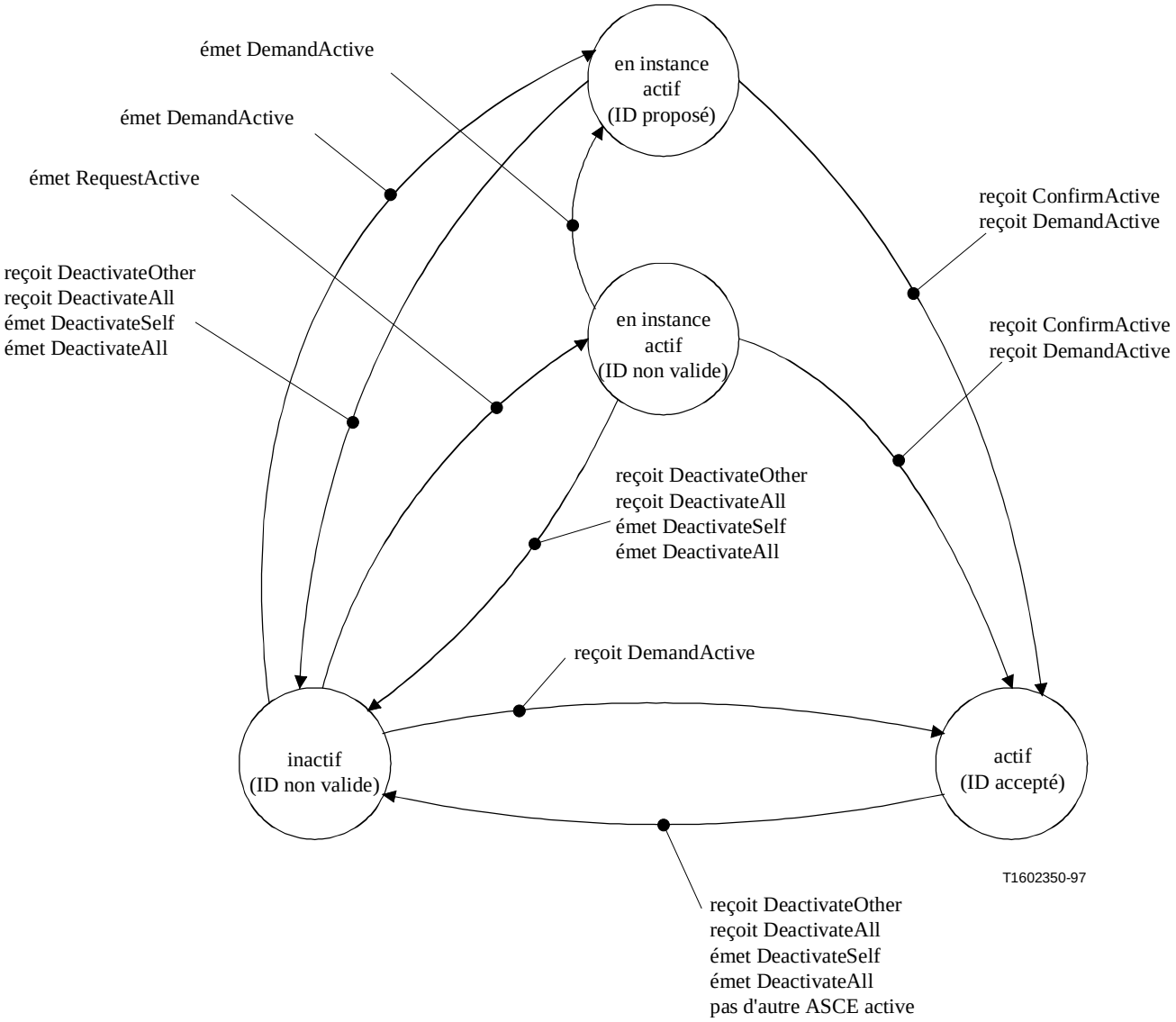


Figure 5/T.128 – Transitions d'état d'ASCE et gestion d'identificateur de partage (mode hérité)

8.4.3 Activation d'ASCE et identificateurs de partage (mode de base)

Dans le mode de base du protocole AS, une activation d'ASCE sera effectuée selon la norme T.121.

Un identificateur de partage est une poignée 32-bits unique présent dans (presque) toutes les ASPDU que les ASCE utilisent pour détecter et ignorer des données générées tardivement ayant un rapport avec les capacités précédentes dans la session AS. Dans le mode de base du protocole AS, une ASCE utilisera le dernier numéro d'instance de répertoire fourni par une indication *GCC-Application-Roster-Report* dans la *Standard Base Session* contenant l'ASCE elle-même et les autres ASCE inscrites ainsi que son identificateur de partage courant.

Dans le mode de base du protocole AS, les ASPDU suivantes ne contiennent pas d'identificateur de partage. Toutes les autres ASPDU du mode de base en contiennent.

- FlowTestPDU: voir 8.5.
- FlowResponsePDU: voir 8.5.
- FlowStopPDU: voir 8.5.

Tandis que l'utilisation d'identificateurs de partage permet à une ASCE d'éviter l'utilisation de données d'AS tardives, il est possible pour chaque ASCE de recevoir des données d'AS en anticipation ayant été générées en rapport avec des capacités n'ayant pas encore été vues par l'ASCE réceptrice. Ceci apparaît parce que le GCC distribue des informations de répertoire via le fournisseur du GCC tandis que les données d'AS sont envoyées en utilisant les primitives *MCS-Send-Data* (non uniformes). Ceci signifie, pour un simple exemple de conférence avec deux ASCE (ASCE A et B) où chacune des ASCE s'inscrit en tant qu'active en même temps, que:

- l'ASCE A peut recevoir un compte rendu de répertoire contenant les capacités des ASCE A et B au temps t_1;
- sur la base de ces capacités, l'ASCE A commence à envoyer des données AS à l'ASCE B au temps t_2;
- les données AS issues de l'ASCE A peuvent arriver à l'ASCE B au temps t_3; à ce moment, l'ASCE B n'a pas encore reçu de compte rendu de répertoire contenant A;
- l'ASCE B peut recevoir un compte rendu de répertoire contenant les capacités pour l'ASCE A au t_4.

Une ASCE pourra détecter des données d'AS en anticipation par le fait que l'identificateur de partage de données sera plus grand modulo 2^{16} que le dernier numéro d'instance de répertoire fourni par l'indication *GCC-Application-Roster-Report* dans la *Standard Base Session*. Dans ce cas, une ASCE traitera de telles données en avance ou les bufferisera jusqu'à réception d'une indication *GCC-Application-Roster-Report* identifiant un numéro d'instance de répertoire supérieur ou égal modulo 2^{16} à celui spécifié dans les données. Si une ASCE choisit de traiter de telles données préalablement à la réception de l'indication de *GCC-Application-Roster-Report,* elle ne supposera pas que les capacités correspondant aux données sont dans les limites définies par ses propres capacités, étant donné que le changement de répertoire amenant le nouveau répertoire d'application peut avoir été déclenché par l'ASCE ayant été retirée de l'application à son initiative ou de manière forcée.

8.5 Contrôle de flux

Un facteur clé de performance pour le partage d'application est la possibilité de découper le contenu de protocole sur la base du trafic potentiel présenté par l'activité de la fenêtre hébergée locale et de la largeur de bande agrégée instantanée des autres nœuds pour assurer un débit et un taux de réponse optimaux. Pendant le déroulement d'une session AS, la largeur de bande agrégée disponible pour le partage d'application variera d'une manière dépendant de nombreux facteurs, incluant:

- la topologie de la conférence et les vitesses des liaisons;
- l'état et le comportement d'activation des autres ASCE;
- la présence et/ou le comportement des autres APE – telles que les APE T.126 et T.127;
- d'autres services (comme la vidéo et/ou l'audio) partant ou arrivant sur des nœuds de la conférence;
- la charge de trafic générale sur les réseaux (dédiés, partagés).

Lorsque la largeur de bande agrégée instantanée disponible pour le partage d'application est réduite, une ASCE a besoin de rajuster le volume de données en transit dans le réseau (à savoir en réduisant les mises à jour d'écrans se recouvrant, en augmentant les taux de compression, etc.) de sorte que, en réduisant la fréquence des mises à jour des écrans distants, le taux de réponse global est maintenu – et en particulier, les utilisateurs présents sur des liaisons plus lentes ne sont pas repoussés en arrière de la conférence. Lorsqu'une largeur de bande agrégée plus instantanée est disponible pour le partage d'application, une ASCE peut avoir la possibilité d'augmenter le volume de données en transit dans

le réseau, de sorte que la fréquence des mises à jour d'écrans distants et le taux de réponse réelle en soient améliorés.

Pour qu'une ASCE ajuste effectivement son protocole de cette manière, ceci requiert un contrôle de flux de bout en bout agrégé et multipoint. Malheureusement, le MCS ne définit par un mécanisme de contrôle de flux approprié. La présente Recommandation définit donc un mécanisme spécifique au protocole AS – contrôle de flux AS – grâce auquel les ASCE peuvent obtenir une information provenant de la largeur de bande agrégée instantanée disponible pour le partage d'application, leur permettant de découper le flux de protocole AS en conséquence.

Le contrôle de flux AS est appliqué sur une base de priorité et de canal. On fera référence en tant que flux à une combinaison particulière canal/priorité. Le contrôle de flux peut être appliqué à zéro, plusieurs ou tous les flux. La présente Recommandation applique le flux de contrôle seulement aux flux AS-CHANNEL et n'applique pas le contrôle de flux aux flux AS-{MCS-USER-ID}-CHANNEL – à savoir, le contrôle de flux est seulement appliqué aux combinaisons canal/priorité diffusées. Ceci signifie qu'une ASCE peut supporter jusqu'à trois flux contrôlés par session.

Pour chaque flux contrôlé, une ASCE spécifie un temps de trajet aller-retour cible et un nombre maximal cible de données en route. Lorsque ces cibles sont activées, une ASCE supervise le temps agrégé aller-retour pour ce flux, via les échanges de *FlowTestPDUs* et de *FlowResponsePDUs* correspondants (voir ci-dessous) et supervise le nombre de données en route en recherchant à le maximiser (jusqu'à la cible maximale) et en préservant le temps de trajet aller-retour cible.

Lorsque le contrôle de flux sera appliqué à un flux particulier, une ASCE enverra périodiquement un *FlowTestPDU* pour ce flux aux autres flux dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *FlowTestPDU* est montré dans le Tableau 8-32. Sur réception d'un *FlowTestPDU*, une ASCE répondra avec un *FlowResponsePDU* à l'ASCE distributrice de la manière indiquée dans le Tableau 6-3. Le contenu de *FlowResponsePDU* est montré dans le Tableau 8-33. La réception d'un *FlowResponsePDU* fournit une estimation sur le délai du trajet aller-retour par rapport à l'ASCE qui a répondu. Il faut noter que cette définition du temps de transfert aller-retour inclut tout le temps passé dans les files d'attente des nœuds finaux, intermédiaires et locaux et est donc, une estimation du temps de livraison de bout en bout à l'ASCE distante.

On demande à une ASCE de distribuer périodiquement des *FlowTestPDUs* et de répondre aux *FlowTestPDUs* par des *FlowResponsePDUs* – sauf là où elle a précédemment envoyé un *FlowStopPDU* (voir ci-dessous).

Une ASCE devrait utiliser les temps de trajet aller-retour fournis par les séries d'échanges de *FlowTestPDUs* et de *FlowResponsePDUs* pour:

- maintenir une estimation dynamique du débit agrégé pour chaque flux contrôlé;
- ajuster l'émission de *FlowTestPDUs* et de *FlowResponsePDUs* pour refléter l'estimation du débit agrégé;
- ajuster le contenu du flux sortant de données de protocole AS pour refléter l'estimation du débit agrégé (voir 8.5.2).

L'implémentation d'un contrôle de flux ASCE prédictif et efficace est un déterminant majeur dans la performance et la facilité d'utilisation du partage d'application. La présente Recommandation requiert donc pour les ASCE l'implémentation d'un algorithme de contrôle de flux qui soit équivalent ou supérieur à l'algorithme décrit au 8.5.1 ci-dessous.

8.5.1 Algorithme de contrôle de Flux

L'algorithme de contrôle de flux est décrit en termes de constantes et variables de contrôle de flux, d'un jeu d'opérations spécifiques sur ces variables et d'une logique décrivant sous quelles conditions

ces opérations sont effectuées. L'Appendice I propose des valeurs de constantes et des valeurs initiales, minimales et maximales pour les variables de contrôle de flux et des suggestions d'expressions pour les opérations, sur la base d'expériences réalisées à partir d'implémentations ASCE particulières. Cependant, des implémentations d'ASCE différentes peuvent trouver que des valeurs et/ou des expressions différentes délivrent un comportement requis équivalent ou supérieur de contrôle de flux.

- Une ASCE utilise les valeurs de constantes de contrôle de flux suivantes pour chaque contrôle de flux:
  - *target_round_trip* ceci est le temps de transfert aller-retour cible pour le flux;
  - *target_in_flight* ceci est le nombre d'octets cible maximal pouvant être simultanément en route pour le flux;
  - *max_queued_recv* ceci est le nombre maximal d'ASPDU reçues pouvant être mises localement dans la file d'attente pour le flux avant que leurs réponses *FlowResponsePDUs* ne soient envoyées.

  L'Appendice I suggère des valeurs pour ces constantes de contrôle de flux sur la base d'expériences réalisées à partir d'implémentations particulières.
- Une ASCE utilise les valeurs de variables de contrôle de flux suivantes pour chaque contrôle de flux:
  - *max_in_flight* ceci est le nombre d'octets maximal pouvant être simultanément en route pour le flux;
  - *flow_period* ceci est le temps minimal courant entre l'envoi de *FlowTestPDUs* successifs.

  Ces variables sont ajustées dynamiquement (voir ci-dessous) sur la base des données envoyées par l'ASCE et sur la base du délai de trajet aller-retour observés avec les autres ASCE.

  L'Appendice I suggère des valeurs initiales, minimales et maximales pour ces variables de contrôle de flux sur la base d'expériences réalisées à partir d'implémentations particulières.
- Une ASCE peut effectuer les opérations suivantes sur les variables de contrôle de flux:
  - diminuer max_in_flight
  - augmenter max_in_flight
  - diminuer flow_period
  - augmenter flow_period

  L'Appendice I suggère des expressions pour ces opérations sur la base d'expériences réalisées à partir d'implémentations ASCE particulières.
- Lors de l'émission de données en mode contrôle de flux, une ASCE devrait vérifier si ces données ont pour effet de faire passer le nombre d'octets en route à une valeur supérieure à *max_in_flight*.
  - si tel n'était pas le cas, l'ASCE devrait envoyer les données;
  - si tel était le cas l'ASCE ne devrait pas envoyer les données – à savoir, l'algorithme de contrôle de flux applique une compression locale en retour dans l'ASCE.

  La manière d'appliquer la contrepression est traitée localement sur l'initiative de l'ASCE.
- Sur réception d'un *FlowTestPDU*, une ASCE devrait vérifier si elle est en train de traiter plus de *max_queued_recv* ASPDU présentes dans le flux.

– si tel n'était pas le cas, elle devrait répondre avec un *FlowResponsePDU* référençant ce flux et utilisant le *flowNumber* fourni dans le *FlowTestPDU*;

– si tel était le cas, elle devrait mettre en instance le *FlowResponsePDU* jusqu'à ce que le reliquat local ait été réduit – à savoir le nombre d'ASPDU localement mis en file d'attente dans le flux soit inférieur ou égal à *max_queued_recv*.

La vérification de l'existence d'un reliquat de données précédemment reçues évite les problèmes liés à une ASCE réceptrice étant l'élément le plus lent dans le réseau et à un *FlowResponsePDU* inconditionnel ayant pour résultat de produire une accumulation de données non traitées sur le récepteur.

• Sur réception d'un *FlowResponsePDU*, une ASCE devrait vérifier si l'algorithme de contrôle de flux applique une compression locale en retour.

– si tel n'était pas le cas, aucune action ne sera nécessaire;

– si tel était le cas et que le délai de transfert aller/retour soit plus mauvais que *target_round_trip*, elle devrait diminuer *max_in_flight* et augmenter *flow_period*. Ceci aura pour effet de réduire le débit en encourageant une contrepression plus précoce;

– si tel était le cas et que le délai de transfert aller/retour soit meilleur que *target_round_trip*, elle devrait augmenter *max_in_flight*. Ceci aura pour effet d'élever la contrepression et d'augmenter le débit lorsque les données seront ensuite envoyées (en supposant qu'un délai de trajet aller/retour ne se produise pas dans l'intervalle). Si le nouveau *max_in_flight* a atteint *target_in_flight*, alors l'ASCE devrait aussi diminuer *flow_period*. Ceci aura pour effet d'augmenter la sensibilité de l'algorithme à toute détérioration des temps de trajet aller/retour comme effet de l'accroissement des données en route.

Ceci signifie que l'ASCE ajustera seulement *max_in_flight* ou *flow_period* là où une compression locale en retour est appliquée. Ceci se produit parce que l'algorithme utilise le trajet aller/retour observé comme une estimation du débit, ce qui est une corrélation tout juste raisonnable là où le trajet aller/retour coïncide avec une période occupée de trafic de données – à savoir, lorsque cette ASCE applique une compression locale en retour parce qu'il y a au moins *max_in_flight* octets en route.

Il faut noter qu'une ASCE devrait faire attention à ce que les estimations successives de trajet aller/retour pendant une compression locale en retour n'aboutissent à un effet d'oscillation – par exemple, là où les trajets aller/retour successifs à l'intérieur d'une même *flow_period* sont plus mauvais que la cible (ce qui aboutit à diminuer *max_in_flight* et augmenter *flow_period*) et ensuite meilleurs que la cible (ce qui aboutit à augmenter *max_in_flight* et diminuer *flow_period*). Une approche heuristique recommandée est de biaiser la décision de sorte qu'une réduction de débit soit effectuée dans les cas où le trajet aller/retour est simplement plus mauvais que *target_round_trip*, tandis qu'une augmentation de débit est seulement effectuée lorsque le trajet aller/retour est inférieur à la moitié de *target_round_trip*.

L'ASCE devrait appliquer cette logique pour chaque *FlowResponsePDU* reçu à l'intérieur d'une *flow_period* – avec ceux qui répondent tôt positionnant les valeurs initiales de la période et ceux qui répondent tard ajustant ces valeurs pour refléter les caractéristiques de largeur de bande applicables aux ASCE correspondantes.

La période *flow_period* est le temps minimal courant entre des *FlowTestPDUs*. Par exemple, si la *flow_period* est de cinq millisecondes, alors une ASCE enverra *FlowTestPDU* au début de *flow_period* et recevra des *FlowResponsePDUs* de toutes les ASCE qui répondent (voir ci-dessous) en moins de quatre millisecondes. Ensuite, elle ne devrait pas envoyer le *FlowTestPDU* suivant avant une milliseconde supplémentaire. A l'opposé, si dans les cinq millisecondes elle n'a pas reçu

de *FlowResponsePDUs* de toutes les ASCE qui répondent, elle ne devrait pas envoyer le prochain *FlowTestPDU* tant qu'elle n'a pas reçu les *FlowResponsePDUs* de toutes les ASCE qui répondent et que la limite de temps supérieure n'a pas été atteinte. Cette approche signifie que, pour un *flow_period* particulier, l'algorithme répond de manière incrémentale au trajet aller/retour observé pour toutes les ASCE qui répondent – en supposant qu'elles répondent dans un intervalle de temps raisonnable.

Le groupe d'ASCE qui répondent en provenance desquelles des réponses sont attendues à l'intérieur de l'intervalle *flow_period* peut ne pas nécessairement correspondre au groupe formé par toutes les autres ASCE actives. Une ASCE émettrice ne devrait pas inclure une ASCE à l'intérieur de son groupe d'ASCE qui répondent lorsque cette ASCE a fourni une réponse initiale et devrait exclure les ASCE qui ne répondent pas dans la limite de temps supérieure (au moins, jusqu'à ce que de telles ASCE répondent à nouveau dans les *flow-periods* suivantes). Cette approche assure que les ASCE fautives ne rendent pas l'algorithme dissymétrique.

Le bénéfice résultant de ce qui est énoncé précédemment est que les ASCE émettrices (pour un flux particulier):

- appliquent la contrepression locale lorsque les données en route atteignent le nombre autorisé;
- accroissent ou décroissent les données en route autorisées pour refléter la largeur de bande agrégée instantanée, sur la base du délai de trajet aller/retour observé;
- accroissent ou décroissent *flow_period* (et ainsi la proportion de largeur de bande occupée par le flux d'ASPDU) pour refléter la largeur de bande agrégée instantanée.

Tableau 8-32/T.128 – FlowTestPDU

Paramètre	Description
flowID	ce paramètre identifie le flux contrôlé pour l'ASCE émettrice. Les valeurs autorisées sont dans l'intervalle 0..127. Ce paramètre est alloué par l'ASCE émettrice et n'a pas de corrélation spéciale avec des priorités et/ou des canaux particuliers.
flowNumber	ce paramètre identifie le *FlowTestPDU* particulier sur le *flowID* ci-dessus. Les valeurs autorisées sont dans l'intervalle 0..255. Une ASCE devra incrémenter ce paramètre pour chaque *FlowTestPDU* successif, en effectuant cette opération modulo 255.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Tableau 8-33/T.128 – FlowResponsePDU

Paramètre	Description
flowID	ce paramètre identifie le flux contrôlé pour l'ASCE émettrice. Les valeurs autorisées sont dans l'intervalle 0..127. L'ASCE émettrice mettra ce paramètre à la valeur de celui utilisé dans le *FlowTestPDU* correspondant.
flowNumber	ce paramètre identifie le *FlowResponsePDU* particulier sur le *flowID* ci-dessus. Les valeurs autorisées sont dans l'intervalle 0..255. L'ASCE émettrice mettra ce paramètre à la valeur du paramètre correspondant de *FlowTestPDU*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Une ASCE peut se retirer elle-même des autres groupes d'ASCE répondant au contrôle de flux en envoyant un *FlowStopPDU* pour ce flux à toutes les autres ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *FlowStopPDU* est montré dans le Tableau 8-34. Ce mécanisme est de toute première importance dans le mode hérité du protocole AS, puisqu'il permet à une ASCE de rester inscrite dans le AS-CHANNEL (et donc de recevoir encore les *FlowTestPDUs*) mais d'être inactive (au sens de l'activation d'ASCE – voir 8.4) et assure ainsi que son absence de réponse n'apportera pas de dissymétrie aux calculs de contrôle de flux des autres ASCE actives.

Lorsque l'ASCE souhaite à nouveau être incluse dans les calculs de contrôle de flux, elle devra répondre au *FlowTestPDU* suivant avec un *FlowResponsePDU*, qui aura pour effet de la rajouter à nouveau dans les autres groupes d'ASCE répondant au contrôle de flux pour ce flux donné.

Tableau 8-34/T.128 – FlowStopPDU

Paramètre	Description
flowID	ce paramètre identifie le flux contrôlé pour l'ASCE émettrice. Les valeurs autorisées sont dans l'intervalle 0..127. L'ASCE émettrice mettra ce paramètre à la valeur de celui utilisé dans le *FlowTestPDU* correspondant.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.5.2 Réponse à la contrepression

Lorsque la contrepression est en cours d'application dans une ASCE comme résultat de l'algorithme de contrôle de flux décrit au 8.5.1, une ASCE peut traiter les données d'AS en instance pour arriver à une réduction du volume de données.

Un exemple de stratégies pour ce qui concerne les données d'AS peut être décrit comme suit:

- commutation entre émission d'ordres et données phototrame:

  lorsqu'une application locale est en train de tracer de manière très active dans une fenêtre hébergée et que l'ASCE subit une contrepression du contrôle de flux, l'ASCE peut préférer accumuler de l'information de limites pour l'activité de tracé plutôt que d'envoyer les ordres, puis envoyer les données de phototrame cumulées. Ceci peut réduire la fréquence des mises à jour sur les ASCE distantes, mais minimisera les données en route et assurera que les ASCE distantes suivent l'activité de tracé. Les points de commutation entre les ordres et les

données de phototrame et inversement dépendront du comportement de tracé de l'application, du contrôle de flux et de l'implémentation particulière de l'ASCE;

- réduction des mises à jour recouvrantes – *spoiling*:

  là où une ASCE effectue une contrepression du contrôle de flux et que l'application locale trace une séquence d'ordres ou de mises à jour de données de phototrame recouvrantes ou occultantes, alors la séquence peut (dans certains cas) être réduite à une séquence plus petite et/ou à une simple mise à jour. A nouveau, ceci peut réduire la fréquence des mises à jour sur les ASCE distantes, mais minimisera les données en route et assurera que les ASCE distantes suivent l'activité de tracé.

Là où une ASCE effectue une contrepression du contrôle de flux et qu'il y a un niveau élevé d'activité de dispositif de pointage d'utilisateur, l'ASCE peut réduire une séquence d'événements de déplacement de dispositif de pointage à un simple mouvement – ce procédé s'appelle le *spoiling*. Le spoiling en entrée peut aussi être appliqué à l'ASCE hôte, là où une séquence d'événements de dispositifs de pointage est en file d'attente à l'intérieur de l'ASCE attendant d'être injectée dans l'environnement du terminal local. Le spoiling sur des entrées émises et reçues peut réduire la fréquence des mises à jour de déplacements de dispositifs de pointage mais minimisera les données en route et assurera que les ASCE distantes suivent l'activité du dispositif de pointage.

La présente Recommandation ne spécifie pas comment une ASCE doit ajuster le contenu des flux de données du protocole AS en entrée ou en sortie lorsque a lieu la contrepression. Cependant la présente Recommandation requiert vraiment de l'ASCE qu'elle s'assure que le flux de protocole contient suffisamment d'information pour permettre aux ASCE homologues d'afficher correctement les fenêtres reflets et/ou de contrôler correctement les applications hébergées. De la même manière, là où le flux de sortie se fie à l'information de contrôle (telle qu'une liste de fenêtres ou une information de palette) qui a changé sur l'ASCE hôte, il est de la responsabilité de l'ASCE hôte de s'assurer que l'information de contrôle sera envoyée préalablement à celle de sortie. Toutes les ASPDU de contrôle et de sortie dépendantes d'ordres sont envoyées avec une basse priorité, de sorte que l'ASCE émettrice puisse ordonner cette sortie de manière fiable.

8.6 Synchronisation

Lorsqu'une ASCE est active dans une session AS, il y a plusieurs classes d'événements de session pouvant lui demander de mettre à jour des ressources et/ou un état relatifs au protocole, ou d'envoyer des ASPDU permettant aux autres ASCE d'effectuer des changements complémentaires. Le traitement d'ASCE associé à ces événements de session s'appelle la synchronisation.

Le protocole AS définit quatre classes de synchronisation – référencées comme synchronisation d'ASCE, d'hôte, de reflet et d'entrée. Chaque classe de synchronisation définit son propre ensemble de besoins de synchronisation d'ASCE associée à un nombre particulier d'événements de session. Par exemple, la synchronisation hôte définit un ensemble de besoins de synchronisation s'appliquant à une ASCE qui héberge (ou qui commence ou finit d'héberger) des fenêtres.

Le présent sous-paragraphe fournit un résumé des opérations de synchronisation ASCE requises pour supporter le protocole AS. Il ne décrit pas chaque opération en détail, mais référence plutôt les sous-paragraphes de la présente Recommandation décrivant le domaine fonctionnel pertinent.

Là où la description référence une ressource locale particulière, telle qu'une valeur d'identificateur ou une antémémoire d'émission, il le fait en termes logiques, en ne faisant aucune hypothèse sur une implémentation locale particulière – la seule contrainte est que l'ASCE qui synchronise accomplisse (ultérieurement) l'effet de protocole requis. De plus, il n'adresse pas les optimisations potentielles pouvant arriver à la suite de la réduction d'une série d'opérations répétées de synchronisation.

8.6.1 Synchronisation d'ASCE

Lorsqu'une ASCE devient active, elle effectue les opérations de synchronisation suivantes. Voir 8.4 pour plus d'information sur l'activation et la désactivation d'ASCE.

- elle mettra à zéro la partie du numéro de séquence d'identificateur de liste de fenêtre. Voir 8.10 pour plus d'information;
- elle mettra à zéro son identificateur d'activation de fenêtre. Voir 8.11 pour plus d'information;
- elle mettra à zéro son identificateur de contrôle. Voir 8.12 pour plus d'information.

Le protocole AS n'impose aucune synchronisation spécifique lorsqu'une ASCE devient inactive.

Quand une ASCE active détectera qu'une autre ASCE est devenue active, elle effectuera les opérations de synchronisation suivantes:

- elle déterminera les capacités négociées pour toutes les ASCE actives (à savoir existantes et nouvelles) dans la session AS et utilisera ces capacités comme base de construction de toutes les ASPDU de données ultérieures. Voir 8.2 pour plus d'information sur les capacités et la négociation des capacités;
- avant d'envoyer d'autres ASPDU de données, elle enverra un *SynchronizePDU* sur tous les flux/priorités sortants (voir 8.3.1) à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3 et marquera tous les flux/priorités entrants pour que chacune des autres ASCE actives reste en attente de synchronisation jusqu'à ce qu'elle reçoive un *SynchronizePDU* – et jusqu'à cet instant-là ignorera toutes les ASPDU de données entrantes sur le flux/priorité provenant de cette ASCE particulière. Ceci assure que, là où l'identificateur de partage ne change pas à la suite de la mise en activité de la nouvelle ASCE (voir 8.4.2), toute donnée provenant d'autres ASCE relative aux capacités précédentes combinées négociées est ignorée. Un *SynchronizePDU* particulier est dirigé sur un flux/priorité unique à destination d'autres ASCE actives particulières, mais est envoyé sur le AS-CHANNEL (et donc à toutes les ASCE). Chaque *SynchronizePDU* inclut l'identificateur d'utilisateur MCS de l'ASCE destinataire de manière à ce que les autres ASCE ignorent l'information. Ce mécanisme assure que le *SynchronizePDU* arrive avant toute diffusion de donnée vers l'ASCE destination. Le contenu de *SynchronizePDU* est montré dans le Tableau 8-35;
- elle ignorera sa liste de polices mises en correspondance, réinitialisera à zéro son compte de *FontPDUs* reçues et pourra envoyer un *FontPDU* pour annoncer à nouveau son jeu de polices de correspondance. Voir 8.8 pour plus d'information;
- elle enverra soit un *Cooperate* ou *Detach ControlPDU* pour annoncer son état de contrôle. Si elle est coopérante et si elle possède l'identificateur de contrôle, elle enverra aussi un *Grant Control ControlPDU* pour se référencer elle-même auprès de toutes les ASCE actives pour annoncer qu'elle possède l'identificateur de contrôle. Voir 8.12 pour plus d'information;
- si l'ASCE héberge une fenêtre, elle effectuera aussi les opérations de synchronisation d'hôte qui en découlent (voir 8.6.2 ci-dessous);
- elle effectuera les opérations de synchronisation d'entrée qui en découlent (voir 8.6.4 ci-dessous).

La synchronisation d'ASCE découlant du fait que d'autres ASCE sont devenues actives peut générer une charge locale de traitement et un trafic ASPDU significatifs, tout spécialement là où elle initialise aussi une synchronisation d'hôte. Il est recommandé que là où une ASCE détecte qu'un certain nombre d'autres ASCE sont devenues actives dans la conférence de manière rapprochée, elle

implémente la synchronisation d'ASCE de manière à décaler dans le temps plusieurs ou toutes les synchronisations demandées pour chaque nouvelle ASCE active jusqu'à ce qu'une proportion significative des ASCE, voire toutes, soit active, en supposant que le décalage dans le temps soit limité eu égard la proportion d'ASCE du groupe et un nombre maximal arbitraire et aussi que la capacité à répondre à l'activation soit maintenue.

Lorsqu'une autre ASCE devient inactive, on peut demander à une ASCE d'effectuer les opérations de synchronisation suivantes. Voir 8.4 pour plus d'information sur l'activation et la désactivation d'ASCE.

- là où l'autre ASCE devenant inactive a pour effet de rendre l'ASCE locale inactive (parce qu'elle est la seule ASCE active restant dans la session AS), alors aucune synchronisation d'ASCE n'est requise;
- là où l'ASCE devenue inactive détenait l'identificateur de contrôle, alors on demande à chaque ASCE restant active de participer à un échange de *ControlPDUs* pour rétablir une ASCE comme possesseur unique de l'identificateur de contrôle. Voir 8.12.1 pour plus d'information;
- là où d'autres ASCE restent actives, l'ASCE locale peut déterminer à nouveau les capacités négociées (voir 8.2) pour toutes les autres ASCE restant actives dans la session AS et utiliser ces capacités comme base de construction des ASPDU de données ultérieures. Ceci est optionnel, comme les capacités négociées pour les ASCE précédemment actives ont été négociées et incluaient les ASCE restantes, et bien que les capacités renégociées puissent permettre la construction d'un flux de protocole AS plus optimisé, les capacités négociées précédemment restent encore valables. Voir 8.2 pour plus d'information;
- là où d'autres ASCE restent actives, l'ASCE locale peut aussi mettre à nouveau en correspondance sa liste de polices déjà mise, en se basant sur l'information de police reçue des ASCE restant actives. A nouveau, ceci est optionnel et bien que ceci puisse avoir pour résultat l'obtention de polices de correspondance supplémentaires, les polices précédemment mises en correspondance sont encore valables. Voir 8.8 pour plus d'information.

Tableau 8-35/T.128 – SynchronizePDU

Paramètre	Description
ShareData Header	le *ShareData Header* est décrit au 8.3.
targetUser	ce paramètre est le *MCS User ID* de l'ASCE vers laquelle cette ASPDU est dirigée. Là où une ASCE reçoit un *SynchronizePDU* contenant un *MCS User ID* autre que le sien, elle l'ignorera.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.6.2 Synchronisation d'hôte

La synchronisation d'hôte représente l'ensemble des opérations de synchronisation associées à l'hébergement de fenêtre.

Lorsqu'une ASCE démarre en premier l'hébergement de fenêtre, lorsqu'elle héberge des fenêtres et que de nouvelles ASCE deviennent actives à leur tour, ou lorsqu'elle héberge des fenêtres et qu'une capacité est mise à jour (voir 8.2), elle effectuera les opérations de synchronisation suivantes:

- elle enverra un *UpdatePDU* (*Synchronization*) à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le *UpdatePDU* (*Synchronization*) sera envoyé avant toute autre ASPDU générée par l'ASCE émettrice comme résultat d'une synchronisation d'hôte et notifiera aux autres ASCE que cette ASCE héberge des fenêtres. Sur réception d'un *UpdatePDU* (*Synchronize*), une ASCE effectuera toute synchronisation de reflet pertinente (voir 8.6.3 ci-dessous). Le contenu de *UpdatePDU* (*Synchronization*) est montré dans le Tableau 8-36;
- elle réinitialisera son antémémoire de curseur de couleur d'émission. Voir 8.14 pour plus d'information;
- dans le mode de base du protocole AS seulement, elle réinitialisera son espace de couleur d'émission à RVB avec aucune information d'exactitude de couleur;
- elle mettra en instance l'émission d'un *UpdatePDU* (*Palette*) avec une nouvelle palette, qui sera envoyé préalablement à toute donnée de phototrame ultérieure;
- elle réinitialisera son état de codage d'ordre. Voir 8.16.3 pour plus d'information;
- elle réinitialisera son antémémoire de phototrame d'émission. Voir 8.16.7 pour plus d'information;
- elle réinitialisera son antémémoire de tableau de couleur d'émission. Voir 8.16.8 pour plus d'information;
- elle réinitialisera son antémémoire de sauvegarde de moniteur. Voir 8.16.17 pour plus d'information;
- elle enverra un *ApplicationPDU* avec une action *NotifyHostedApplications* indiquant le nombre d'applications hébergées sur ce terminal;
- elle enverra un *WindowListPDU* contenant de l'information sur la structure de fenêtre locale courante qui sera envoyée avant tout *WindowActivation*. Voir 8.10 pour plus d'information;
- elle enverra un *WindowActivationPDU* fournissant de l'information sur l'état d'activation de la fenêtre locale. Voir 8.11 pour plus d'information;
- elle mettra en file d'attente un ordre *Desktop Origin* qui sera envoyé dans une ASPDU *UpdatePDU* (*Orders*) avant toute donnée de phototrame ou ordre ultérieurs. Voir 8.16.18 pour plus d'information;
- elle construira un flux de sortie AS, constitué d'un mélange de données de phototrame et/ou d'ordres, contenant suffisamment d'information pour permettre aux autres ASCE de tracer les fenêtres reflets correspondant à ses fenêtres hébergées.

Lorsqu'une ASCE arrête d'héberger une fenêtre tout en restant active, elle enverra un *ApplicationPDU* avec l'action *NotifyHostedApplications* indiquant qu'aucune application n'est maintenant hébergée pour permettre aux autres ASCE d'effectuer toutes les synchronisations de reflet requises.

Tableau 8-36/T.128 – UpdatePDU (Synchronization)

Paramètre	Description
ShareData Header	le *ShareData Header* est décrit au 8.3
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités correspondantes hors normes sont présentes dans le jeu de capacités négocié.

8.6.3 Synchronisation de reflet

La synchronisation de reflet comprend l'ensemble des opérations de synchronisation associées avec le tracé de fenêtres reflets. Elle est mise en œuvre en relation avec une ou toutes les fenêtres hôtes des ASCE.

Lorsqu'une ASCE reçoit un *UpdatePDU* (*Synchronisation*) d'une autre ASCE (voir 8.6.2), il lui est demandé de traiter la synchronisation de reflet en relation avec cette ASCE hôte. Là où une ASCE détermine qu'un jeu de capacités a été mis à jour (dans le mode hérité, sur réception d'un *UpdateCapabilityPDU*; dans le mode de base sur réception d'un *GCC-Application-Roster-Report* avec des capacités changées), il lui est demandé de traiter la synchronisation de reflet en relation avec toutes les ASCE hôtes. Là où la synchronisation de reflet est requise, une ASCE effectuera les opérations de synchronisation suivantes pour chaque ASCE hôte affectée.

- elle réinitialisera son antémémoire de curseur de couleurs de réception pour cette ASCE hôte. Voir 8.14 pour plus d'information;
- dans le mode de base du protocole AS seulement, elle réinitialisera son espace de couleurs de réception pour cette ASCE hôte à la valeur RVB sans information d'exactitude de couleur;
- elle réinitialisera son état de décodage d'ordre pour cette ASCE hôte. Voir 8.16.3 pour plus d'information;
- elle réinitialisera son antémémoire de phototrame de réception pour cette ASCE hôte. Voir 8.16.7 pour plus d'information;
- elle réinitialisera son antémémoire de table de couleur de réception pour cette ASCE hôte. Voir 8.16.8 pour plus d'information;
- elle réinitialisera son antémémoire de sauvegarde de moniteur de réception pour cette ASCE hôte. Voir 8.16.17 pour plus d'information;
- elle réinitialisera son origine de moniteur à la valeur (0,0) pour cette ASCE hôte. Voir 8.16.18 pour plus d'information.

Le protocole AS ne requiert aucune synchronisation spécifique lorsqu'une ASCE détecte qu'une autre ASCE n'est plus hôte – par exemple, lorsqu'une ASCE hôte devient inactive ou que l'ASCE reçoit un *ApplicationPDU* avec une action *NotifyHostedApplications* indiquant qu'aucune application n'est plus hébergée sur une ASCE hôte précédemment. Cependant, une ASCE peut utiliser la réception d'un *ApplicationPDU* avec une action *NotifyHostedApplications* et aucune application pour libérer les ressources locales (telles que les antémémoires) qui étaient allouées pour cette ASCE.

8.6.4 Synchronisation d'entrée

La synchronisation d'entrée est formée de l'ensemble des opérations de synchronisation associées à la maintenance de l'état du clavier entre les ASCE contrôlantes et contrôlées. Voir 8.18 pour plus d'information.

Lorsqu'une ASCE détectera qu'une nouvelle ASCE est devenue inactive, elle réinitialisera l'état d'émission de son clavier et mettra en file d'attente l'événement *Input Synchronization* pour le prochain *InputPDU*. Sur réception d'un événement de synchronisation d'entrée dans un *InputPDU*, une ASCE réinitialisera son état de réception de clavier pour ce qui concerne l'ASCE distributrice.

8.7 Partage distant

Dans certains scénarios de partage d'application (à savoir le travail à distance et/ou le soutien à l'aide de bureau), il est utile d'initialiser à distance le partage d'applications et/ou de fenêtres – à savoir que

les applications s'exécutant sur le terminal local ne sont pas partagées sur l'initiative d'un utilisateur final local ou d'une action de programmation, mais plutôt comme le résultat d'une requête en provenance d'une ASCE homologue. Ceci s'appelle le partage distant.

Quand une ASCE désirera initialiser le partage distant sur une ASCE homologue, elle enverra un *RemoteSharePDU* avec l'action *Request Remote Share* et un mot de passe crypté à l'ASCE homologue de la manière décrite dans le Tableau 6-3. Le contenu de *RemoteSharePDU* est décrit dans le Tableau 8-37.

Sur réception d'un *Request Remote Share RemoteSharePDU*, une ASCE utilise un mécanisme purement local (tel que l'interaction avec l'utilisateur final local) pour déterminer s'il faut autoriser le partage distant avec l'ASCE qui le demande et/ou valider le mot de passe fourni. Si elle accepte la demande de partage distant, elle répond avec un *Confirm Remote Share RemoteSharePDU* à l'ASCE demandeuse et partage les applications et/ou les fenêtres locales qui sont éligibles pour le partage distant. La liste particulière des applications et/ou fenêtres locales éligibles pour le partage distant est une affaire locale. Par exemple, une ASCE peut offrir des options de configuration où les utilisateurs finaux peuvent restreindre par classe, par nom ou par ASCE demandeuse, la liste des applications et/ou des fenêtres partageables distantes.

Si l'ASCE n'accepte pas la demande de partage distant, elle répond avec un *Deny Remote Share RemoteSharePDU* à l'ASCE demandeuse, indiquant pourquoi la tentative de partage distant a été refusée. Les valeurs de refus sont définies pour les cas suivants:

- l'ASCE demandeuse a fourni un mot de passe non valable;
- l'ASCE réceptrice ne supporte pas le partage distant ou n'a pas validé le partage distant;
- l'ASCE réceptrice a déjà exécuté un partage distant, c'est-à-dire que ses applications et/ou fenêtres partageables à distance sont déjà partagées;
- l'ASCE réceptrice a demandé un partage distant à une troisième ASCE.

Tableau 8-37/T.128 – RemoteSharePDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
action	ce paramètre identifie l'action *RemoteSharePDU* particulière. Les valeurs autorisées sont les suivantes. • *Request Remote Share* • *Confirm Remote Share* • *Deny Remote Share*
additionalData	ce paramètre fournit des données optionnelles supplémentaires pour des actions spécifiques: • pour l'action *Request Remote Share*, ce paramètre sera le *MCS User ID* de l'ASCE émettrice. Ceci permet aux ASCE réceptrices d'implémenter la sécurité du partage d'application ASCE distante par ASCE distante; • pour l'action *Deny Remote Share*, ce paramètre fournit un code donnant la raison du refus comme suit: – *Incorrect Password* – *Remote Share Not Enabled/Supported* – *Remote Share In Operation (Incoming)* – *Remote Share In Operation (Outgoing)*

Tableau 8-37/T.128 – RemoteSharePDU *(fin)*

Paramètre	Description
encryptedPassword	ce paramètre n'est présent que pour un message *Request Remote Share* et il s'agit alors d'un simple mot de passe d'authentification protégé de type X.509 sans horodatage ni nombre aléatoire.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.8 Polices de caractères

Une ASCE envoie une information de police de caractères à toutes les ASCE de la conférence en envoyant un *FontPDU* de la manière décrite dans le Tableau 6-3. Le contenu de *FontPDU* est décrit dans le Tableau 8-38.

L'information d'attribut de police de caractères fournit une information sur les polices de caractères satisfaisant aux exigences du protocole AS pour lequel l'ASCE émettrice est prête à recevoir les ordres *Text* et *Extended Text* (voir 8.16.11 et 8.16.12).

Une ASCE peut envoyer un *FontPDU* pendant la synchronisation d'ASCE pour permettre aux autres ASCE actives d'effectuer la correspondance de police, ce qui est un préalable à l'envoi des ordres *Text* et *Extended Text* dans la session AS. Voir 8.6 pour plus d'information sur la synchronisation. Une ASCE n'a pas besoin d'envoyer des *FontPDUs* là où les polices de caractères locales autorisées (voir 8.8.1) et/ou là où les ordres *Text* et *Extended Text* ne sont pas supportés.

Là où une ASCE n'établit pas de correspondance avec une police particulière et que cette police est ensuite utilisée localement par une application fonctionnant sur un terminal local partagé dans la conférence, l'ASCE ne peut pas envoyer d'ordres *Text* ou *Extended Text* faisant référence à cette police. Cependant, elle peut envoyer une ASPDU *UpdatePDU* (*Bitmap*) contenant les bits en provenance de la zone d'affichage du terminal local correspondants aux limites de texte; ceci devrait donner comme résultat une apparence visuelle de fenêtre reflet équivalente sur les ASCE homologues.

Par exemple, le protocole d'ASCE courant autorise seulement l'annonce de polices où tout ou partie des codes ont des glyphes dans la grille de codage du cœur du protocole AS ou de sa version complète (voir 8.8.1) et où la liste des codes de l'ordre *Text* ou *Extended Text* suppose une grille de codage 8-bits simple. Ceci exclut d'annoncer des polices double-octet ou l'envoi d'ordres *Text* et *Extended Text* pour de telles polices. Cependant là où de telles polices sont utilisées pour tracer du texte à l'intérieur des applications partagées sur une ou plusieurs ASCE dans une conférence, les ASCE peuvent utiliser des ASPDU *UpdatePDU* (*Bitmap*) appropriées pour produire la même apparence visuelle de fenêtre reflet sur les ASCE homologues.

Tableau 8-38/T.128 – FontPDU

Paramètre	Description
shareData Header	le *Share Data Header* est décrit au 8.3.
entrySize	ce paramètre spécifie la taille en octets d'un attribut de police dans le paramètre *fontList* (voir ci-dessous). Ce paramètre est seulement utilisé pour le mode hérité du *FontPDUs*.
fontList	ce paramètre est une liste d'attributs de police.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Une ASCE réceptrice traitera la position des attributs de police dans le *FontPDU* comme un identificateur numérique partant de zéro et référencé comme le *FontID* pour tout usage ultérieur dans une correspondance de police et lors de l'envoi d'ordres *Text* ou *Extended Text*. Par exemple, là où un *FontPDU* contient 6 attributs de police, l'ASCE réceptrice utilisera les *FontIDs* de 0 à 5 pour référencer respectivement les polices correspondantes au niveau de l'ASCE émettrice. Voir le Tableau 8-39.

Tableau 8-39/T.128 – Attributs de Police (Font)

Paramètre	Description
faceName	ce paramètre est une chaîne de caractère terminée par NULL spécifiant le nom de la police.
fontFlags	ce paramètre est un ensemble d'indicateurs binaires indiquant les caractéristiques de la police. Les valeurs définies d'indicateur binaire sont les suivantes. • *Fixed Pitch* si cet indicateur est positionné, il indique que la police est à chasse fixe, sinon elle est proportionnelle. • *Fixed Size* si cet indicateur est positionné, il indique que la police est à taille fixe, sinon elle peut être mise à l'échelle.
averageWidth	ce paramètre est la taille moyenne de caractère de la police en pixels. C'est la moyenne des largeurs de caractères pour les codes 0x61 à 0x7A plus le code 0x20 (à savoir "a" à "z" plus "espace" pour la grille de codage du protocole AS – voir 8.8.1), où la largeur de caractère est la cellule caractère pour les polices à chasse fixe et la somme des largeurs A + B + C pour les polices proportionnelles. Là où la police peut être mise à l'échelle, cette valeur sera calculée pour une police de taille 100 pixels par 100 pixels.
height	ce paramètre est la hauteur de cellule de caractère pour cette police. Là où la police peut être mise à l'échelle, la valeur sera calculée pour une police de taille 100 pixels par 100 pixels.
aspectX	ce paramètre spécifie l'aspect horizontal en pixels par pouce du dispositif pour lequel la police a été créée.
aspectY	ce paramètre spécifie l'aspect vertical en pixels par pouce du dispositif pour lequel la police a été créée.

Tableau 8-39/T.128 – Attributs de Police (Font) *(fin)*

Paramètre	Description
signature1	ce paramètre est le checksum pour le premier groupe de codes. Il est calculé comme étant la somme des largeurs de caractères pour les codes de 0x30 à 0x5A plus 0x24 à 0x26 inclus divisée par 2 et tronquée à 8 bits, où la largeur de caractères est la cellule de caractère pour les polices à chasse fixe et la somme des largeurs A + B + C pour les polices proportionnelles. Là où la police peut être mise à l'échelle, cette valeur sera calculée pour une police de taille 100 pixels par 100 pixels. Voir 8.8.1 pour plus d'information sur les codes et les pages de code. la combinaison de signature1 égale à 0, signature2 égale à 0 et signature3 égale à 0 représente la valeur spéciale NO_SIGNATURE.
signature2	ce paramètre est le checksum pour le second groupe de codes. Il est calculé comme étant la somme des largeurs de caractères pour les codes de 0x20 à 0x7E inclus, moins la somme des largeurs de caractères pour les codes spécifiés dans le calcul de paramètre de la signature1, avec le résultat divisé par 2 et tronqué à 8 bits, où la largeur de caractères est la cellule de caractère pour les polices à chasse fixe et la somme des largeurs A + B + C pour les polices proportionnelles. Là où la police peut être mise à l'échelle, cette valeur sera calculée pour une police de taille 100 pixels par 100 pixels. Voir 8.8.1 pour plus d'information sur les codes et les pages de code. la combinaison de signature1 égale à 0, signature2 égale à 0 et signature3 égale à 0 représente la valeur spéciale NO_SIGNATURE.
signature3	ce paramètre est le checksum pour le troisième groupe de codes. Il est calculé comme étant la somme des largeurs de caractères pour les codes de 0x00 à 0x1E plus 0x80 à 0xFF inclus. Là où la police peut être mise à l'échelle, cette valeur sera calculée pour une police de taille 100 pixels par 100 pixels. voir 8.8.1 pour plus d'information sur les codes et les pages de code. si les valeurs calculées des paramètres signature1, signature2 et signature3 sont toutes à 0, alors signature3 sera mise à zéro. la combinaison de signature1 égale à 0, signature2 égale à 0 et signature3 égale à 0 représente la valeur spéciale NO_SIGNATURE.
codePage	ce paramètre indique quels codes de grille de codage de protocole AS sont supportés par cette police. Les valeurs autorisées sont: • *All defined codepoints* • *Core codepoints only*. Voir 8.8.1 pour plus d'information sur les codes et les pages de code.
nonStandardFontAttributes	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.8.1 Grille de codage

La grille de codage du protocole AS est constituée de la grille de codage ISO/CEI 8859-1 (Latin-1) avec des codes supplémentaires dans les intervalles 0x82-0x8C, 0x91-0x9C plus 0x9F. Les codes dans les intervalles 0x00 à 0x1F, 0x7F à 0x81, 0x8D à 0x90, et les codes 0x9D et 0x9E ne sont pas dans l'ISO/CEI 8859-1 et ne sont pas des extensions AS – ils ne sont donc pas dans la grille de codage du protocole AS.

Les codes dans l'intervalle 0x20 à 0x7E sont considérés comme les codes de la grille de codage du noyau de protocole AS.

Le caractère *Break* de la grille de codage du protocole AS est le code 0x20.

Une ASCE incluera des attributs de police dans un *FontPDU* seulement pour des polices où tous les codes ont des glyphes dans la grille de codage du protocole AS. D'une manière identique, elle enverra des codes dans les ordres *Text* et *Extended Text* seulement là où les glyphes correspondants sont présents dans la grille de codage du protocole AS.

Le Tableau 8-40 résume les codes supportés dans la grille de codage du protocole AS, indiquant si le code est dans l'ISO/CEI 8859-1 ou dans une extension AS, et met en correspondance les codes Unicode (ISO/CEI 10646-1) et le nom de chaque code correspondant.

Tableau 8-40/T.128 – Grille de codage du protocole AS

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x0020	√	0x0020	ESPACE
0x0021	√	0x0021	POINT D'EXCLAMATION
0x0022	√	0x0022	GUILLEMET
0x0023	√	0x0023	SIGNE NUMERO
0x0024	√	0x0024	SIGNE DOLLAR
0x0025	√	0x0025	SIGNE POUR CENT
0x0026	√	0x0026	A COMMERCIAL
0x0027	√	0x0027	APOSTROPHE
0x0028	√	0x0028	PARENTHESE GAUCHE
0x0029	√	0x0029	PARENTHESE DROITE
0x002A	√	0x002A	ASTERISQUE
0x002B	√	0x002B	SIGNE PLUS
0x002C	√	0x002C	VIRGULE
0x002D	√	0x002D	TIRET-MINUSCULE
0x002E	√	0x002E	POINT
0x002F	√	0x002F	SOLIDE
0x0030	√	0x0030	CHIFFRE ZERO
0x0031	√	0x0031	CHIFFRE UN
0x0032	√	0x0032	CHIFFRE DEUX
0x0033	√	0x0033	CHIFFRE TROIS
0x0034	√	0x0034	CHIFFRE QUATRE
0x0035	√	0x0035	CHIFFRE CINQ
0x0036	√	0x0036	CHIFFRE SIX
0x0037	√	0x0037	CHIFFRE SEPT

Tableau 8-40/T.128 – Grille de codage du protocole AS *(suite)*

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x0038	√	0x0038	CHIFFRE HUIT
0x0039	√	0x0039	CHIFFRE NEUF
0x003A	√	0x003A	DEUX POINTS
0x003B	√	0x003B	POINT VIRGULE
0x003C	√	0x003C	SIGNE INFERIEUR OU EGAL
0x003D	√	0x003D	SIGNE EGAL
0x003E	√	0x003E	SIGNE SUPERIEUR OU EGAL
0x003F	√	0x003F	POINT D'INTERROGATION
0x0040	√	0x0040	A COMMERCIAL
0x0041	√	0x0041	MAJUSCULE LATINE A
0x0042	√	0x0042	MAJUSCULE LATINE B
0x0043	√	0x0043	MAJUSCULE LATINE C
0x0044	√	0x0044	MAJUSCULE LATINE D
0x0045	√	0x0045	MAJUSCULE LATINE E
0x0046	√	0x0046	MAJUSCULE LATINE F
0x0047	√	0x0047	MAJUSCULE LATINE G
0x0048	√	0x0048	MAJUSCULE LATINE H
0x0049	√	0x0049	MAJUSCULE LATINE I
0x004A	√	0x004A	MAJUSCULE LATINE J
0x004B	√	0x004B	MAJUSCULE LATINE K
0x004C	√	0x004C	MAJUSCULE LATINE L
0x004D	√	0x004D	MAJUSCULE LATINE M
0x004E	√	0x004E	MAJUSCULE LATINE N
0x004F	√	0x004F	MAJUSCULE LATINE O
0x0050	√	0x0050	MAJUSCULE LATINE P
0x0051	√	0x0051	MAJUSCULE LATINE Q
0x0052	√	0x0052	MAJUSCULE LATINE R
0x0053	√	0x0053	MAJUSCULE LATINE S
0x0054	√	0x0054	MAJUSCULE LATINE T
0x0055	√	0x0055	MAJUSCULE LATINE U
0x0056	√	0x0056	MAJUSCULE LATINE V
0x0057	√	0x0057	MAJUSCULE LATINE W
0x0058	√	0x0058	MAJUSCULE LATINE X
0x0059	√	0x0059	MAJUSCULE LATINE Y
0x005A	√	0x005A	MAJUSCULE LATINE Z
0x005B	√	0x005B	CROCHET GAUCHE
0x005C	√	0x005C	BARRE RENVERSEE
0x005D	√	0x005D	CROCHET DROIT
0x005E	√	0x005E	ACCENT CIRCONFLEXE
0x005F	√	0x005F	BLANC SOULIGNE

Tableau 8-40/T.128 – Grille de codage du protocole AS *(suite)*

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x0060	√	0x0060	ACCENT GRAVE
0x0061	√	0x0061	MINUSCULE LATINE A
0x0062	√	0x0062	MINUSCULE LATINE B
0x0063	√	0x0063	MINUSCULE LATINE C
0x0064	√	0x0064	MINUSCULE LATINE D
0x0065	√	0x0065	MINUSCULE LATINE E
0x0066	√	0x0066	MINUSCULE LATINE F
0x0067	√	0x0067	MINUSCULE LATINE G
0x0068	√	0x0068	MINUSCULE LATINE H
0x0069	√	0x0069	MINUSCULE LATINE I
0x006A	√	0x006A	MINUSCULE LATINE J
0x006B	√	0x006B	MINUSCULE LATINE K
0x006C	√	0x006C	MINUSCULE LATINE L
0x006D	√	0x006D	MINUSCULE LATINE M
0x006E	√	0x006E	MINUSCULE LATINE N
0x006F	√	0x006F	MINUSCULE LATINE O
0x0070	√	0x0070	MINUSCULE LATINE P
0x0071	√	0x0071	MINUSCULE LATINE Q
0x0072	√	0x0072	MINUSCULE LATINE R
0x0073	√	0x0073	MINUSCULE LATINE S
0x0074	√	0x0074	MINUSCULE LATINE T
0x0075	√	0x0075	MINUSCULE LATINE U
0x0076	√	0x0076	MINUSCULE LATINE V
0x0077	√	0x0077	MINUSCULE LATINE W
0x0078	√	0x0078	MINUSCULE LATINE X
0x0079	√	0x0079	MINUSCULE LATINE Y
0x007A	√	0x007A	MINUSCULE LATINE Z
0x007B	√	0x007B	ACCOLADE GAUCHE
0x007C	√	0x007C	LIGNE VERTICALE
0x007D	√	0x007D	ACCOLADE DROITE
0x007E	√	0x007E	TILDE
0x0082	Extension	0x201A	GUILLEMET SIMPLE BAS A DROITE
0x0083	Extension	0x0192	MINUSCULE LATINE F AVEC CROCHET
0x0084	Extension	0x201E	GUILLEMET DOUBLE BAS A DROITE
0x0085	Extension	0x2026	ELLIPSE HORIZONTALE
0x0086	Extension	0x2020	CROIX
0x0087	Extension	0x2021	CROIX DOUBLE
0x0088	Extension	0x02C6	ACCENT CIRCONFLEXE
0x0089	Extension	0x2030	SIGNE POUR MILLE
0x008A	Extension	0x0160	MAJUSCULE LATINE S AVEC ACCENT CIRCONFLEXE INVERSE

Tableau 8-40/T.128 – Grille de codage du protocole AS *(suite)*

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x008B	Extension	0x2039	GUILLEMET ANGULEUX GAUCHE SIMPLE
0x008C	Extension	0x0152	OE ENTRELACES MAJUSCULE
0x0091	Extension	0x2018	GUILLEMET SIMPLE A GAUCHE
0x0092	Extension	0x2019	GUILLEMET SIMPLE A DROITE
0x0093	Extension	0x201C	GUILLEMET DOUBLE A GAUCHE
0x0094	Extension	0x201D	GUILLEMET DOUBLE A DROITE
0x0095	Extension	0x2022	POINT CENTRE
0x0096	Extension	0x2013	TIRET COURT
0x0097	Extension	0x2014	TIRET LONG
0x0098	Extension	0x02DC	PETITE TILDE
0x0099	Extension	0x2122	SIGNE NOM COMMERCIAL
0x009A	Extension	0x0161	MINUSCULE LATINE S ACCENT CIRCONFLEXE INVERSE
0x009B	Extension	0x203A	GUILLEMET ANGULEUX DROIT SIMPLE
0x009C	Extension	0x0153	OE MINUSCULE ENTRELACES
0x009F	Extension	0x0178	MAJUSCULE LATINE Y AVEC TREMA
0x00A0	√	0x00A0	ESPACE INSECABLE
0x00A1	√	0x00A1	POINT D'EXCLAMATION INVERSE
0x00A2	√	0x00A2	SIGNE CENTIME
0x00A3	√	0x00A3	SIGNE LIVRE
0x00A4	√	0x00A4	SIGNE MONNAIE
0x00A5	√	0x00A5	SIGNE YEN
0x00A6	√	0x00A6	BARRE VERTICALE TIRETEE
0x00A7	√	0x00A7	SIGNE PARAGRAPHE
0x00A8	√	0x00A8	TREMA
0x00A9	√	0x00A9	SIGNE DROITS D'AUTEUR
0x00AA	√	0x00AA	INDICATEUR ORDINAL FEMININ
0x00AB	√	0x00AB	GUILLEMET ANGULEUX GAUCHE DOUBLE
0x00AC	√	0x00AC	SIGNE DE NEGATION
0x00AD	√	0x00AD	TIRET AUTOMATIQUE
0x00AE	√	0x00AE	SIGNE MARQUE DEPOSEE
0x00AF	√	0x00AF	SIGNE DE VOYELLE LONGUE
0x00B0	√	0x00B0	SYMBOLE DEGRE
0x00B1	√	0x00B1	SIGNE PLUS OU MOINS
0x00B2	√	0x00B2	EXPOSANT DEUX
0x00B3	√	0x00B3	EXPOSANT TROIS
0x00B4	√	0x00B4	ACCENT AIGU
0x00B5	√	0x00B5	SYMBOLE MICRO
0x00B6	√	0x00B6	SIGNE ALINEA
0x00B7	√	0x00B7	POINT MEDIAN
0x00B8	√	0x00B8	CEDILLE

Tableau 8-40/T.128 – Grille de codage du protocole AS *(suite)*

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x00B9	√	0x00B9	EXPOSANT UN
0x00BA	√	0x00BA	INDICATEUR ORDINAL MASCULIN
0x00BB	√	0x00BB	GUILLEMET ANGULEUX DROITE DOUBLE
0x00BC	√	0x00BC	FRACTION UN QUART
0x00BD	√	0x00BD	FRACTION UN DEMI
0x00BE	√	0x00BE	FRACTION TROIS QUARTS
0x00BF	√	0x00BF	POINT D'INTERROGATION INVERSE
0x00C0	√	0x00C0	MAJUSCULE LATINE A ACCENT GRAVE
0x00C1	√	0x00C1	MAJUSCULE LATINE A ACCENT AIGU
0x00C2	√	0x00C2	MAJUSCULE LATINE A ACCENT CIRCONFLEXE
0x00C3	√	0x00C3	MAJUSCULE LATINE A TILDE
0x00C4	√	0x00C4	MAJUSCULE LATINE A TREMA
0x00C5	√	0x00C5	MAJUSCULE LATINE A ROND
0x00C6	√	0x00C6	MAJUSCULE LATINE AE ENTRELACES
0x00C7	√	0x00C7	MAJUSCULE LATINE C CEDILLE
0x00C8	√	0x00C8	MAJUSCULE LATINE E ACCENT GRAVE
0x00C9	√	0x00C9	MAJUSCULE LATINE E ACCENT AIGU
0x00CA	√	0x00CA	MAJUSCULE LATINE E ACCENT CIRCONFLEXE
0x00CB	√	0x00CB	MAJUSCULE LATINE E TREMA
0x00CC	√	0x00CC	MAJUSCULE LATINE I ACCENT GRAVE
0x00CD	√	0x00CD	MAJUSCULE LATINE I ACCENT AIGU
0x00CE	√	0x00CE	MAJUSCULE LATINE I ACCENT CIRCONFLEXE
0x00CF	√	0x00CF	MAJUSCULE LATINE I TREMA
0x00D0	√	0x00D0	MAJUSCULE LATINE ETH
0x00D1	√	0x00D1	MAJUSCULE LATINE N TILDE
0x00D2	√	0x00D2	MAJUSCULE LATINE O ACCENT GRAVE
0x00D3	√	0x00D3	MAJUSCULE LATINE O AVEC ACCENT
0x00D4	√	0x00D4	MAJUSCULE LATINE O AVEC ACCENT CIRCONFLEXE
0x00D5	√	0x00D5	MAJUSCULE LATINE O TILDE
0x00D6	√	0x00D6	MAJUSCULE LATINE O TREMA
0x00D7	√	0x00D7	SIGNE DE MULTIPLICATION
0x00D8	√	0x00D8	MAJUSCULE LATINE O BARRE
0x00D9	√	0x00D9	MAJUSCULE LATINE U AVEC ACCENT GRAVE
0x00DA	√	0x00DA	MAJUSCULE LATINE U AVEC ACCENT AIGU
0x00DB	√	0x00DB	MAJUSCULE LATINE U AVEC ACCENT CIRCONFLEXE
0x00DC	√	0x00DC	MAJUSCULE LATINE U TREMA
0x00DD	√	0x00DD	MAJUSCULE LATINE Y AVEC ACCENT AIGU
0x00DE	√	0x00DE	MAJUSCULE LATINE TH DUR
0x00DF	√	0x00DF	MINUSCULE LATINE ESZETT

Tableau 8-40/T.128 – Grille de codage du protocole AS *(fin)*

Code	ISO/CEI 8859-1	Code Unicode	Nom Unicode
0x00E0	√	0x00E0	MINUSCULE LATINE A AVEC ACCENT GRAVE
0x00E1	√	0x00E1	MINUSCULE LATINE A AVEC ACCENT AIGU
0x00E2	√	0x00E2	MINUSCULE LATINE A AVEC ACCENT CIRCONFLEXE
0x00E3	√	0x00E3	MINUSCULE LATINE A TILDE
0x00E4	√	0x00E4	MINUSCULE LATINE A TREMA
0x00E5	√	0x00E5	MINUSCULE LATINE A ROND SUPERIEUR
0x00E6	√	0x00E6	MINUSCULE LATINE AE ENTRELACES
0x00E7	√	0x00E7	MINUSCULE LATINE C CEDILLE
0x00E8	√	0x00E8	MINUSCULE LATINE E AVEC ACCENT GRAVE
0x00E9	√	0x00E9	MINUSCULE LATINE E AVEC ACCENT AIGU
0x00EA	√	0x00EA	MINUSCULE LATINE E AVEC ACCENT CIRCONFLEXE
0x00EB	√	0x00EB	MINUSCULE LATINE E TREMA
0x00EC	√	0x00EC	MINUSCULE LATINE I AVEC ACCENT GRAVE
0x00ED	√	0x00ED	MINUSCULE LATINE I AVEC ACCENT
0x00EE	√	0x00EE	MINUSCULE LATINE I AVEC ACCENT CIRCONFLEXE
0x00EF	√	0x00EF	MINUSCULE LATINE I AVEC TREMA
0x00F0	√	0x00F0	LETTRE MINUSCULE LATINE ETH (ISLANDAIS)
0x00F1	√	0x00F1	MINUSCULE LATINE N TILDE
0x00F2	√	0x00F2	MINUSCULE LATINE O AVEC ACCENT GRAVE
0x00F3	√	0x00F3	MINUSCULE LATINE O AVEC ACCENT
0x00F4	√	0x00F4	MINUSCULE LATINE O AVEC ACCENT CIRCONFLEXE
0x00F5	√	0x00F5	MINUSCULE LATINE O TILDE
0x00F6	√	0x00F6	MINUSCULE LATINE O AVEC TREMA
0x00F7	√	0x00F7	SIGNE DE DIVISION
0x00F8	√	0x00F8	MINUSCULE LATINE O BARRE
0x00F9	√	0x00F9	MINUSCULE LATINE U AVEC ACCENT GRAVE
0x00FA	√	0x00FA	MINUSCULE LATINE U AVEC ACCENT AIGU
0x00FB	√	0x00FB	MINUSCULE LATINE U AVEC ACCENT CIRCONFLEXE
0x00FC	√	0x00FC	MINUSCULE LATINE U TREMA
0x00FD	√	0x00FD	MINUSCULE LATINE Y AVEC ACCENT AIGU
0x00FE	√	0x00FE	MINUSCULE LATINE TH DUR
0x00FF	√	0x00FF	MINUSCULE LATINE Y TREMA

8.8.2 Correspondance de Police

Une ASCE utilisera l'algorithme de correspondance de police décrit dans le présent sous-paragraphe pour déterminer le jeu courant de polices mises en correspondance pour toutes les ASCE actives (elle comprise).

Une ASCE déterminera le jeu courant de polices mises en correspondance en comparant son information de police locale (à savoir les attributs de police envoyés dans son dernier *FontPDU*) avec les attributs de police fournis par chacune des autres ASCE actives, utilisant les critères de correspondance de police décrits dans le Tableau 8-41 ci-dessous. Les critères de correspondance sont appliqués successivement aux paramètres *FontPDU* deux par deux en série et on ajoute une police au jeu courant lorsqu'une correspondance est trouvée avec toutes les autres ASCE actives. Ceci signifie que:

- toutes les ASCE actives doivent mettre en correspondance une police pour qu'elle soit ajoutée au jeu courant de polices mises en correspondance;
- là où une autre ASCE active n'a pas encore fourni d'attributs de police, il n'y a pas de police courante mise en correspondance.

Le résultat de la projection de police est un mappage pour chaque police mise en correspondance entre les *FondIDs* locaux et distants de chacune des autres ASCE actives. Voir 8.8 pour plus d'information sur les *FontIDs*. Les mappages croisés de polices sont utilisés comme suit.

- lors de l'envoi d'ordres *Text* et *Extended Text* pour une police locale mise en correspondance, l'ASCE émettrice positionnera le paramètre *FontID* au *FontID* de cette police dans le dernier *FontPDU* envoyé;
- lors de la réception d'ordres *Text* et *Extended Text*, l'ASCE réceptrice mappera le *FontID* de cet ordre avec la police locale ayant été mise en correspondance dans le dernier *FontPDU* de cette ASCE.

Ce mécanisme de mappage permet aux ASCE d'utiliser un simple jeu de *FontIDs* lors de l'envoi de *Text* et *Extended Text* et laisse à la responsabilité des ASCE réceptrices le mappage du *FontID* de cet ordre avec la police locale du terminal.

L'utilisation du même algorithme de mise en correspondance de police assure que chaque doublet d'ASCE s'accorde sur un jeu identique de polices mises en correspondance et donc, génère un jeu identique de mappages de police interopérables.

Le Tableau 8-41 décrit les critères de correspondance minimaux pour chaque paramètre de police (correspondant au paramètre du *FontPDU* valide) considéré pendant la mise en correspondance de police, décrits en termes de police locale et d'une autre police distante.

Le Tableau 8-42 fournit une information sur les types croisés autorisés référencés dans le Tableau 8-41. Une ASCE peut ou non autoriser les croisements approximatifs, basés sur des mécanismes purement locaux (tels que les options locales de configuration de terminal). Là où une ASCE ne permet pas les croisements approximatifs, seuls les exacts seront permis. Là où une ASCE permet vraiment les croisements approximatifs, elle peut le faire sur la base de certains critères de croisement (à nouveau dépendants de mécanismes locaux), supposant que les capacités négociées correspondantes autorisent ce type de croisement.

Un doublet de polices est dit à croiser lorsque chaque paramètre de police à considérer dans le Tableau 8-41 est en correspondance exacte ou satisfait aux exigences minimales des types de projection définis dans le Tableau 8-41 (dépendant de la configuration de croisement approximative locale sur l'ASCE). Par exemple, une ASCE particulière peut autoriser la projection approximative pour *Delta X Position* mais pas pour *Code Page* et, en supposant que les capacités négociées permettent la simulation de *Delta X*, peut ensuite demander qu'un doublet de polices soit en correspondance exacte avec les paramètres à considérer définis comme requérant des types de projection minimale pour *Exact* et *Code Page*, mais requérant seulement une projection approximative pour les paramètres définis comme requérant le type de correspondance minimale pour *Delta X Position*.

Tableau 8-41/T.128 – Critères de croisement de police

Paramètre	Critère	Type de croisement
faceName	les noms faciaux courant et distant sont identiques.	*Exact*
fontFlags Fixed Pitch	les polices sont toutes deux à chasse fixe OU toutes deux à classe variable.	*Exact*
codePage	les deux polices supportent tous les codes. la police locale OU bien la police distante supporte seulement les codes du noyau.	*Exact* *Code Page*
fontFlags Fixed Size	les polices sont toutes deux à corps fixe OU toutes deux à corps variable. la police locale est à corps fixe et la police distante est à corps variable. (Note 1)	*Exact* *Exact*
averageWidth and height	les deux polices sont à corps fixe et les valeurs moyennes en largeur et en hauteur sont identiques. (Note 2)	*Exact*
signature1, signature2 and signature3	ces critères dépendent des capacités négociées pour la vérification de signature de police (à savoir, dans le mode hérité de l'indicateur binaire *Check font signatures* de la capacité *Order.textFlags*; dans le mode de base de la capacité négociée *Order.checkFontSignatures*). • la vérification de signature de police est validée et les valeurs des signature1, signature2 et signature3 sont identiques. • la vérification de signature de police est validée et les valeurs des signature1, signature2 sont identiques.	 *Exact* *Delta X Position*
	• la vérification de signature de police est validée et les valeurs des signature1, signature2 et signature3 ne sont pas identiques ou bien la police possède la valeur spéciale NO_SIGNATURE. • la vérification de signature de police n'est pas validée.	*Delta X Position* *Delta X Position*
aspectX and aspectY	ces critères dépendent des capacités négociées pour ce qui concerne la vérification d'aspect de police (à savoir, dans le mode hérité de l'indicateur binaire *Check font aspect* de la capacité *Order.textFlags*; dans le mode de base de la capacité négociée *Order.checkFontAspectFlag*). • la vérification d'aspect de police est validée et les valeurs *aspectX* et *aspectY* locales et distantes sont identiques. • la vérification d'aspect de police n'est pas validée.	 *Exact* *Delta X Position*

NOTE 1 – Ceci fait l'hypothèse qu'une ASCE peut mettre en correspondance des polices locales de taille fixe avec des polices distantes de taille variable et ultérieurement d'envoyer des ordres *Text* et *Extended Text* pour cette police, en se fiant aux ASCE locales pour mettre à l'échelle le texte de manière appropriée. On suppose aussi que l'inverse (à savoir projeter des tailles variables sur des tailles fixes et envoyer des tailles variables) n'est pas autorisé.

NOTE 2 – Si une police est à taille variable, ce critère n'est pas considéré.

Tableau 8-42/T.128 – Types de croisement de police

Type de correspondance	Définition
Exact	les deux paramètres de police considérés doivent se correspondre exactement.
Code Page	là où l'une ou les deux grilles de codage des polices considérées sont conformes seulement à la grille de codage du noyau du protocole AS, une ASCE peut autoriser un croisement de grille de codage approximatif. Lorsqu'elle le fait, elle restreindra ultérieurement les codes de cette police contenus dans les ordres *Text* et *Extended Text* à la grille de codage du noyau du protocole AS. Voir 8.16.11 et 8.16.12 pour plus d'information sur les ordres *Text* et *Extended Text*.
Delta X Position	là où l'information de position pour l'une ou les deux polices considérées ne correspond pas exactement, une ASCE peut autoriser une projection de *Delta X Position* approximative. Une ASCE autorisera seulement les projections approximatives en *Delta X* là où les capacités négociées permettent la simulation *Delta X* (à savoir dans le mode hérité l'indicateur binaire *Order.textFlags Allow DeltaX* est positionné; dans le mode de base *Order.allowDeltaXFlag* vaut TRUE) et simulera ensuite l'information *position X* du texte pour la police utilisant l'information *Delta X* dans les ordres *Extended Text*. Voir 8.16.12 pour plus d'information sur l'ordre *Extended Text* et la simulation *Delta X*.

8.8.3 Aliasing de police

La correspondance de police agit sur les attributs de police dans les *FontPDUs*. Ceci permet à une ASCE d'effectuer l'aliasing de polices sur le terminal local et sur les attributs de police qu'elle envoie dans les *FontPDUs*. C'est une décision d'implémentation locale d'ASCE, mais qui est particulièrement utile dans les scénarios suivants:

- les polices identiques (ou très voisines) sont disponibles auprès d'une variété de fournisseurs de polices pour ce qui concerne le terminal local (mais utilisent des noms faciaux différents). Ici, une ASCE peut mapper les polices du terminal local avec un nom facial générique (indépendant d'un fournisseur) pour augmenter la probabilité de correspondance de police;
- les polices identiques (ou très voisines) ont des noms faciaux différents sur des types de terminaux différents. Ici une ASCE peut mapper les polices du terminal local avec un nouveau nom (ou nom supplémentaire) facial pour augmenter la probabilité de correspondance de police.

Considérons par exemple deux ASCE où ASCE A supporte Font_Supplier_A:Courier et Font_Supplier_B:Courier et ASCE B supporte juste Courier. Si ASCE A fournit des attributs de police pour Font_Supplier_A:Courier, Font_Supplier_B:Courier et deux Couriers aliasés (correspondants au deux Couriers spécifiques d'un fournisseur), alors (supposant que les autres critères correspondent), les deux ASCE sont capables d'envoyer des ordres pour tout ou partie de texte *Courier* local.

8.9 Gestion d'application

Pendant une session AS, une ASCE peut héberger une ou plusieurs applications, chacune étant composée d'une collection de fenêtres hébergées partagées par des ASCE actives homologues.

Une ASCE notifiera aux ASCE homologues actives lorsque le nombre d'applications hébergées sur le terminal local de l'ASCE changera, en envoyant un *ApplicationPDU* avec *NotifyHostedApplications* contenant ce nouveau nombre à toutes les ASCE dans la conférence, de la manière indiquée dans le Tableau 6-3. Le contenu de *ApplicationPDU* est décrit dans le Tableau 8-43.

Ceci fournit une information au fil de l'eau pour activer les autres ASCE sur le nombre d'applications hébergées par cette ASCE, information qu'elles peuvent utiliser pour fournir de l'information à l'utilisateur final. Les ASCE réceptrices peuvent utiliser cette ASPDU pour superviser l'instant où les ASCE distantes arrêtent d'héberger des applications – ce qui peut leur permettre de libérer des ressources allouées aux ASCE hôtes. Voir 8.6 pour plus d'information sur la synchronisation.

Une ASCE peut commencer et arrêter d'héberger des applications locales à chaque instant. Ceci est une affaire strictement locale et peut être pilotée par l'utilisateur final, le comportement du terminal local et/ou l'initialisation ou la fin de la conférence.

Cependant, il existe des situations où il est utile d'arrêter d'héberger des applications distantes. Par exemple, là où une ASCE contrôle des applications hébergées qui sont partagées à distance (Voir 8.7) ou lorsque l'utilisateur final est inexpérimenté. Une ASCE peut demander à une ASCE homologue d'arrêter d'héberger une application sur le terminal de l'ASCE homologue en envoyant un *ApplicationPDU* avec l'action *UnhostApplication* contenant un *windowID* pour l'application à arrêter de la manière indiquée dans le Tableau 6-3. Le *windowID* fourni devrait être un *hosted_window_ID* appartenant à l'application et obtenu à partir du *WindowListPDU* le plus récent envoyé par l'ASCE hôte (voir 8.10). Une ASCE enverra seulement un *ApplicationPDU* avec l'action *UnhostApplication* là où la capacité *General.remoteUnshareFlag* des ASCE homologues vaut TRUE.

Sur réception d'une *ApplicationPDU* avec *UnhostApplication* contenant un *windowID* pour une application hébergée sur le terminal local, une ASCE cessera d'héberger l'application concernée.

Tableau 8-43/T.128 – ApplicationPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
action	ce paramètre identifie l'action *ApplicationPDU* particulière. Les valeurs autorisées sont *NotifyHostedApplications* ou *UnhostApplication*.
numberApplications	ce paramètre indique le nombre d'applications présentées par l'ASCE émettrice. Ce paramètre est seulement valable là où l'action est *NotifyHostedApplications*.
windowID	ce paramètre spécifie la fenêtre de sommet possédée par une application présentée sur l'ASCE homologue dont il faut arrêter la présentation. Ce paramètre est seulement valable là où l'action est *UnhostApplications*
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.10 Gestion de liste de fenêtres

Pendant une session AS, là où une ASCE héberge une ou plusieurs fenêtres partagées avec des ASCE actives homologues, cela revient à gérer une collection de fenêtres qui peuvent être:

- hébergées: les fenêtres hébergées sont possédées par une application hébergée sur le terminal local. Pour chaque fenêtre hébergée, il y a une fenêtre reflet correspondante sur chaque ASCE homologue;
- reflets: les fenêtres reflets sont tracées par l'ASCE et correspondent à une fenêtre hébergée sur une ASCE homologue particulière;

- locales: les fenêtres locales ne sont pas partagées – les sorties de leurs applications ne sont visibles que sur le terminal local.

On demande à une ASCE de seulement surveiller les fenêtres hébergées. Sur certains terminaux et/ou gestionnaires de fenêtres de terminaux, un utilisateur final ou une action programmée peut rendre les fenêtres d'application temporairement invisibles (à savoir qu'elles sont retirées de l'écran du terminal) – par exemple, lorsqu'une application de supervision système permanente reste cachée jusqu'à l'arrivée d'un événement d'alarme requérant une intervention de l'utilisateur, et redevient à nouveau visible. Là où une fenêtre locale hébergée devient invisible, alors on ne demande pas à une ASCE de surveiller la fenêtre, ce qui ne devrait pas fournir d'information de mise à jour de la liste de fenêtres pour cette fenêtre-ci (voir ci-dessous).

Là où une fenêtre hébergée est masquée par une ou plusieurs fenêtres locales et que l'ASCE ne peut pas obtenir d'information de tracé valide pour cette fenêtre hébergée, le protocole AS requiert que l'ASCE hôte inclue la où les fenêtres locales masquées dans la liste de fenêtres appropriée (voir ci-dessous) et que les ASCE réceptrices marquent de telles zones masquées (d'une manière déterminée localement) pour indiquer que les contenus ne sont pas nécessairement valides.

Il faut noter qu'une ASCE à besoin de surveiller les fenêtres locales seulement lorsqu'elle les héberge et n'a besoin de surveiller que celles qui masquent les fenêtres hébergées l'empêchant ainsi d'obtenir une information de tracé valide pour ces fenêtres hébergées. Par exemple, une ASCE peut afficher des fenêtres hébergées et reflets dans une zone d'écran dédiée du terminal de laquelle les fenêtres locales sont exclues. D'une manière identique, un terminal particulier peut ne pas supporter du tout d'application locale (et donc de fenêtre). Hors, sur certains équipements de terminaux, une ASCE peut encore être capable d'obtenir de l'information de tracé valide de la part de fenêtres hébergées lorsqu'elles sont masquées par des fenêtres locales. Dans ces cas là, on ne demandera pas à une ASCE de surveiller les fenêtres locales.

Une ASCE enverra un *WindowListPDU* à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3, lorsque:

- elle commence ou arrête d'héberger des fenêtres;
- elle héberge des fenêtres et détecte un changement d'ordre en Z de la fenêtre visible hébergée;
- elle héberge des fenêtres et une ou plusieurs de ses positions de fenêtres visibles hébergées changent;
- elle héberge des fenêtres et détecte un ordre en Z de fenêtre locale ou un changement de position modifiant le masquage des fenêtres visibles hébergées.

Ceci signifie qu'une ASCE envoie seulement un *WindowListPDU* lorsqu'elle héberge des fenêtres visibles ou dans le premier cas, lorsqu'elle réalise une transition vers ou à partir de fenêtres hôtes. Le contenu de *WindowListPDU* est décrit dans le Tableau 8-44. *WindowListPDU* contient des informations sur deux classes de fenêtres.

- la vue de l'ASCE émettrice sur toutes les fenêtres visibles hébergées ou réfléchies de la conférence. Cette vue est (généralement – mais voir ci-dessous) commune à toutes les ASCE actives, comme elle inclut les fenêtres hébergées et reflets – et une fenêtre incluse dans cette liste par une ASCE parce qu'elle est hébergée sera incluse dans les listes correspondantes par les autres ASCE parce qu'elle est réfléchie;
- toute fenêtre locale d'ASCE émettrice masquant au moins une partie d'au moins une fenêtre visible hébergée. Cette information est possédée et générée par une ASCE particulière – comme elle est reliée aux fenêtres locales, il n'y a aucun recouvrement avec l'information identique générée par des ASCE homologues.

WindowListPDU contient une liste simple de fenêtres, contenant les fenêtres hébergées et reflets et les fenêtres locales satisfaisant aux critères de masquage, ordonnée de sorte que la première fenêtre corresponde à la fenêtre du dessus (la plus en avant) et la dernière à celle du fond (la plus en arrière) selon l'ordre en Z du terminal local.

Pour chaque fenêtre dans la liste de fenêtres, l'ASCE fournit une information d'ordre en Z (implicitement par position dans la liste), une position et une taille, une information de propriété et tout qualificatif approprié (tel que si la fenêtre est *minimized*).

Le protocole AS suppose que l'information fournie pour les fenêtres hébergées et reflets dans la liste de fenêtres est normalement semblable sur toutes les ASCE actives. Celle-ci ne supporte pas l'indépendance de l'ordre en Z, la taille et le positionnement ou la qualification (telle que *minimized*) de fenêtres reflets par rapport à leur fenêtre hébergée correspondante. A savoir, le protocole AS requiert l'application d'un changement équivalent sur les fenêtres reflets correspondantes sur toutes les autres ASCE actives.

En pratique, des différences entre listes de fenêtres dans les fenêtres hébergées et reflets peut se produire, là où le *WindowListPDU* reflétant un changement de liste de fenêtres n'a pas encore été reçu et/ou traité par les autres ASCE. Une bonne partie de la complexité dans la manipulation de listes de fenêtres a un lien avec la résolution des échanges intempestifs concernant les mises à jour de liste de fenêtres. La description suivante ignore la possibilité d'échanges intempestifs concernant les listes de fenêtres et de leur résolution. Cet aspect est couvert par 8.10.1.

Sur réception d'un *WindowListPDU*, une ASCE effectue en pratique les opérations suivantes – bien que leur spécificité dépende de l'environnement du terminal.

- Elle traite la liste de fenêtres *WindowListPDU* pour:
  - enlever toutes les anciennes fenêtres reflets;
  - créer toutes les nouvelles fenêtres reflets;
  - mettre à jour la taille et/ou la position des fenêtres ayant été changées;
  - ajuster l'ordre d'emplacement des fenêtres hébergées et reflets sur le terminal local de manière à obtenir le même ordre relatif que sur la liste des fenêtres, en prenant soin de maintenir la correspondance entre l'ordre d'emplacement et les applications locales.
- Elle détermine la nouvelle zone de fenêtres masquées – ce qui détermine quelles sont les zones de fenêtres reflets valides pour le tracé – sur la base de la taille de moniteur de l'ASCE émettrice et de toutes les fenêtres locales masquantes dans *WindowListPDU*.

On demande à une ASCE d'appliquer les changements de liste de fenêtres aussi exactement que possible en liaison avec les contraintes de l'environnement du terminal local.

Tableau 8-44/T.128 – WindowListPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
listTime	ce paramètre est le temps local de l'ASCE en millisecondes correspondant au moment de la construction de ce *WindowsListPDU*. Il est utilisé en association avec le paramètrre *listID* (voir ci-dessous) pour résoudre les échanges intempestifs de *WindowListPDU*. Voir 8.10.1 ci-dessous pour plus d'information.
listID	ce paramètre est l'identificateur attribué à l'ASCE émettrice au moment de la construction de ce *WindowListPDU*. Il est utilisé en association avec le paramètre *listTime* (voir ci-dessus) pour résoudre les échanges intempestifs de *WindowListPDU*. Voir 8.10.1 pour plus d'information.
windowAttributeList	ce paramètre est une liste d'attributs de fenêtres (voir le Tableau 8-45) décrivant la structure de fenêtres de l'ASCE émettrice. La liste est en ordre Z, de telle sorte que la première fenêtre soit sur le dessus (la plus en avant) et la dernière soit au fond (la plus en arrière) selon l'ordre en Z du terminal local.
windowTitleList	ce paramètre est une liste de titres de fenêtres. La liste est en ordre Z, de telle sorte que la première fenêtre soit sur le dessus (la plus en avant) et la dernière soit au fond (la plus en arrière) selon l'ordre en Z du terminal local. Chaque titre est soit la valeur spéciale NO_TITLE ou une chaîne de texte T.50 décrivant la fenêtre.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

De nombreux gestionnaires de fenêtres fournissent des facilités locales (telles qu'une liste de tâches, une barre de tâches ou un bandeau d'icônes) pour permettre aux utilisateurs d'activer ou d'arranger les fenêtres. De telles facilités sont typiquement restreintes aux fenêtres du dessus – de sorte que les utilisateurs puissent visualiser l'ensemble des fenêtres comprises dans l'application. Une ASCE peut fournir un titre pour une fenêtre hébergée où le titre a pour vocation d'aider les utilisateurs sur les ASCE homologues à identifier plus facilement de manière lisible la fenêtre reflet correspondante. On recommande que les ASCE ne fournissent des titres que pour les fenêtres hébergées. Là où la fenêtre est reflet ou locale, l'ASCE devrait fournir la chaîne de titre spéciale NO_TITLE.

Ce qui suit requiert qu'une ASCE affecte un identificateur local unique pour chaque application hébergée (son *hosted_application_ID*) et un identificateur local unique pour chaque fenêtre hébergée (*hosted_window_ID*).

Tableau 8-45/T.128 – Attributs de fenêtres

Paramètre	Description
windowID	ce paramètre est l'identificateur de fenêtre pour cette fenêtre. • là où la fenêtre est hébergée, ce paramètre est le *hosted_window_ID* affecté par l'ASCE émettrice. • là où la fenêtre est une fenêtre reflet, ce paramètre est le *hosted_window_ID* affecté par la fenêtre propriétaire (qui a été fourni dans le dernier *WindowListPDU* pour la fenêtre hébergée correspondante).
windowExtra	ce paramètre est l'identificateur de fenêtre pour cette fenêtre. • là où la fenêtre est hébergée, ce paramètre est le *hosted_application_ID* affecté par l'ASCE émettrice. • là où la fenêtre est une fenêtre reflet, ce paramètre est le *MCS User ID* de l'ASCE hôte.
windowOwner	ce paramètre indique la fenêtre propriétaire de cette fenêtre. • là où la fenêtre est hébergée, ce paramètre est le *hosted_window_ID* de la fenêtre propriétaire (à savoir, la fenêtre parente de cette fenêtre dans la hiérarchie de fenêtres du terminal local). Il faut noter que là où la fenêtre est hébergée et que la fenêtre propriétaire est le moniteur, ce paramètre sera égal à zéro.
windowFlags	ce paramètre est un jeu d'indicateurs binaires permettant de qualifier la fenêtre. Les valeurs d'indicateurs définies sont les suivantes: • *Minimized* (fenêtres hébergées seulement) • *Taggable* (fenêtres hébergées seulement) • *Hosted* • *Shadow* • *Local* • *Always On Top* (fenêtres hébergées seulement) • *Window Manager Minimized* (fenêtres hébergées seulement) • *Window Manager Invisible* (fenêtres hébergées seulement) voir ci-dessous pour plus d'information sur l'interprétation des indicateurs binaires.
windowLeft	ce paramètre est la coordonnée X gauche de la fenêtre sur le moniteur virtuel. Pour les fenêtres locales et hébergées, ce paramètre sera la coordonnée X gauche de la fenêtre (pouvant être en dehors du moniteur local).
windowTop	ce paramètre est la coordonnée Y gauche de la fenêtre sur le moniteur virtuel. Pour les fenêtres locales et hébergées, ce paramètre sera la coordonnée Y gauche de la fenêtre (pouvant être en dehors du moniteur local).
windowRight	ce paramètre est la coordonnée X droite de la fenêtre sur le moniteur virtuel. Pour les fenêtres locales et hébergées, ce paramètre sera la coordonnée X droite de la fenêtre (pouvant être en dehors du moniteur local).

Tableau 8-45/T.128 – Attributs de fenêtres *(fin)*

Paramètre	Description
windowBottom	ce paramètre est la coordonnée Y droite de la fenêtre sur le moniteur virtuel. Pour les fenêtres locales et hébergées, ce paramètre sera la coordonnée Y droite de la fenêtre (pouvant être en dehors du moniteur local).
nonStandardWindowAttributes	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Etiquetable (Taggable)

Les utilisateurs peuvent éprouver des difficultés à identifier les fenêtres appartenant à un utilisateur donné dans la conférence. C'est un problème lorsque l'utilisateur sélectionne un attribut d'application qui aura une incidence sur le terminal hôte (par exemple sauvegarder un fichier) et/ou lorsque de multiples ASCE dans la conférence sont en train d'héberger des applications, augmentant ainsi le nombre de fenêtres sur le moniteur du terminal local. L'indicateur binaire *Taggable* de l'attribut de fenêtre fournit un mécanisme dans lequel les ASCE fournissent de l'information d'interface utilisateur supplémentaire concernant l'emplacement de la fenêtre hébergée.

Une ASCE construisant une mise à jour de liste de fenêtres devrait utiliser l'indicateur binaire *Taggable* pour indiquer si la ou les fenêtres sont éligibles pour être étiquetées. Il est recommandé que les ASCE émettrices ne positionnent que les fenêtres du dessus comme *Taggable* et n'étiquettent pas les fenêtres secondaires ou temporaires comme les bandeaux d'aide ou les titres d'icônes. L'algorithme utilisé pour déterminer si une fenêtre est étiquetable ou non est une affaire purement locale, mais il est recommandé que les ASCE marquent comme *Taggable* seulement les fenêtres qu'un utilisateur devrait percevoir comme constituant un dialogue ou une fenêtre distincte d'application principale. Une ASCE recevant une liste de fenêtres contenant des fenêtres hébergées avec l'indicateur binaire *Taggable* positionné peut fournir pour chaque fenêtre reflet des signaux d'interface utilisateur supplémentaires (tels que les étiquettes de fenêtres et/ou une augmentation du titre de la fenêtre) identifiant le nœud possédant les fenêtres hébergées correspondantes.

Réduction de fenêtres (Minimized Windows)

Certains terminaux supportent le concept de réduction de fenêtre, par lequel une fenêtre est réduite à une icône ou une fenêtre plus petite, pouvant ensuite être affichée sur le moniteur du terminal local ou dans une fenêtre possédée par le gestionnaire de fenêtres du terminal (tel qu'une liste de tâches de gestionnaire de fenêtres, une barre de tâches ou un bandeau d'icônes). Là où une application est composée de plusieurs fenêtres, alors réduire l'application a pour résultat de réduire les fenêtres du dessus de l'application et de rendre invisible les autres fenêtres de rang inférieur.

Les nombreux styles de réduction de fenêtres pour les fenêtres hébergées peuvent être représentés dans des mises à jour de liste de fenêtres utilisant plusieurs attributs *windowsFlags*, c'est-à-dire les indicateurs binaires *Minimized, Window Manager Minimized* et *Window Manager Invisible*.

- l'indicateur binaire *Minimized* devrait être positionné par une ASCE lorsqu'une fenêtre hébergée est réduite – sans respecter le comportement de réduction locale particulier;
- l'indicateur binaire *Window Manager Minimized* devrait être positionné par une ASCE lorsqu'une fenêtre hébergée est réduite à une fenêtre de gestionnaire de fenêtre locale. Cet indicateur binaire est typiquement appliqué aux fenêtres du dessus;

- l'indicateur binaire *Window Manager Invisible* devrait être positionné par une ASCE lorsqu'une fenêtre hébergée est rendue invisible comme faisant partie du mécanisme local de réduction de gestionnaire de fenêtre et là où l'indicateur *Window Manager Minimized* est attribué à une fenêtre parente – à savoir là où les fenêtres hébergées d'une application du niveau le plus élevé sont réduites à une fenêtre locale de gestionnaire de fenêtre. Cet indicateur ne devrait pas être positionné là où les fenêtres hébergées du niveau le plus élevé sont réduites sur le moniteur.

L'utilisation de ces indicateurs binaires ne préjuge pas d'un comportement de réduction de fenêtre du terminal local ou de celui d'un gestionnaire de fenêtre du terminal local. Les ASCE hôtes devraient affecter la combinaison autorisée d'indicateurs binaires exprimant le mieux les caractéristiques de chaque fenêtre hébergée ayant été réduite. D'une manière identique, une ASCE traitant une mise à jour de liste de fenêtres contenant des fenêtres hébergées avec l'un ou plusieurs de ces indicateurs binaires ayant été positionné, devrait présenter les fenêtres reflets correspondantes d'une manière adaptée au comportement de réduction du terminal local et/ou du gestionnaire de fenêtre. Cette approche permet aux ASCE de supporter les nombreux styles de réduction au travers de multiples types de terminaux, avec chaque ASCE présentant les fenêtres reflets d'une manière adaptée au terminal local.

Toujours au premier plan *(always on top)*

Certains types de terminaux supportent le concept de fenêtre *always on top*, où de telles fenêtres sont affichées normalement au-dessus de toutes les autres. Par exemple ces fenêtres sont souvent utilisées pour des barres d'outils de suite d'application. Là où le terminal local supporte toujours les fenêtres *always on top*, il peut y a voir plusieurs fenêtres traitées de cette manière. Typiquement elles ne font pas partie de l'ordre en Z principal du terminal mais sont plutôt gérées en tant que membres d'un ordre en Z *always on top* distinct. A l'intérieur de cet ordre en Z distinct, les fenêtres *always on top* peuvent changer dans l'ordre en Z les unes par rapport aux autres, mais restent toujours au-dessus des fenêtres ordinaires.

Là où les fenêtres *always on top* sont supportées sur un terminal particulier, l'indicateur binaire *Always On Top* devrait être positionné par une ASCE lorsqu'une fenêtre hébergée est traitée localement comme étant toujours sur le dessus. Cet indicateur binaire est typiquement appliqué aux fenêtres hébergées qui sont sur le devant de l'écran. Les ASCE devraient s'assurer que les fenêtres hébergées ayant été marquées comme *Always On Top* sont plus en avant que les fenêtres n'étant pas marquées de cette manière dans la liste de fenêtres *WindowListPDU*. Une ASCE traitant une mise à jour de liste de fenêtres contenant des fenêtres hébergées avec l'indicateur binaire *Always On Top* devraient essayer de faire en sorte que les fenêtres reflets correspondantes se comportent comme étant toujours sur le dessus dans l'environnement du terminal local – ce qui peut être réalisé en utilisant une facilité semblable dans l'environnement du terminal local ou en manipulant l'ordre en Z local pour obtenir l'effet désiré.

8.10.1 Echanges intempestifs d'ordre en Z de liste de fenêtres

Comme discuté plus haut, chaque ASCE active dans une conférence maintient une liste de fenêtres pour les fenêtres hébergées, reflets et locales masquantes. Comme des multiples ASCE peuvent rendre compte de mises à jours de listes de fenêtres aux autres ASCE en utilisant le *WindowListPDU*, il existe toujours la possibilité d'échanges intempestifs. La création et la destruction de fenêtres ainsi que la modification de leur taille et position peut uniquement arriver sur l'application hôte du terminal possédant la fenêtre concernée. Par contraste, la position d'une fenêtre partagée dans l'ordre en Z peut être altérée sur n'importe quel terminal.

Si les listes de fenêtres *WindowListPDU* étaient simplement appliquées par les ASCE réceptrices, ceci aurait pour effet la mise en harmonie des taille, position et existence de fenêtres (comme la mise

à jour dans les deux listes rentrant en collision serait orthogonale). Cependant, une application simpliste des changements de l'ordre en Z peut aboutir à un croisement infini de listes de fenêtres. Ainsi donc, chaque ASCE maintient un identificateur de liste de fenêtres qu'elle place dans chaque *WindowListPDU* pour la résolution d'échanges intempestifs dans l'ordre en Z. L'identificateur comporte trois parties:

- numéro de séquence: c'est la valeur courante du numéro de séquence tel qu'il a été calculé par l'ASCE émettrice (voir ci-dessous); il est consigné dans les bits 4-15 (à savoir les trois quartets les plus significatifs) du paramètre *listID*;
- incrément: c'est le montant absolu utilisé par l'ASCE pour incrémenter le dernier numéro de séquence vu afin de générer le nouveau numéro de séquence (voir ci-dessous); il est consigné dans les bits 0-3 (à savoir le quartet le moins significatif) du paramètre *listID;*
- top d'horloge: c'est un top d'horloge en millisecondes utilisé pour résoudre les échanges intempestifs lorsque les numéros de séquence sont identiques; il est consigné dans le paramètre *listTime*.

Chaque ASCE calcule la valeur d'identificateur suivant en incrémentant le dernier numéro de séquence vu (à savoir reçu ou envoyé) par la priorité de changement de liste de fenêtres pour créer un nouveau numéro de séquence. Le nouvel identificateur est constitué à partir du nouveau numéro de séquence, de l'incrément (ou priorité) et du temps local courant en millisecondes. Les valeurs de priorité/incrément recommandées sont les suivantes:

- 0 $\Rightarrow$ pas de changement dans l'ordre en Z mais changements dans les positions et/ou tailles de fenêtres;
- 2 $\Rightarrow$ changement dans l'ordre en Z;
- 3 $\Rightarrow$ changement dans l'ordre en Z et changement dans l'activation de fenêtre;
- 4 $\Rightarrow$ changement dans l'ordre en Z et changement dans l'activation de fenêtre pour fenêtre spéciale (voir 8.10.2);
- 5 $\Rightarrow$ l'ASCE ne présente plus.

Un incrément de 5 ne devrait être utilisé que lorsqu'une ASCE arrête d'héberger des applications pour forcer une disparition dans le temps de toutes les fenêtres reflets correspondantes.

Les mises à jour de liste de fenêtres avec un identificateur d'incrément à zéro sont toujours appliquées comme elles n'affectent pas l'ordre en Z – et ne sont donc pas sujettes à la résolution d'échanges intempestifs. Cependant, les identificateurs de liste de fenêtres ayant des incréments de 2 à 5 requièrent de la part de l'ASCE réceptrice une vérification des échanges intempestifs d'ordre en Z qui seront résolus comme suit:

- l'application d'une liste de fenêtres reçue peut provoquer la perte de changements déjà effectués (mais pas encore envoyés) dans le système local. Pour minimiser les incohérences temporaires et pour minimiser les "rebonds", une ASCE devrait simuler de telles collisions en testant l'identificateur reçu par rapport à l'identificateur (prévu) qu'il utiliserait dans son prochain *WindowListPDU*. S'il existe des changements de fenêtres en instance et que l'identificateur prévu est plus tardif que le dernier reçu/envoyé, alors l'ordre en Z de la liste de fenêtres reçue n'est pas appliqué et la prochaine liste de fenêtres (contenant les changements en instance) sera envoyée au moment voulu;
- s'il existe des changements de fenêtres en instance et que l'identificateur prévu est présent plus tôt que le dernier reçu/envoyé, alors l'ordre en Z de la liste de fenêtres reçue est appliqué normalement, et l'identificateur reçu devient le nouvel identificateur. Ainsi donc, lorsque la nouvelle liste de fenêtre sera générée, elle sera envoyée avec un identificateur n'indiquant aucune collision;

- s’il existe des changements de fenêtres en instance, l’ASCE réceptrice compare l’identificateur reçu par rapport au dernier vu. Il existe deux cas à considérer:
  - si l’identificateur est reçu plus tard, l’ordre en Z de liste de fenêtres reçue est appliqué et l’ASCE devrait prévoir l’envoi conditionnel d’un *WindowListPDU* pour renvoyer les changements de fenêtre en instance;
  - si l’identificateur est reçu plus tôt, alors l’ordre en Z de la liste de fenêtres reçue est écarté.

Les identificateurs sont comparés en utilisant la règle suivante. L’identificateur A est plus tard que l’identificateur B si:

- numéro de séquence de A > numéro de séquence de B;
- numéro de séquence de A = numéro de séquence de B ET incrément de A > incrément de B;
- numéro de séquence de A = numéro de séquence de B ET incrément de A = incrément de B ET top d’horloge de A > top d’horloge de B.

Le test final du cas final (top d’horloge de A > top d’horloge de B) est utilisé seulement pour la résolution d'échanges intempestifs – ceci n'implique pas d’ordonnancement particulier de temps global.

8.10.2 Considérations d’implémentation

Sur certains terminaux, des boîtes d’erreur de gestionnaire de fenêtre critiques (apparaissant comme des fenêtres) sont en fait tracées directement sur la fenêtre du moniteur. Une ASCE devrait manipuler ce scénario en créant une fenêtre virtuelle plein écran partagée en tête de l’ordre en Z. Lorsque des ASCE homologues traceront cette fenêtre virtuelle, elle apparaîtra transparente aux zones ne contenant pas la boîte d’erreur, puisque les mises à jour seront reçues uniquement pour la zone de boîte d’erreur réelle (en fait une majeure partie de la fenêtre plein écran n’est pas dessinée). Lorsque la boîte d’erreur est refermée, la fenêtre virtuelle est détruite et on observe un retour à la normale.

Certains terminaux supportent des sessions d’écran multiples, là où l’une ou plusieurs des sessions d’écrans peuvent être des sessions utilisant du texte plein écran pouvant ne pas être accessible à l’ASCE. Lorsqu’une telle session de texte plein écran a le contrôle de l’écran, l’ASCE devrait ajouter une fenêtre plein écran locale virtuelle dans son prochain *WindowListPDU*. Ceci sera interprété par les ASCE homologues comme une fenêtre masquant totalement les fenêtres hébergées, avec pour conséquence que toutes les fenêtres reflets sur les ASCE réceptrices seront à l’intérieur de cette zone masquée.

8.11 Activation de fenêtre

De nombreux terminaux et/ou gestionnaires de fenêtres de terminal supportent le concept d’activation de fenêtre, dans lequel une fenêtre locale particulière est considérée comme la fenêtre active – souvent appelée la fenêtre focalisée ou la fenêtre focalisée d’entrée. Lorsqu’un tel concept est supporté, alors la fenêtre active reçoit toutes les saisies du clavier et du dispositif de pointage du terminal local.

Le protocole AS fournit un mécanisme d’activation de fenêtre, par lequel les ASCE peuvent indiquer les changements dans les états d’activation de fenêtres locales et demander aux autres ASCE les changements dans l’état d’activation de fenêtres. On demande à une ASCE de mettre en correspondance le modèle du terminal local concernant l’activation de fenêtre avec le modèle du protocole AS décrit dans le présent sous-paragraphe. Là où le terminal local ne supporte pas l’activation de fenêtre, alors l’ASCE peut effectuer un mappage grossier.

Certains terminaux et/ou gestionnaires de fenêtres de terminal permettent à l'application de capturer le dispositif de pointage. Lorsque c'est le cas, la fenêtre d'application qui capture peut ne pas être la fenêtre active (par exemple, la fenêtre d'application qui capture peut se réduire elle-même ou devenir invisible tant que la capture est en cours), mais continuer de recevoir les saisies capturées du dispositif de pointage. Dans le protocole AS, la capture de dispositif de pointage par une fenêtre hébergée est traitée comme un cas spécial (voir ci-dessous).

L'activation de fenêtre AS est étroitement intégrée avec la présidence (si en vigueur dans la conférence) et avec l'état de contrôle d'AS de la conférence. Si une ASCE ne possède pas le droit de fournir des entrées aux fenêtres hébergées et réfléchies alors il ne peut pas changer l'état d'activation d'une fenêtre dans la conférence. Voir 8.19 pour plus d'information sur le mode présidé et 8.12 et 8.13 pour plus d'information sur les mécanismes de contrôle.

Lorsque la fenêtre active change sur un terminal à l'intérieur de la conférence, la nouvelle fenêtre active peut être l'une des suivantes:

- une fenêtre hébergée visible, invisible ou qui capture;
- une fenêtre locale visible ou invisible;
- une fenêtre reflet visible (typiquement les ASCE ne maintiennent pas des fenêtres reflets invisibles).

8.11.1 Indications et requêtes d'activation

Une ASCE supervisera deux classes d'événements d'activation appelés indications et requêtes.

- indications: ce sont des changements d'état ou de capture d'action vers une fenêtre locale ou hébergée. Ils affectent une fenêtre possédée par une application sur un terminal local et nécessitent d'être propagés vers les ASCE homologues. Lorsqu'une ASCE détectera un état de changement d'état de capture ou d'activation affectant une fenêtre locale ou hébergée, alors elle enverra un *WindowActivationPDU* avec l'indication appropriée (voir ci-dessous) à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *WindowActivationPDU* est décrit dans le Tableau 8-46;
- requêtes: ce sont des changements de (ou correspondant à des) fenêtres reflets. Certains gestionnaires de fenêtres de terminal fournissent des séquences d'activation spécifiques au terminal, tel que la séquence de touches ALT-TAB commutant de manière cyclique le focus de fenêtre ou les menus contextuels du gestionnaire de fenêtre pour permettre à un utilisateur de focaliser sur une fenêtre spécifique. Là où la fenêtre affectée est réfléchie, le changement ne devrait pas être effectué localement, mais devrait être redirigé vers l'ASCE possédant la fenêtre hébergée correspondante. Lorsqu'une ASCE reconnaît un tel changement sur une fenêtre reflet, elle enverra un *WindowActivationPDU* avec la demande appropriée (voir ci-dessous) à l'ASCE homologue hébergeant la fenêtre correspondante de la manière indiquée dans le Tableau 6-3.

Les changements dans l'état d'activation de fenêtre des fenêtres hébergées peuvent aboutir à des changements de liste de fenêtres. Par exemple, une requête *Restore Window* aboutira normalement à un changement de position et/ou d'ordre en Z dans la fenêtre hébergée correspondante. Par contraste, une indication *Hosted Window Active* peut aboutir ou non à un changement d'ordre en Z, ceci dépendant de la politique de focalisation du gestionnaire de fenêtres du terminal de l'ASCE hôte.

Les changements d'état d'activation de fenêtre des fenêtres hébergées ne provoquent pas de changement dans le contrôle.

Tableau 8-46/T.128 – WindowActivationPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
action	ce paramètre identifie l'action particulière *WindowActivationPDU*. L'ensemble des actions définies est classé en tant qu'indications ou requêtes. Les actions autorisées sont les suivantes. • *Local Window Active* (indication) • *Hosted Window Active* (indication) • *Hosted Window Invisible* (indication) • *Pointing Device Capture* (indication) • *Activate Window* (requête) • *Close Window* (requête) • *Restore Window* (requête) • *WindowManagerMenu* (requête) • *ActivationHelpKey* (requête) • *ActivationHelpIndexKey* (requête) • *ActivationHelpExtendedKey* (requête) voir ci-dessous pour plus d'information.
activationID	ce paramètre est un identificateur assigné par l'ASCE émettrice. Ce paramètre est seulement valide pour les indications où il est utilisé en association avec les priorités d'indication définies pour résoudre les échanges intempestifs de *WindowActivationPDU*. L'intervalle autorisé de valeurs est 1..65534. Voir 8.11.2 ci-dessous pour plus d'information.
activationWindow	ce paramètre fournit un *windowID* spécifique pour les types de messages spécifiques (voir ci-dessous).
activationPoint	pour ce qui concerne les requêtes *WindowManagerMenu*, c'est un paramètre optionnel pouvant contenir la position du dispositif de pointage ayant provoqué la requête d'activation.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Ce qui suit détaille chaque indication ou requête de *WindowActivationPDU*. Certaines des actions requises imposent qu'une ASCE soit capable de changer l'activation du terminal local pour faire en sorte qu'aucune fenêtre n'ait plus le focus local. Sur certains types de terminaux, ceci peut imposer à une ASCE de maintenir une fenêtre spéciale invisible appelée "aucun focus". Ce qui suit demande (comme au 8.10) qu'une ASCE assigne un identificateur local unique pour chaque fenêtre hébergée (son *hosted_window_ID*).

- l'indication *Local Window Active* indique que l'activation s'est déplacée sur une fenêtre locale de l'ASCE émettrice. Sur réception d'un *Local Window Active*, une ASCE changera l'activation locale à "aucun focus";

- l'indication *Hosted Window Active* indique que l'activation s'est déplacée sur une fenêtre hébergée de l'ASCE émettrice. Le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée devenue active. Sur réception d'une indication *Hosted Window Active*, une ASCE déplacera l'activation locale sur la fenêtre locale reflet correspondant à la fenêtre hébergée indiquée;
- l'indication *Hosted Window Invisible* indique que l'activation s'est changée en fenêtre hébergée invisible sur l'ASCE émettrice. Sur réception d'une indication *Hosted Window Invisible*, une ASCE changera l'activation locale en *absence de focus*;
- l'indication *Pointing Device Capture* indique que l'activation s'est portée sur la fenêtre hébergée ayant capturé le dispositif de pointage sur l'ASCE émettrice. Là où la fenêtre hébergée qui capture est visible, le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée ayant capturé le dispositif de pointage. Là où la fenêtre hébergée qui capture est invisible, le paramètre *WindowActivationPDU activationWindow* vaut zéro. Sur réception d'une indication *Pointing Device Capture* avec un *hosted_Window_ID*, une ASCE portera l'activation locale sur la fenêtre reflet locale correspondant à la fenêtre hébergée indiquée. Sur réception d'un *Pointing Device Capture* sans *hosted_Window_ID*, une ASCE changera l'activation locale en *absence de focus*;
- la requête *Activate Window* indique que l'ASCE émettrice a détecté une requête locale d'activation de fenêtre reflet. Le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée sur l'ASCE homologue correspondant à la fenêtre reflet de l'ASCE émettrice. Sur réception d'une requête *Activate Window*, une ASCE activera la fenêtre hébergée indiquée. Ceci devrait faire en sorte que l'ASCE réceptrice envoie ultérieurement une indication *WindowActivationPDU Hosted Window Active* à la fenêtre hébergée;
- la requête *Close Window* indique que l'ASCE émettrice a détecté une requête locale de fermeture de fenêtre reflet. Le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée sur l'ASCE homologue correspondant à la fenêtre reflet de l'ASCE émettrice. Sur réception d'une requête *Close Window*, une ASCE fermera la fenêtre hébergée indiquée. Ceci devrait faire en sorte que l'ASCE réceptrice envoie une indication *WindowActivationPDU* à la fenêtre locale héritant de l'activation;
- la requête *Restore Window* indique que l'ASCE émettrice a détecté une requête locale de restauration de fenêtre reflet. Le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée sur l'ASCE homologue correspondant à la fenêtre reflet de l'ASCE émettrice. Sur réception d'une requête *Restore Window*, une ASCE restaurera la fenêtre hébergée indiquée. Ceci devrait faire en sorte que l'ASCE réceptrice envoie une indication *WindowActivationPDU Hosted Window Active* à la fenêtre hébergée;
- la requête *WindowManagerMenu* indique que l'ASCE émettrice a détecté une opération de type menu du gestionnaire de fenêtre locale correspondant à une fenêtre hébergée réduite. Le paramètre *WindowActivationPDU activationWindow* contient le *hosted_window_ID* de la fenêtre hébergée sur l'ASCE homologue correspondant à la fenêtre reflet de l'ASCE émettrice et le paramètre *activationPoint* contient de manière optionnelle les coordonnées x et y du moniteur virtuel de l'événement d'activation (là où il a été initialisé par le dispositif de pointage). Sur réception d'une requête *WindowManagerMenu*, une ASCE soumettra une requête de menu localement équivalente pour la fenêtre hébergée indiquée;
- les requêtes *ActivationHelpKey*, *ActivationHelpIndexKey* et *ActivationHelpExtendedKey* indiquent que l'ASCE émettrice a détecté pour une fenêtre reflet la requête correspondante d'activation d'aide du gestionnaire de fenêtre local. Le paramètre *WindowActivationPDU*

activationWindow contient le *hosted_window_ID* de la fenêtre hébergée sur l'ASCE homologue correspondant à la fenêtre reflet de l'ASCE émettrice. Sur réception d'une requête *ActivationHelpKey, ActivationHelpIndexKey* ou *ActivationHelpExtendedKey*, une ASCE soumettra une requête de menu localement équivalente pour la fenêtre hébergée indiquée.

Une ASCE peut être capable de détecter seulement certaines requêtes d'activation sur certains terminaux et/ou peut être capable de traiter seulement certaines requêtes d'activation sur certains terminaux. Par exemple, alors que l'activation de gestionnaire de fenêtre du terminal est typiquement spécifique du type de terminal et/ou du type de gestionnaire de fenêtre, le concept sous-jacent est encore suffisamment générique pour qu'une ASCE détecte localement l'activation (d'une manière spécifique au gestionnaire de fenêtre du terminal A) et pour qu'une ASCE homologue le traite (d'une manière spécifique au gestionnaire de fenêtre du terminal B). Par contraste, les fonctions d'activation d'aide (commutant vers une classe spécifique de fenêtres d'aide) sont moins largement implémentées et peuvent être disponibles seulement lorsque les deux ASCE (à savoir l'ASCE ayant la fenêtre reflet et l'ASCE homologue ayant la fenêtre hébergée correspondante) s'exécutent sur le même type de terminal.

- une ASCE enverra un *WindowActivationPDU* avec l'action *WindowManagerMenu* seulement là où la capacité *Window Activation.windowManagerMenuFlag* de l'ASCE homologue est à TRUE;
- une ASCE enverra un *WindowActivationPDU* avec l'action *ActivationHelpKey* seulement là où la capacité *Window Activation.helpKeyFlag* de l'ASCE homologue est à TRUE;
- une ASCE enverra un *WindowActivationPDU* avec l'action *ActivationHelpIndexKey* seulement là où la capacité *Window Activation.helpExtendedKeyFlag* de l'ASCE homologue est à TRUE;
- une ASCE enverra un *WindowActivationPDU* avec l'action *ActivationHelpExtendedKey* seulement là où la capacité *Window Activation.helpExtendedKeyFlag* de l'ASCE homologue est à TRUE.

8.11.2 Priorités et identificateurs d'activation

La nature asynchrone du protocole d'activation d'AS et la multiplicité des raisons de changements d'état d'activation de fenêtre signifient que peuvent apparaître des collisions de *WindowActivationPDU*s. Pour les détecter, et ainsi maintenir un état d'activation cohérent parmi toutes les ASCE actives dans la conférence, une ASCE placera un identificateur d'activation dans toutes les indications *WindowActivationPDU*. On ne demande pas d'identificateur pour les requêtes *WindowActivationPDU*.

Dans une conférence, chaque ASCE conserve le dernier identificateur reçu ou envoyé dans une indication *WindowActivationPDU* et incrémente l'identificateur (dans l'intervalle 1..65534 avec remise à zéro) lors de l'envoi de chaque indication *WindowActivationPDU*.

- si une indication reçue possède un identificateur inférieur au dernier reçu ou envoyé, il est ignoré;
- si une indication reçue possède un identificateur égal au dernier reçu ou envoyé, alors la priorité d'indication reçue (voir ci-dessous) est comparée à la priorité de la dernière indication reçue ou envoyée. Si la priorité de l'indication reçue est inférieure ou égale à celle de la dernière indication reçue ou envoyée, alors l'indication reçue est écartée. Dans le cas contraire, celle-ci est appliquée;
- si une indication reçue possède un identificateur supérieur au dernier reçu ou envoyé, celle-ci est appliqué.

Le Tableau 8-47 décrit les priorités définies pour les indications *WindowActivationPDU*.

Les identificateurs ne sont pas utilisés pour les requêtes *WindowActivationPDU*. Ceci assure une application des requêtes indépendamment de l'identificateur d'indication et de la manipulation de priorité. Par exemple, là où une ASCE envoie une requête *Activate Window* très rapidement suivie d'une requête *Restore Window*, alors la seconde requête est encore appliquée quand bien même elle pourrait plus tard engendrer une indication *Hosted Window Active* de la part de l'ASCE réceptrice.

Tableau 8-47/T.128 – Priorités d'indication WindowActivationPDU

Indication	Priorité
Local Window Active	1
Hosted Window Active	1
Hosted Window Invisible	1
Pointing Device Capture	2

8.12 Contrôle

La politique de contrôle de conférence est un déterminant majeur dans le caractère utilisable du partage d'application. L'expérience montre que des politiques de contrôle sont applicables pour des mélanges différents de taille de conférence et/ou d'expériences d'utilisateur. Ainsi donc, le protocole AS n'impose aucune politique de contrôle particulière, mais fournit plutôt un ensemble de mécanismes de contrôle grâce auxquels les ASCE peuvent implémenter une variété de politiques ayant (potentiellement) différentes caractéristiques – soit séquentielles ou concurrentes à l'intérieur de la conférence. Le protocole AS définit aussi un jeu de médiations de mécanismes de contrôle, qui est construit au-dessus des mécanismes de contrôle de base décrits dans le présent sous-paragraphe – Voir 8.13 pour plus d'information.

Le protocole de contrôle d'AS de base est fondé sur la gestion du droit de fournir des entrées aux fenêtres hébergées et/ou reflets. En les combinant ces droits supportent les modes contrôle de base suivants:

- détaché: dans ce mode, une ASCE:
  - possède le droit de fournir des entrées aux fenêtres hébergées;
  - ne possède pas le droit de fournir des entrées aux fenêtres reflets;
  - refuse aux ASCE homologues le droit de fournir des entrées aux fenêtres reflets correspondant aux fenêtres hébergées sur cette ASCE.

  En pratique, ceci permet à l'utilisateur de travailler avec les applications hébergées sans aucune interférence de la part des autres utilisateurs – les autres utilisateurs ne peuvent pas fournir d'entrée, de changements d'activation et de changements d'ordre en Z;
- coopérant: dans ce mode, les ASCE coopérantes à l'intérieur de la conférence acquièrent en série le droit de fournir des entrées aux fenêtres hébergées et reflets. A n'importe quel instant de la conférence:
  - l'une des ASCE coopérante peut fournir des entrées aux fenêtres hébergées et reflets – mais seulement là où les autres ASCE ne sont pas de type *Detached* (sont de type *In Control*);
  - les autres ASCE coopérantes ne peuvent pas fournir d'entrées aux fenêtres hébergées et reflets (ils sont de type *viewing*).

Là où une ASCE n'a pas le droit de fournir des entrées aux fenêtres reflets – elle est détachée ou coopérante en visualisation – elle peut encore fournir de l'information sur le mouvement du dispositif de pointage aux autres ASCE. Là où une ASCE est dans l'un de ces états de contrôle et fournit aussi cette information, alors les autres ASCE peuvent utiliser l'information pour fournir un retour utilisateur sur l'activité du dispositif de pointage de l'ASCE émettrice – ce qui peut (pour certains types de terminaux) améliorer de manière significative la perception de l'utilisateur distant du caractère utilisable du partage d'application. Le fait de fournir de l'information sur le dispositif de pointage lorsque l'ASCE est dans ces états et comment la présenter lors de la réception est une décision d'implémentation d'ASCE. Voir 8.18 pour plus d'information sur les saisies et les événements de dispositif de pointage.

Le protocole de base de contrôle d'AS ne spécifie pas les droits du terminal local ou de l'ASCE pour fournir des entrées aux fenêtres locales lorsqu'elles sont en modes détachés ou coopérantes. La politique particulière adoptée pour les droits locaux d'entrée sera normalement déterminée par les caractéristiques particulières du terminal local.

Une conférence peut contenir n'importe quel mélange d'ASCE détachées ou coopérantes. Les ASCE peuvent librement se déplacer entre les deux modes. Par contraste, une ASCE peut seulement être en mode contrôle tant qu'elle possède l'identificateur de contrôle. Chaque ASCE (qu'elle soit coopérante ou détachée) piste la valeur de l'identificateur de contrôle courant dans la conférence et l'ASCE détenant l'identificateur.

Quand une ASCE désire changer l'état de contrôle de la conférence, elle enverra un *ControlPDU* à toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *ControlPDU* est décrit dans le Tableau 8-48.

Tableau 8-48/T.128 – ControlPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
action	ce paramètre identifie l'action *ControlPDU* particulière. Les actions autorisées sont les suivantes. • *Request Control* • *Grant Control* • *Detach* • *Cooperate* voir ci-dessous pour plus d'information.
grantID	lorsque le paramètre d'action (voir ci-dessus) est *Grant Control*, ce paramètre spécifie le *MCS User ID* de l'ASCE à laquelle on a accordé le contrôle – à savoir la nouvelle détentrice de l'identificateur de contrôle.
controlID	lorsque le paramètre d'action (voir ci-dessus) est *Grant Control*, ce paramètre est l'identificateur de contrôle assigné par l'ASCE émettrice (voir ci-dessous).
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Lorsqu'une ASCE souhaite obtenir l'identificateur de contrôle, elle enverra un *Request Control ControlPDU* aux autres ASCE. Cette ASPDU est envoyée à toutes les autres ASCE (ainsi le sont tous les *ControlPDUs*) de sorte que, même si les ASCE pistent l'ASCE détenant l'identificateur de contrôle, elles puissent détecter des situations où l'identificateur de contrôle est en cours de déplacement ou bien où les ASCE détenant l'identificateur de contrôle deviennent inactives ou quittent la conférence.

Sur réception d'un *Request Control ControlPDU*, l'ASCE détenant l'identificateur de contrôle devra normalement (mais voir ci-dessous) envoyer un *Grant Control ControlPDU* avec la valeur de l'identificateur de contrôle courant et le *MCS User ID* de l'ASCE auquel le contrôle est en train d'être accordé par toutes les autres ASCE. Cette ASPDU est envoyée à toutes les ASCE de sorte que toutes les ASCE actives puissent pister le possesseur actuel de l'identificateur de contrôle. Là où une ASCE reçoit un *Request Control ControlPDU* quand elle ne détient pas l'identificateur de contrôle, elle écarte cette ASPDU.

Il existe des situations où le fait d'accorder un contrôle de manière inconditionnelle peut ne pas être faisable. Par exemple, sur certains types de terminaux, certaines fonctions de gestionnaire de fenêtre (tel que traîner une fenêtre et/ou redimensionner une fenêtre locale) ont besoin d'être terminées de manière explicite. Mais si l'ASCE est en mode coopérant et perd le contrôle (et donc perd le droit de fournir toute entrée), celle-ci ne peut pas achever l'opération du gestionnaire de fenêtre, qui laisse l'opération active et l'ASCE homologue incapable de la terminer (parce qu'elle ne peut pas fournir d'entrée à une fenêtre locale sur cette ASCE). Là où une telle situation s'applique, une ASCE détenant l'identificateur de contrôle peut répondre à un *Request Control ControlPDU* avec un *Grant Control ControlPDU* en renvoyant la valeur de l'identificateur de contrôle courant et son propre *MCS User ID* (c'est-à-dire qu'elle s'accorde le contrôle à elle-même). Comme ceci est envoyé à toutes les ASCE, les ASCE homologues, y compris le demandeur, la traitent comme un échange ordinaire de contrôle, et l'ASCE locale retient le contrôle jusqu'à l'achèvement de l'opération posant problème.

Là où une ASCE se déplace du mode coopérant vers le mode détaché, elle enverra un *Detach ControlPDU* à toutes les ASCE. D'une manière identique, là où une ASCE se déplace du mode détaché vers le mode coopérant, elle enverra un *Cooperate ControlPDU* à toutes les ASCE. Le fait de changer du mode coopérant vers le mode détaché et vice versa est indépendant de la propriété de l'identificateur de contrôle, et une ASCE détenant l'identificateur de contrôle peut se déplacer du mode coopérant dans le mode détaché et vice versa sans accorder l'identificateur de contrôle si aucune ASCE homologue la réclame dans l'intérim.

8.12.1 Identificateurs de contrôle

L'identificateur de contrôle est une valeur simple visible par toutes les ASCE dans la conférence, dans l'intervalle 0..0x80000000.

La valeur initiale de l'identificateur de contrôle dans une conférence est toujours zéro. Le détenteur initial de l'identificateur de contrôle est l'ASCE dont l'identificateur de partage est le plus élevé au moment de l'activation de l'ASCE (Voir 8.4).

Ensuite, la valeur de l'identificateur de contrôle change lorsqu'une ASCE ne reçoit pas la réponse *GrantControl* à un *Request Control ControlPDU* (dans un délai raisonnable) ou lorsque le détenteur de l'identificateur de contrôle devient inactif ou quitte la conférence. Lorsque c'est le cas, chaque ASCE détectrice génère un nouvel identificateur de contrôle en incrémentant la dernière valeur connue de l'identificateur de contrôle de sa valeur *MCS User ID* et envoie un *Grant Control ControlPDU* le référençant elle-même comme l'ASCE possesseur (à savoir, elle s'annonce elle-même comme la nouvelle détentrice de l'identificateur de contrôle). Comme de multiples ASCE peuvent avoir détecté le problème, ceci provoque un échange intempestif d'identificateurs de

contrôle qui est gagné par l'ASCE ayant la valeur d'identificateur de contrôle la plus élevée. Ceci crée la contrainte qu'à tout instant, si une ASCE reçoit un *Grant Control ControlPDU* avec une valeur d'identificateur de contrôle plus élevée que la dernière valeur d'identificateur de contrôle connue pour cette ASCE, elle devra reconnaître la valeur plus élevée comme la nouvelle valeur d'identificateur de contrôle et la nouvelle ASCE comme la nouvelle détentrice de l'identificateur de contrôle.

Lorsqu'une ASCE détecte qu'une nouvelle ASCE est devenue active, elle annoncera son état de contrôle comme suit:

- si l'ASCE est détachée, elle enverra un *Detach ControlPDU*;
- si l'ASCE est coopérante, elle enverra un *Cooperate ControlPDU*;
- si l'ASCE détient l'identificateur de contrôle, elle enverra un *Grant Control ControlPDU* se référençant elle-même comme la nouvelle détentrice de l'identificateur de contrôle.

Le bénéfice de ce qui est décrit ci-dessus est que les nouveaux arrivants (et les ASCE existantes) reçoivent un rafraîchissement de l'état de contrôle de chaque ASCE et en plus de l'information sur le détenteur de l'identificateur de contrôle. Voir 8.4 pour plus d'information sur l'activation d'ASCE et 8.6 pour plus d'information sur la synchronisation.

8.12.2 Interaction en mode présidé

Le fonctionnement du mode présidé (voir 8.19) interagit avec le protocole de contrôle d'AS comme suit.

Lorsqu'une conférence entrera en mode dirigé, toutes les ASCE enverront un *Cooperate ControlPDU*, l'ASCE sur le nœud président enverra un *Request Control ControlPDU* et l'ASCE détenant l'identificateur de contrôle répondra avec un *Grant Control ControlPDU*. A savoir, toutes les ASCE entrant dans le mode coopérant et le nœud président acquièrent le contrôle – toutes les autres ASCE sont visualisantes.

Lorsque la Présidence se déplace d'un nœud à un autre, le nouveau nœud président enverra un *Request Control ControlPDU* et l'ASCE détenant l'identificateur de contrôle (à savoir, le nœud président précédent) répondra avec un *Grant Control ControlPDU*. A savoir le contrôle suit la Présidence.

Lorsque la conférence sort du mode présidé, toutes les ASCE restent en mode coopérant et le dernier nœud président conserve l'identificateur de contrôle, mais les ASCE sont encore une fois libres de demander le contrôle et de commuter entre les modes détaché et coopérant. A savoir, le protocole complet de contrôle d'AS est réinstallé.

8.13 Contrôle médiatisé

Le protocole AS décrit au 8.12 fournit un ensemble raisonnable de facilités de contrôle de base. Mais il ne fournit pas des facilités comme le passage explicite de contrôle à une ASCE spécifique ou la possibilité pour les ASCE de refuser des requêtes de contrôle de manière conditionnelle ou inconditionnelle.

Le protocole de contrôle médiatisé d'AS est construit sur le protocole de contrôle de base pour fournir des facilités de contrôle supplémentaires plus conditionnelles. Le protocole de contrôle médiatisé est négociable et est supporté seulement là où les capacités négociées le permettent (à savoir dans le mode hérité où l'indicateur binaire négocié *Control.controlFlags capability Allow Mediated Control* est positionné; dans le mode de base, là où la capacité négociée *Control.mediatedControlFlag* vaut TRUE). Voir 8.2.10 pour plus d'information sur les jeux de capacités de contrôle.

Le protocole de contrôle médiatisé est implémenté sous la forme d'un jeu de messages de requêtes et de réponses, qui sont construits au-dessus et médiatisent l'effet du protocole de contrôle de base. Là où l'explication d'une facilité ou d'un échange de message de protocole de contrôle médiatisé dans le présent paragraphe requiert de la part d'une ASCE l'initialisation d'une action et/ou du changement d'état du protocole de contrôle de base, alors celle-ci est décrite comme suit:

- *Core*(*Request Control*): l'ASCE envoie un *Request Control ControlPDU* pour prendre le contrôle;
- *Core*(*Detach*): l'ASCE envoie un *Detach ControlPDU* pour notifier aux ASCE homologues qu'elle est entrée dans le mode détaché.

Lorsqu'une ASCE désire envoyer une requête ou une réponse de contrôle médiatisé, elle enverra un *MediatedControlPDU* à l'une ou toutes les ASCE dans la conférence de la manière indiquée dans le Tableau 6-3. Le contenu de *MediatedControlPDU* est décrit dans le Tableau 8-49.

Tableau 8-49/T.128 – MediatedControlPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
action	ce paramètre identifie l'action *MediatedControlPDU* particulière. Les actions autorisées sont les suivantes. • *Take Control Request* • *Pass Control Request* • *Detach Request* • *Confirm Take Response* • *Deny Take Response* • *Confirm Detach Response* • *Deny Detach Response* • *Deny Pass Response* • *Remote Detach Request* • *Deny Remote Detach Response* voir ci-dessous pour plus d'information.
passControlFlag	ce paramètre indique si ce M*ediatedControlPDU* fait partie de la séquence *Pass Control* (voir ci-dessous). Là où ce paramètre en fait partie, sa valeur est TRUE. Sur tous les autres *MediatedControlPDUs*, ce paramètre sera positionné à FALSE.
sendingReference	ce paramètre est une référence à message utilisé pour établir une corrélation entre les requêtes et les réponses. Là où ce *MediatedControlPDU* est une requête (voir le paramètre d'action ci-dessus), c'est la référence allouée par l'ASCE émettrice. Là où ce *MediatedControlPDU* est une réponse, c'est la référence de la requête correspondante.

Tableau 8-49/T.128 – MediatedControlPDU *(fin)*

Paramètre	Description
originatorReference	ce paramètre est une référence à message utilisée pour établir une corrélation entre les requêtes et les réponses. Là où ce *MediatedControlPDU* est une requête de prise de commande provenant de la requête *Pass Control* (voir ci-dessous), ce paramètre est la référence de la requête *Pass Control* d'origine. Là où ce *MediatedControlPDU* est une réponse, c'est la référence de la requête correspondante.
originatorID	ce paramètre est un *MCS User ID*. Là où ce *MediatedControlPDU* est une requête (voir le paramètre d'action ci-dessus), c'est le *MCS User ID* de l'ASCE émettrice. Là où ce *MediatedControlPDU* est une réponse, c'est le *MCS User ID* de la requête correspondante.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.13.1 Prise de contrôle

Lorsqu'une ASCE désire prendre le contrôle en utilisant le protocole de contrôle médiatisé, ses actions dépendent de la valeur de la capacité négociée *Control.controlInterest*. Là où la capacité négociée vaut *Always*, la prise de contrôle est non médiatisée et l'ASCE initialise l'action *Core* (*Control request*) pour prendre le contrôle (voir 8.2.10 pour plus d'information sur le jeu de capacités de contrôle). Là où la valeur négociée est *Never*, une ou plusieurs ASCE homologues ne permettent pas la prise de contrôle et l'ASCE ne peut pas la réaliser. Là où la valeur négociée est *Confirm*, une ou plusieurs ASCE demandent que la prise de contrôle soit confirmée par les ASCE homologues et que l'ASCE envoie un *Take Control Request MediatedControlPDU* à toutes les ASCE.

Sur réception d'un *Take Control Request MediatedControlPDU*, la réponse d'une ASCE dépend de sa valeur de capacité locale *Control.controlInterest* (qui ne devrait jamais être *Never* – car la requête n'aurait jamais dû être envoyée). Là où la valeur locale est *Always*, l'ASCE répond avec un *Confirm Take Response MediatedControlPDU* à l'ASCE demandeuse. Là où la valeur locale est *Confirm*, l'ASCE utilise un mécanisme purement local (tel qu'une interaction avec un utilisateur local) pour déterminer si elle autorise l'ASCE demandeuse à prendre le contrôle et répond ensuite selon la suite donnée avec un *Confirm Take Response* ou un *Deny Take Response MediatedControlPDU* à l'ASCE demandeuse.

Sur réception d'un *Confirm Take Response MediatedControlPDUs* de toutes les autres ASCE homologues (à savoir unanimité), l'ASCE demandeuse initie l'action *Core* (*Request Control*) pour prendre le contrôle. Cependant si elle reçoit un ou plusieurs *Deny Take Response MediatedControlPDUs*, alors elle abandonne sa tentative de prise de contrôle.

8.13.2 Passation de contrôle

Lorsqu'une ASCE désire passer le contrôle à une ASCE homologue en utilisant le protocole de contrôle médiatisé, ses actions dépendent de sa valeur de capacité locale *Control.controlInterest*. Si la valeur locale est *Never* (à savoir cette ASCE n'abandonne jamais le contrôle), elle ne devrait pas essayer de passer le contrôle. Là où sa valeur locale est *Confirm* ou *Always*, elle envoie un *Pass Control Request MediatedControlPDU* à l'ASCE particulière.

Sur réception d'un *Pass Control Request MediatedControlPDU*, l'ASCE utilise un mécanisme purement local (tel qu'une interaction avec un utilisateur local) pour déterminer si elle autorise

l'ASCE demandeuse à prendre le contrôle. Si elle s'avère ne pas accepter la requête de passation, elle répond avec un *Deny Pass Response MediatedControlPDU* à l'ASCE demandeuse. Si elle s'avère accepter la requête de passation, elle envoie un *Take Control Request* à l'ASCE demandeuse qui devrait en retour aboutir à la réception d'un *Confirm Take Response MediatedControlPDUs* de l'ASCE demandeuse, à la suite duquel elle peut initialiser l'action *Core* (*Request Control*) pour prendre le contrôle.

Il est à noter que *Take Control Request MediatedControlPDU* est ici envoyé à une ASCE homologue unique en réponse à *Pass Control Request MediatedControlPDU*, tandis qu'au 8.13.1 ci-dessus, il est envoyé à toutes les ASCE homologues pour prendre le contrôle de manière conditionnelle. Pour faire la distinction entre les deux cas, tous les *MediatedControlPDUs* utilisés dans les parties de séquence de contrôle de passation (à savoir *Pass Control Request*, *Take Control Request* vers une seule ASCE et *Deny Pass Response*) ont leur paramètre *passControlFlag* mis à TRUE – il est mis à FALSE dans tous les autres *MediatedControlPDUs*.

8.13.3 Détachement

Lorsqu'une ASCE désire se détacher en utilisant le protocole de contrôle médiatisé, ses actions dépendent de la valeur de capacité *negociée Control.detachInterest*. Là où la valeur négociée vaut *Always*, le détachement est non médiatisé et l'ASCE initie l'action *Core* (*Detach*) pour se détacher. Là où la valeur négociée vaut *Never*, une ou plusieurs ASCE homologues ne permettront pas aux ASCE de se détacher et l'ASCE ne pourra pas le faire. Là où la valeur négociée vaut *Confirm*, une ou plusieurs ASCE homologues demandent confirmation du détachement et l'ASCE envoie alors un *Detach Request MediatedControlPDU* à toutes les autres ASCE.

Sur réception d'un *Detach Request MediatedControlPDU*, la réponse d'une ASCE dépend de sa valeur de capacité locale *Control.detachInterest* (qui ne devrait jamais être *Never* – car la requête n'aurait jamais dû être envoyée). Là où la valeur locale est *Always*, l'ASCE répond avec un *Confirm Detach Response MediatedControlPDU* à l'ASCE demandeuse. Là où la valeur locale est *Confirm*, l'ASCE utilise un mécanisme purement local (tel qu'une interaction avec un utilisateur local) pour déterminer si elle autorise l'ASCE demandeuse à prendre le contrôle et répond ensuite selon la suite donnée avec un *Confirm Detach Response* ou un *Deny Detach Response MediatedControlPDU* à l'ASCE demandeuse.

Sur réception d'un *Confirm Detach Response MediatedControlPDUs* de toutes les autres ASCE homologues (à savoir unanimité), l'ASCE demandeuse initie l'action *Core* (*Detach*) pour se détacher. Cependant si elle reçoit un ou plusieurs *Deny Detach Response MediatedControlPDUs* alors elle abandonne sa tentative de détachement.

8.13.4 Détachement distant

Là où une ASCE désire détacher une ASCE homologue, ses actions dépendent de la valeur de capacité *Control.remoteDetachFlag* de l'ASCE homologue. Là où la valeur vaut FALSE, l'ASCE homologue ne permet pas le détachement distant et l'ASCE abandonne sa tentative. Là où la valeur vaut TRUE, l'ASCE envoie *Remote Detach Request MediatedControlPDU* à l'ASCE particulière.

Sur réception d'un *Remote Detach Request MediatedControlPDU*, une ASCE tente de démarrer le processus de détachement décrit au 8.13.3. Si elle ne peut pas initier la tentative de détachement (parce que une ou plusieurs ASCE n'autoriseront pas le détachement aux ASCE) elle répond avec un *Deny Remote Detach Response MediatedControlPDU* à l'ASCE demandeuse. Si elle peut initier la tentative de détachement, alors elle procède comme indiqué au 8.13.3.

Les *MediatedControlPDUs* sont envoyés soit à toutes les ASCE homologues ou à des ASCE homologues spécifiques. Ceci dépend du type de message. Le Tableau 8-50 donne un résumé des caractéristiques d'envoi des réponses et des requêtes *MediatedControlPDU*.

Tableau 8-50/T.128 – Canaux MCS MediatedControlPDU

Requête/Réponse	Cible
Take Control Request (Note)	toutes les ASCE
Take Control Request (Note)	ASCE homologue
Pass Control Request	ASCE homologue
Detach Request	toutes les ASCE
Confirm Take Response	ASCE homologue
Deny Take Response	ASCE homologue
Confirm Detach Response	ASCE homologue
Deny Detach Response	ASCE homologue
Deny Pass Response	ASCE homologue
Remote Detach Request	ASCE homologue
Deny Remote Detach Response	ASCE homologue
NOTE – Les deux variantes *Take Control Request* sont utilisées respectivement dans les contrôles de prise et de passage (voir 8.13.1 et 8.13.2 ci-dessus).	

8.14 Curseurs

Lorsque la forme du curseur local changera, ou lorsqu'une application changera par programme la position du curseur local, une ASCE enverra une information sur la forme de curseur local et/ou sa position à toutes les ASCE dans la conférence en envoyant des *PointerPDUs* de la manière indiquée dans le Tableau 6-3.

Les *PointerPDUs* fournissant une nouvelle forme de curseur (par opposition à la mise à jour de position de curseur ou le référencement à un curseur précédemment mis en antémémoire) sont de l'un des trois types suivants:

- une valeur de curseur système prédéfini qui est soit le curseur *null* ou le curseur par défaut. Le curseur *null* devrait être envoyé lorsque le curseur local n'est pas affiché. Le curseur par défaut devrait être envoyé lorsque le curseur local n'est pas au-dessus d'une fenêtre hébergée ou n'est pas capturé par une fenêtre hébergée;
- une définition de curseur monochrome. Ce curseur monochrome peut être utilisé lorsqu'un curseur par défaut n'est pas adapté (voir ci-dessus), un curseur local est visible et un curseur de couleur a été désactivé pendant la négociation des capacités;
- une définition de curseur de couleur. Le curseur de couleur peut être utilisé lorsqu'un curseur par défaut n'est pas adapté (voir ci-dessus), un curseur local est visible et un curseur de couleur a été désactivé pendant la négociation des capacités.

La manière dont une ASCE affiche l'information de curseur reçue dépend de l'état de contrôle courant et de la présidence (voir 8.12, 8.13 et 8.19), de la position courante de curseur et des capacités d'affichage du terminal local. Par exemple, une ASCE réceptrice pourrait:

- afficher la forme de curseur local en mode coopérant et contrôlé;
- afficher le curseur correspondant à l'ASCE hôte en mode coopérant et visualisant;
- afficher une version modifiée d'un ou plusieurs curseurs d'ASCE hôtes en mode détaché avec le ou les curseurs particuliers au-dessus des fenêtres reflets correspondantes.

8.14.1 Curseurs système

Lorsque le curseur local deviendra invisible (à savoir il n'est pas affiché sur le terminal local), une ASCE enverra un *PointerPDU (System)* avec un *systemPointerType* de *pointerNull* à toutes les autres ASCE dans la conférence. Sur réception d'un *PointerPDU (System)* avec un *systemPointerType* de *pointerNull*, une ASCE positionnera son curseur de couleur courant pour cet hôte sur le curseur spécial *null*.

Lorsque le curseur local n'est pas au-dessus d'une fenêtre hébergée ou n'étant pas capturée par une fenêtre hébergée, une ASCE enverra un *PointerPDU (System)* avec un *systemPointerType* de *pointerDefault* à toutes les autres ASCE dans la conférence. Sur réception d'un *PointerPDU (System)* avec un *systemPointerType* de *pointerDefault*, une ASCE positionnera son curseur de couleur courant pour cet hôte sur le curseur spécial par défaut.

Dans les deux cas, on demande à l'ASCE réceptrice de pister le curseur courant de chaque ASCE hôte, mais l'affichage de ce curseur est une initiative purement locale. Voir le Tableau 8-51.

Tableau 8-51/T.128 – PointerPDU (System)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
systemPointerType	ce paramètre identifie le curseur à utiliser. Les valeurs autorisées sont *pointerDefault* ou *pointerNull*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.14.2 Curseurs monochromes

On ne demande pas à une ASCE de supporter les curseurs couleur (voir 8.14.3) et une ASCE peut représenter tous les curseurs non-système comme des curseurs monochromes (même là où ils sont affichés localement en couleur), en supposant qu'elle puisse implémenter une conversion locale adaptée. Cependant, les curseurs couleur sont largement utilisés sur les terminaux normaux et le fait de fournir un curseur seulement monochrome peut aboutir à une baisse significative de l'acceptabilité au niveau de l'utilisateur.

Là où le curseur local change en un curseur représenté, ou pouvant être représenté par un curseur monochrome, une ASCE enverra un *PointerPDU (Mono)* contenant la définition de curseur monochrome du nouveau curseur. On demande à une ASCE réceptrice de se rappeler seulement de la dernière définition de curseur monochrome de chaque ASCE hôte qui devient ensuite le curseur monochrome courant pour cet hôte.

On demande à l'ASCE réceptrice de pister le curseur monochrome courant de chaque ASCE hôte, mais l'affichage de ce curseur reste une initiative purement locale.

Les curseurs monochromes ne sont pas mis en antémémoire. On demande à une ASCE émettrice d'envoyer un nouveau *PointerPDU (Mono)* pour chaque changement de curseur monochrome et à une ASCE réceptrice de se rappeler seulement de la dernière définition de curseur monochrome pour chaque ASCE hôte.

Les données de curseur monochrome représentent un curseur comme un doublet de masques ET et OU exclusif, où le curseur peut être tracé en appliquant la fonction ET au masque ET et la fonction OU exclusif au masque OU exclusif, permettant ainsi de définir le point chaud (voir ci-dessous).

C’est une représentation commune pouvant être facilement mappée de manière lisible sur des fonctions standard locales sur la plupart des terminaux rencontrés.

Les deux masques sont composés d’une série de rangées de pixels monochrones (c'est-à-dire à 1 bit par point), où la rangée zéro commence à la coordonnée Y du point le plus haut, c’est-à-dire la rangée 0 commence en haut à gauche. A l’intérieur d’une rangée, les valeurs des pixels sont empaquetées par octets, en commençant par le point le plus à gauche. Chaque octet contient 8 points, avec le pixel le plus à gauche dans le bit le plus significatif. Chaque rangée du curseur est complétée à droite par remplissage à une frontière de double octet.

Le point de visée du pointeur monochrome définit un point à l’intérieur du curseur correspondant à la position sur l’écran de l’endroit ou le curseur devrait être tracé. Par exemple, si le point de visée du curseur monochrome est au point (3,4) à l’intérieur du curseur et que la position de curseur local est au point (50,50) alors le point le plus haut à gauche (à savoir 0,0) de la définition du curseur est tracé à la position (47,46) et le pixel du point de visée est tracé à la position (50,50). Voir le Tableau 8-52.

Tableau 8-52/T.128 – PointerPDU (Mono)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
hotSpotX	ce paramètre est la coordonnée en X du point de visée de curseur relative au coin haut gauche de la définition de curseur.
hotSpotY	ce paramètre est la coordonnée en Y du point de visée de curseur relative au coin haut gauche de la définition de curseur.
width	ce paramètre est la largeur du curseur en pixels.
height	ce paramètre est la hauteur du curseur en pixels.
monoPointer	c’est une définition de curseur monochrome, composée d’un masque OU exclusif 1 bit/pixel et suivi d’un masque ET 1 bit/pixel.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C’est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.14.3 Curseurs couleur

Le support de curseur couleur est optionnel. Une ASCE peut envoyer des ASPDU *PointerPDU (Color)* et *PointerPDU (Cached)* seulement là où la capacité négociée *Pointer.colorPointerFlag* vaut TRUE. Là où les curseurs couleur ne sont pas supportés, on demande à une ASCE d’envoyer tous les changements de curseur couleur en curseur monochromes (voir 8.14.2). Il est recommandé que les ASCE supportent les curseurs couleur.

Les curseurs couleur peuvent être mis en antémémoire, ceci dépendant de la négociation des capacités *Pointer*. Si une ASCE supporte la mise en antémémoire de curseur couleur, elle mettra sa capacité annoncée *Pointer.pointerCacheSize* au nombre d’entrées de curseurs couleur mis en antémémoire qu'elle est prête à supporter pour chaque ASCE hôte (voir 8.2.11).

Là où les curseurs couleur sont supportés et où la mise en antémémoire de curseur couleur est activée à la suite d’une négociation de capacités (c'est-à-dire le *Pointer.pointerCacheSize* négocié est supérieur à 1), une ASCE émettrice peut allouer une définition de curseur couleur à une entrée d’antémémoire de curseur spécifique sur les ASCE réceptrices en utilisant un *PointerPDU (Color)* et ensuite réutiliser ultérieurement cette entrée mise en antémémoire en envoyant un

PointerPDU (Cached) référençant cette entrée. Il est laissé à la responsabilité de l'ASCE émettrice de gérer l'espace des entrées de l'antémémoire de curseurs couleur.

Sur réception d'un *PointerPDU (Color)* de la part d'une ASCE hôte particulière référençant une entrée d'antémémoire de curseur couleur, une ASCE placera la définition du curseur couleur dans l'entrée d'antémémoire pour cet hôte et affectera la définition fournie de curseur couleur comme le curseur couleur courant de cet hôte. Sur réception d'un *PointerPDU (Cached)* de la part d'une ASCE hôte particulière référençant une entrée d'antémémoire de curseur couleur, une ASCE affectera la définition de curseur couleur mise en antémémoire comme le curseur couleur courant pour cet hôte.

Là où la mise en antémémoire de curseur couleur est désactivée, une ASCE n'enverra pas l'ASPDU *PointerPDU (Cached)* et pourra seulement envoyer de l'information sur les changements de curseur local via une série de *PointerPDU (Color).* On demande à une ASCE réceptrice ayant la fonction mise en antémémoire de curseur couleur désactivée de se rappeler seulement de la dernière définition de curseur couleur de chaque ASCE hôte qui est toujours le curseur couleur courant pour cet hôte.

On demande à l'ASCE réceptrice de pister le curseur couleur courant de chaque ASCE hôte, mais l'affichage de ce curseur est une initiative purement locale.

Les données de curseur couleur représentent le curseur comme une série de rangées où la rangée zéro commence à la coordonnée Y du pixel le plus bas (à savoir, la rangée 0 commence au coin bas gauche). A l'intérieur d'une rangée, les valeurs de pixel sont empaquetées en octets à partir du pixel le plus à gauche. Pour les données de masque ET 1 bit/pixel, chaque octet contient huit pixels, avec le pixel le plus à gauche dans le bit le plus significatif. Pour les données de masque OU exclusif 24 bits/pixel, chaque triplet d'octets contient un pixel d'information de couleur RVB.

Les données de curseur couleur représentent un curseur comme un doublet de masques ET et OU exclusif, où le curseur peut être tracé en appliquant la fonction ET au masque ET et la fonction OU exclusif au masque OU exclusif, permettant ainsi de définir le point de visée (voir ci-dessous). Le masque ET est monochrome et le masque OU exclusif est couleur. C'est une représentation commune pouvant être facilement mappée de manière lisible sur des fonctions standard locales sur la plupart des terminaux rencontrés.

Le masque ET est composé d'une série de rangées de pixels monochromes (à savoir 1 bit par pixel), où la rangée zéro commence à la coordonnée Y du pixel le plus haut, c'est-à-dire la rangée 0 commence en haut à gauche. A l'intérieur d'une rangée, les valeurs de pixel sont empaquetées en octets, en commençant par le pixel le plus à gauche. Chaque octet contient 8 pixels, avec le pixel le plus à gauche dans le bit le plus significatif. Chaque rangée de curseur est alignée sur une frontière de double octet.

Le masque OU exclusif est composé de 24 bits/pixel, où la rangée zéro commence à la coordonnée Y du pixel le plus haut (à savoir la rangée 0 commence en haut à gauche). A l'intérieur d'une rangée, chaque triplet d'octets contient un pixel d'information couleur en RVB. A l'intérieur d'un triplet RVB, le premier octet est la valeur Bleu dans l'intervalle 0..255, le second est la valeur Vert dans l'intervalle 0..255 et le troisième la valeur Rouge dans l'intervalle 0..255. Les octets sont étroitement empaquetés – il n'y a pas d'octets de bourrage entre les valeurs RVB adjacentes. Chaque rangée de données de curseur est alignée sur une frontière de quatre octets.

Le point de visée du curseur couleur définit un point à l'intérieur du curseur correspondant à la position sur l'écran de l'endroit ou le curseur devrait être tracé. Par exemple, si le point de visée du curseur couleur est au point (3,4) à l'intérieur du curseur et que la position de curseur local est au point (50,50) alors le pixel le plus haut à gauche (à savoir 0,0) de la définition du curseur est tracé à la position (47,46) et le pixel du point de visée est tracé à la position (50,50). Voir les Tableaux 8-53 et 8.54.

Tableau 8-53/T.128 – PointerPDU (Couleur)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
cacheIndex	ce paramètre spécifie le curseur mis en antémémoire à utiliser. Les valeurs autorisées sont dans l'intervalle zéro à une valeur au dessous de la valeur *Pointer.pointerCacheSize* négociée. Voir 8.2.11 pour plus d'information sur le jeu de capacités *Pointer*.
hotSpotX	ce paramètre est la coordonnée en X du pixel de visée de curseur relative au coin haut gauche de la définition de curseur.
hotSpotY	ce paramètre est la coordonnée en Y du pixel de visée de curseur relative au coin haut gauche de la définition de curseur.
width	ce paramètre est la largeur du curseur en pixels.
height	ce paramètre est la hauteur du curseur en pixels.
colorPointer	c'est une définition de curseur couleur, composée d'un masque OU exclusif 24 bits/pixel et suivi d'un masque ET 1 bit/pixel.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Tableau 8-54/T.128 – PointerPDU (Mis en antémémoire)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
cacheIndex	ce paramètre spécifie le curseur mis en antémémoire à utiliser. Les valeurs autorisées sont dans l'intervalle zéro à une valeur au dessous de la valeur *Pointer.pointerCacheSize* négociée. Voir 8.2.11 pour plus d'information sur le jeu de capacités *Pointer*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.14.4 Mises à jour de position de curseur

Normalement, l'information de position de curseur est acheminée par des événements d'entrée pilotés par l'activité du dispositif de pointage de l'utilisateur. Voir 8.18 pour plus d'information sur le déplacement du dispositif de pointage. Cependant, certains terminaux permettent aux applications de mettre à jour par programme la position de curseur local.

Sur de tels terminaux, là où une application met à jour par programme la position de curseur local, une ASCE enverra un *PointerPDU (PointerPosition)* contenant la nouvelle position de curseur de moniteur virtuel. Sur réception d'un *PointerPDU (PointerPosition)* contenant une nouvelle position de curseur, une ASCE réceptrice met à jour la position courante du curseur de cet hôte.

On demande à l'ASCE réceptrice de pister le curseur courant des autres ASCE hôtes, mais la manière de traiter l'information de position de ce curseur est du ressort local. Voir le Tableau 8-55.

Tableau 8-55/T.128 – PointerPDU (PointerPosition)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
pointerX	ce paramètre est la coordonnée X du moniteur virtuel de la nouvelle position de curseur.
pointerY	ce paramètre est la coordonnée Y du moniteur virtuel de la nouvelle position de curseur.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.15 Mises à jour de palette

Une ASCE envoie des mises à jour de palette à toutes les ASCE dans la conférence en envoyant un *UpdatePDU* contenant une palette de la manière indiquée dans le Tableau 6-3. Le contenu de *UpdatePDU* contenant une palette est décrit au 8-56.

Par exemple, si une application locale change une palette locale et ensuite dessine une phototrame selon cette palette, alors l'ASCE s'assurera qu'elle envoie un *UpdatePDU* de palette avant le *UpdatePDU* de phototrame contenant les données de phototrame. A chaque fois qu'une ASCE envoie une nouvelle palette, elle s'assurera que toutes les fenêtres hébergées sur son terminal local sont redessinées, pour faire en sorte que les ASCE réceptrices puissent redessiner toutes les zones déjà reçues des fenêtres hébergées en référence à la nouvelle palette.

Le protocole AS supporte des profondeurs de couleur de 1, 4 et 8 bits/pixel (voir 8.2.4). Une ASCE n'enverra pas de mises à jour de palette là où le *sendingBitsPerPixel* vaut 1. Pour ce cas, le protocole définit respectivement les indices de palette 0 et 1 pour les couleurs blanc et noir.

Une palette contient 16 ou 256 valeurs de couleurs RVB. La manière dont les couleurs sont arrangées dans la palette est significatif et représente une séquence d'indices de palette dans l'intervalle 0..15 ou 0..255, ceci dépendant de *sendingBitsPerPixel*. Dans le mode de base du protocole AS, cette palette peut aussi contenir des informations optionnelles d'exactitude de couleur.

Sur réception d'un *UpdatePDU* contenant une palette, une ASCE utilisera cette palette pour interpréter les valeurs de points des données de phototrame ultérieures. Par exemple, dans des données entrantes de phototrame à 8 bits/pixel, l'ASCE réceptrice interprète un point de phototrame contenant la valeur 25 comme référençant la 26^{e} (indexation autorisée à partir de zéro) valeur de couleur dans la dernière palette reçue de l'ASCE émettrice. Elle n'utilisera pas la palette pour interpréter les valeurs de pixels de phototrames mises en antémémoire – qui devraient être interprétées en utilisant la table de couleurs mise en antémémoire appropriée (voir 8.16.10).

Tableau 8-56/T.128 – UpdatePDU (Palette)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
palette	ce paramètre est une liste de valeurs de couleurs constituant la palette. Dans le mode de base du protocole AS, la palette peut aussi contenir des informations optionnelles d'exactitude de couleur.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16 Mises à jour d'ordre

Une ASCE envoie des mises à jour d'ordres à toutes les ASCE dans la conférence en envoyant un *UpdatePDU* contenant des ordres de la manière indiquée dans le Tableau 6-3. Le contenu de *UpdatePDU* contenant des mises à jours d'ordres est décrit dans le Tableau 8-57.

Tableau 8-57/T.128 – UpdatePDU (Orders)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
orderList	ce paramètre contient une liste d'ordres.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Les mises à jour d'ordres peuvent être des deux types suivants:

- ordres primaires (voir le Tableau 8-58): les ordres primaires sont des ordres de tracé pouvant aboutir à des sorties sur les ASCE distantes, pouvant être sujets à troncature d'ASCE, à un ordre et/ou une présentation de fenêtre;
- ordres secondaires (voir le Tableau 8-59): les ordres secondaires fournissent des fonctions auxiliaires pour être utilisées ultérieurement par les ordres primaires. Par exemple, les ordres secondaires sont utilisés pour garnir des antémémoires distantes de tables de couleurs et de phototrames d'ASCE avant toute mise en antémémoire de références effectuées par des ordres primaires ultérieurs.

Un *UpdatePDU* contenant des ordres peut contenir n'importe quel mélange d'ordres primaires et secondaires.

Tableau 8-58/T.128 – Ordres primaires

Ordre	Référence
Destination Blt	voir le Tableau 8-65
Pattern Blt	voir le Tableau 8-66
Screen Blt	voir le Tableau 8-67
Memory Blt	voir le Tableau 8-70
Memory Three Way Blt	voir le Tableau 8-71
Text	voir le Tableau 8-72
Extended Text	voir le Tableau 8-73
Frame	voir le Tableau 8-74
Rectangle	voir le Tableau 8-75
Opaque Rectangle	voir le Tableau 8-76
Line	voir le Tableau 8-77
Desktop Save	voir le Tableau 8-78
Desktop Origin	voir le Tableau 8-79

Tableau 8-59/T.128 – Ordres secondaires

Ordre	Référence
Cache Bitmap (Compressed)	voir le Tableau 8-68
Cache Bitmap (Uncompressed)	voir le Tableau 8-68
Cache ColorTable	voir le Tableau 8-69
Color Space	voir le Tableau 8-80

8.16.1 Ordres primaires

Les ordres primaires contiendront le *Primary Order Header*. Le *Primary Order Header* du mode hérité (voir le Tableau 8-60) contient une information explicite de codage d'ordre, tandis que le *Primary Order Header* du mode de base (Voir le Tableau 8-61) n'en contient pas. Voir 8.16.3 pour plus d'information sur le codage d'ordre.

Une ASCE enverra un *UpdatePDU* contenant des ordres primaires seulement là où les capacités négociées indiquent que les autres ASCE peuvent recevoir des ordres (à savoir, dans le mode hérité, la capacité négociée *Order.orderFlags* n'a pas positionné son indicateur binaire *Cannot receive*; dans le mode de base la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre correspondant est supporté (à savoir dans le mode hérité, l'entrée de la capacité négociée correspondante *Order.ordersupport* est différente de zéro; dans le mode de base la capacité correspondante de niveau d'ordre négocié de cet ordre est différente de zéro). Il peut exister des restrictions ultérieures de manière spécifique à un ordre. Celles-ci sont documentées, là où elles sont applicables, dans la description de l'ordre primaire particulier.

Une ASCE peut fournir des coordonnées de frontières sur les ordres primaires. Les coordonnées de frontières définissent un rectangle de troncature pour l'ordre sur lequel elles sont présentes. Là où les coordonnées de frontières sont fournies sur un ordre (à la suite de tout décodage d'ordre requis), l'ASCE réceptrice tracera l'ordre tronqué aux frontières et au moniteur virtuel. Si les coordonnées de frontières ne sont pas fournies sur un ordre (à la suite de tout décodage d'ordre requis), l'ASCE réceptrice ne tronquera pas l'ordre, d'une manière autre que sur le moniteur virtuel. Là où les

coordonnées de frontières sont fournies (préalablement au codage d'ordre) dans le mode hérité du protocole AS, l'ASCE devrait positionner l'indicateur binaire *Bounds* dans les indicateurs de contrôle de l'en-tête d'ordre.

Tableau 8-60/T.128 – En-tête d'ordre primaire (mode hérité)

Paramètre	Description
controlFlags	ce paramètre indique les options de codage d'ordre applicables à l'ordre qui suit. Voir le Tableau 8-63 pour plus d'information sur les indicateurs de contrôle d'en-tête d'ordre.
orderType (Optionnel)	ce paramètre est présent là où le paramètre *controlFlags* possède l'indicateur *Type Change* positionné. Lorsqu'il est présent, il spécifie le type de l'ordre qui suit. Voir le Tableau 8-58 pour un résumé des valeurs autorisées.
encodingFlags	ce paramètre est composé de 1..3 octets indiquant le nombre de paramètres codables qui sont présents dans l'ordre. Voir le Tableau 8-64 pour plus d'information sur le nombre de paramètres codables autorisés par ordre primaire.
boundsFlags (Optionnel)	ce paramètre est présent là où le paramètre *ControlFlags* possède son indicateur *Bounds* positionné. Lorsqu'il est présent, c'est un jeu d'indicateurs binaires indiquant quelles sont les coordonnées de frontières présentes dans le paramètre de frontières (voir ci-dessous) et dans quel format, ce dernier étant fonction du codage des coordonnées de frontières. Les valeurs d'indicateurs binaires définies sont les suivantes: • *Absolute left bounds coordinate present* • *Absolute top bounds coordinate present* • *Absolute right bounds coordinate present* • *Absolute bottom bounds coordinate present* • *Delta left bounds coordinate present* • *Delta top bounds coordinate present* • *Delta right bounds coordinate present* • *Delta bottom bounds coordinate present* voir 8.16.3.2 pour plus d'information sur le codage des coordonnées de frontières
bounds (Optionnel)	ce paramètre est composé d'un ensemble de 0 à 4 coordonnées absolues ou relatives de frontières. Ceci dépend des indicateurs binaires positionnés dans le paramètre *boundFlags* (voir ci-dessus).

Tableau 8-61/T.128 – En-tête d'ordre primaire (mode de base)

Paramètre	Description
bounds (Optionnel)	ce paramètre est composé d'un ensemble de 0 à 4 coordonnées absolues ou relatives de frontières.

8.16.2 Ordres secondaires

Dans le mode hérité du protocole AS, les ordres secondaires contiendront le *Secondary Order Header* décrit dans le Tableau 8-62. Dans le mode de base du protocole AS, les ordres secondaires ne contiennent pas d'en-tête spécifique.

Une ASCE enverra un *UpdatePDU* contenant des ordres secondaires seulement là où les capacités négociées indiquent que les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité, la capacité négociée *Order.orderFlags* n'a pas positionné son indicateur binaire *Cannot receive*; dans le mode de base la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre correspondant est supporté (c'est-à-dire uniquement dans le mode de base la capacité correspondante de niveau d'ordre négocié de cet ordre est différente de zéro). Il peut exister des restrictions ultérieures de manière spécifique à un ordre. Celles-ci sont documentées, là où elles sont applicables, dans la description de l'ordre secondaire particulier.

Tableau 8-62/T.128 – En-tête d'ordre secondaire (mode hérité)

Paramètre	Description
controlFlags	ce paramètre indique les options de codage d'ordre applicables à l'ordre qui suit. Voir le Tableau 8-63 pour plus d'information sur les indicateurs de contrôle d'en-tête d'ordre.
extraFlags	ce paramètre est un jeu d'indicateurs binaires indiquant si l'ordre qui suit est primaire ou secondaire. La seule valeur d'indicateur binaire définie est *Secondary* qui sera positionnée.

8.16.3 Codage d'ordre

Le protocole AS utilise le codage d'ordre pour minimiser le nombre de paramètres à l'intérieur d'un ordre en se basant sur la comparaison des différences avec les ordres précédents. Il existe un nombre de méthodes de codage d'ordre définies qui lorsqu'elles sont applicables et appliquées, sont appliquées en série selon l'ordre suivant:

1) codage de type: voir 8.16.3.1;
2) codage des coordonnées de frontières: voir 8.16.3.2;
3) codage de paramètre: voir 8.16.3.3;
4) codage de coordonnée: voir 8.16.3.4.

Le codage d'ordre peut être appliqué seulement aux ordres primaires (c'est-à-dire que les ordres secondaires ne sont pas codés). Dans le mode hérité du protocole AS, l'information de codage d'ordre est acheminée via les *controlFlags*, *encodingFlags* et *boundsFlags* du *Primary Order Header* (voir le Tableau 8-60). Dans le mode de base du protocole AS, l'information de codage d'ordre est acheminée de manière implicite grâce à l'absence ou la présence des paramètres d'ordre optionnels.

Les deux modes requièrent que les ASCE réceptrices et émettrices supervisent le flux d'ordres envoyé dans les *UpdatePDU (Orders)* pour maintenir un état de codage et potentiellement de multiples états de décodage (lors de la réception d'ordres de la part d'ASCE hôtes multiples), pour que si des ASCE émettrices omettent des absences de changement ou fournissent des variations pour de petits changements dans les paramètres d'ordres, les ASCE réceptrices puissent reconstruire correctement après décodage le flux d'ordres d'origine.

Dans le mode hérité du protocole AS, tous les ordres contiennent un jeu initial d'indicateurs de contrôle indiquant comment ce qui reste de l'ordre devrait être interprété. Voir le Tableau 8-63 pour

plus d'information sur les indicateurs de contrôle autorisés. Pour les ordres primaires du mode hérité, une ASCE positionnera toujours l'indicateur de contrôle *Standard Encoding*, ne positionnera pas l'indicateur de contrôle *Secondary* et pourra positionner n'importe quel autre indicateur de contrôle selon ce qui est requis par le codage d'ordre. Pour les ordres secondaires dans le mode hérité, une ASCE positionnera toujours les indicateurs de contrôle *Standard Encoding* et *Secondary* et ne positionnera jamais les autres indicateurs de contrôle autorisés.

Tableau 8-63/T.128 – Indicateurs de contrôle d'en-tête d'ordre (mode hérité)

Indicateur de contrôle	Description
Standard Encoding	cet indicateur indique que l'ordre qui suit est conforme au codage normalisé pour la présente Recommandation. Cet indicateur sera positionné dans tous les ordres.
Secondary	cet indicateur (si positionné) indique que l'ordre qui suit est un ordre secondaire. Cet indicateur sera positionné seulement dans les ordres secondaires.
Bounds	cet indicateur (si positionné) indique que l'ordre qui suit inclut des frontières. Cet indicateur sera positionné seulement dans les ordres primaires. Voir 8.16.3.2 pour plus d'information sur le codage des coordonnées de frontières.
Type Change	cet indicateur (si positionné) indique que l'ordre qui suit est d'un type différent de celui de l'ordre précédent. Cet indicateur sera positionné seulement dans les ordres primaires. Voir 8.16.3.1 pour plus d'information sur le codage de type.
Delta Coordinates	cet indicateur (si positionné) indique que l'ordre qui suit utilise les coordonnées relatives (plutôt que les coordonnées absolues). Cet indicateur sera positionné seulement dans les ordres primaires. Voir 8.16.3.4 pour plus d'information sur le codage des coordonnées.

8.16.3.1 Codage de type

Dans le mode hérité du protocole AS, là où un ordre est du même type que celui de l'ordre précédent, alors une ASCE peut remettre à zéro l'indicateur binaire *Type Change* dans le paramètre *controlFlags* et omettre le paramètre *orderType* dans l'ordre codé. Là où un ordre primaire n'est pas du même type que celui de l'ordre précédent, une ASCE positionnera l'indicateur binaire *Type Change* dans le paramètre *controlFlags* et fournira le paramètre *orderType*.

Le codage de type est appliqué sur le flux continu d'ordres en tramant les ASPDU *UpdatePDU (Orders).*

Le codage de type n'est pas supporté dans le mode de base du protocole AS.

8.16.3.2 Codage des coordonnées de frontières

Lorsqu'une ou plusieurs coordonnées de frontières d'un ordre sont identiques à celles de l'ordre précédent (qui n'est pas nécessairement du même type), une ASCE peut omettre les coordonnées de frontières inchangées. Le codage d'ordre de frontières est appliqué à chaque coordonnée de frontières par rapport à la coordonnée de frontières correspondante de l'ordre précédent. Par exemple, lorsque les ordres ^{i} et $^{i+1}$ ont tous les deux une valeur de coordonnée gauche de frontières égale à 100, alors la coordonnée gauche peut être omise dans l'ordre $^{i+1}$. A l'opposé, lorsque les coordonnées de frontières haut gauche de l'ordre ^{i} sont de (110, 100) et celle haut gauche de l'ordre $^{i+1}$ de (100, 110), aucune des coordonnées gauche ou haute ne peut être omise.

Le codage de coordonnée de frontières est appliqué sur le flux continu d'ordres et relie ensemble les *ASPDUs UpdatePDU (Orders).*

Là où une coordonnée de frontières a été changée par rapport à l'ordre précédent, elles est alors soumise à codage (voir 8.16.3.4). Ceci signifie qu'un ordre particulier peut omettre certaines coordonnées de frontières et peut contenir un mélange de coordonnées de frontières relatives et absolues pour les coordonnées de frontières restantes.

Dans le mode de base du protocole AS, la présence ou l'absence de coordonnées de frontières, et, quand elles sont présentes les cas où elles sont représentées sous la forme de bits de coordonnées absolues ou relatives, est acheminée dans le code ASN.1 (voir le paragraphe 9).

Dans le mode hérité du protocole AS:

- là où toute ou partie des coordonnées de frontières sont omises, une ASCE remettra à zéro les indicateurs binaires de frontières absolues et relatives dans le paramètre *boundsFlag* du *Primary Order Header* (voir le Tableau 8-60);
- là où une ou plusieurs coordonnées de frontières ont changé par rapport à l'ordre précédent et que le changement est dans l'intervalle –128..+127 en pixels, une ASCE fournira des coordonnées relatives sur un seul octet pour les coordonnées de frontières et positionnera les bits de coordonnées relatives correspondantes dans le paramètre *boundsFlag*;
- là où une ou plusieurs coordonnées de frontières ont changé par rapport à l'ordre précédent et que le changement est hors de l'intervalle –128..+127 en pixels, une ASCE peut fournir des coordonnées absolues sur deux octets pour les coordonnées de frontières et positionnera les bits de coordonnées absolues correspondantes dans le paramètre *boundsFlag*.

8.16.3.3 Codage de paramètre

Là où un paramètre est identique au paramètre correspondant de l'ordre correspondant de même type, une ASCE peut omettre le paramètre inchangé. Par exemple, là où les ordres ^{i} et $^{i+n}$ (n >1) sont tous les deux des ordres *Pattern Blt* consécutifs et qu'ils ont tous les deux un paramètre ROP3 à 0xCC, alors le paramètre ROP3 peut être omis dans l'ordre $^{i+n}$ codé. Voir 8.16.5 pour plus d'information sur les ordres *Pattern Blts* et 8.16.20 pour plus d'information sur les *three-way ROPs*.

Le codage de coordonnée de frontières est appliqué sur le flux continu d'ordres et couvre l'ensemble des *ASPDU* de *UpdatePDU (Orders).*

Dans le mode de base du protocole AS, la présence ou l'absence de coordonnées de frontières est acheminée dans le code ASN.1 (voir le paragraphe 9).

Dans le mode hérité du protocole AS, la présence ou l'absence d'un paramètre particulier est indiquée par l'indicateur binaire dans le paramètre *encodingFlags* du *Primary Order Header* (voir le Tableau 8-60). Pour chaque ordre:

- le premier paramètre est le paramètre zéro;
- le nombre d'octets requis pour le paramètre *encodingFlags* est (nombre de paramètres codables + 7) div 8;
- l'indicateur binaire du paramètre N est représenté dans l'octet (N div 8) de *encodingFlags* à l'indicateur binaire (N mod 8), où le premier octet de *encodingFlags* est l'octet zéro et le premier bit dans un octet est le bit le moins significatif.

Les paramètres courants comme la brosse et le stylet sont codés en fonction de leurs paramètres constitutifs. Par exemple, le paramètre brosse dans l'ordre *Pattern Blt* (voir le Tableau 8-66), qui est documenté comme un paramètre simple (pour plus de clarté), contribue à hauteur de cinq paramètres codables sur le nombre total des douze paramètres codables de l'ordre *Pattern Blt*. De tels

regroupements de paramètres sont marqués comme (***Group***) dans les descriptions d'ordres respectives.

Par exemple, dans le codage de paramètre de l'ordre *Pattern Blt* (voir 8.16.5)

- le paramètre *foregroundColor* est le paramètre 6 et est représenté par l'octet 0, bit 6;
- le paramètre *brush hatch* est le paramètre 10 et est représenté par l'octet 1, bit 2.

Le Tableau 8-64 donne un aperçu du nombre de paramètres codables pour chaque ordre.

Tableau 8-64/T.128 – Ordres primaires: paramètres codables

Ordre	Nombre de paramètres codables	Nombre d'octets encodingFlags	Référence
Destination Blt	5	1	voir le Tableau 8-65
Pattern Blt	12	2	voir le Tableau 8-66
Screen Blt	7	1	voir le Tableau 8-67
Memory Blt	9	2	voir le Tableau 8-70
Memory Three Way Blt	17	3	voir le Tableau 8-71
Text	14	2	voir le Tableau 8-72
Extended Text	20	3	voir le Tableau 8-73
Frame	14	2	voir le Tableau 8-74
Rectangle	17	3	voir le Tableau 8-75
Opaque Rectangle	5	1	voir le Tableau 8-76
Line	10	2	voir le Tableau 8-77
Desktop Save	6	1	voir le Tableau 8-78
Desktop Origin	2	1	voir le Tableau 8-79

8.16.3.4 Codage de coordonnée

Après avoir appliqué un codage sur tous les paramètres (voir 8.16.3.3), une ASCE appliquera le codage de coordonnée à tous les paramètres de coordonnées restants.

Là où le changement de paramètre de coordonnée, par rapport au même paramètre de coordonnée de l'ordre précédent de même type, est dans l'intervalle –128..+127 en pixels, une ASCE peut fournir des coordonnées relatives de paramètre de coordonnée tenant sur un seul octet.

Dans le mode hérité du protocole AS, le codage de coordonnée requiert que toutes les coordonnées n'ayant pas été précédemment codées dans un ordre puissent être représentées en coordonnées relatives. Ainsi donc, là où le changement dans une ou plusieurs coordonnées non précédemment codées est hors de l'intervalle –128..+127 en pixels, une ASCE utilisera des coordonnées tenant sur deux octets pour tous les paramètres de coordonnées. Dans le mode de base du protocole AS, une telle restriction n'existe pas et une ASCE peut fournir un mélange de coordonnées relatives et absolues, en prenant en compte seulement le changement d'un paramètre de coordonnée non précédemment codé par rapport au même paramètre de l'ordre précédent de même type.

Dans le mode hérité du protocole AS, une ASCE positionnera ou remettra à zéro l'indicateur binaire *Delta Coordinates* dans le paramètre *controlFlags* pour indiquer si toutes les coordonnées sont acheminées en coordonnées absolues ou relatives. Dans le mode de base du protocole AS, la présence ou l'absence de coordonnées est acheminée dans le code ASN.1 (voir le paragraphe 9).

Le codage de coordonnée est appliqué en relation avec les paramètres de coordonnées (non précédemment codés) d'un ordre en relation avec les paramètres de coordonnées identiques d'un ordre précédent, ceci étant répété tout au long du flux continu d'ordres et relie ainsi les ASPDU *UpdatePDUs (Orders).*

Les paramètres de coordonnée éligibles pour un codage de coordonnée sont marqués comme (***Coordinate***) dans les descriptions de paramètres d'ordres et dans les descriptions de paramètres groupés.

8.16.4 Destination Blt

Une ASCE enverra un ordre *Destination Blt* là où les capacités négociées indique que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Destination Blt* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.DestinationBltLevel* est différente de zéro).

Une ASCE recevant l'ordre *Destination Blt* effectue le traitement de points de paramètre ROP3 sur le rectangle destination sur le moniteur virtuel, soumis éventuellement à troncature de frontières. Voir le Tableau 8-65.

Tableau 8-65/T.128 – Ordre Destination Blt

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de ce *Destination Blt*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haut de moniteur virtuel du rectangle destination de ce *Destination Blt*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du rectangle destination de ce *Destination Blt*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du rectangle destination de ce *Destination Blt*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour ce *Destination Blt*. Pour ce qui concerne les ordres *Destination Blt*, le *three-way ROP* référencera une destination et ne référencera pas une source ou un motif. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.5 Pattern Blt

Une ASCE enverra un ordre *Pattern Blt* là où les capacités négociées indique que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Pattern Blt* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.PatternBltLevel* est différente de zéro).

Une ASCE recevant l'ordre *Pattern Blt* effectue le traitement de points de paramètre ROP3 sur le rectangle destination sur le moniteur virtuel, soumis éventuellement à troncature de frontières. Voir le Tableau 8-66.

Tableau 8-66/T.128 – Ordre Pattern Blt

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de ce *Pattern Blt*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haut de moniteur virtuel du rectangle destination de ce *Pattern Blt*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du rectangle destination de ce *Pattern Blt*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du rectangle destination de ce *Destination Blt*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour ce *Pattern Blt*. Pour ce qui concerne les ordres *Pattern Blt*, le *three-way ROP* référencera une destination et ne référencera pas une source ou un motif. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
backgroundColor	ce paramètre est la couleur de fond à utiliser pour ce *Pattern Blt*.
foregroundColor	ce paramètre est la couleur de surface à utiliser pour ce *Pattern Blt*.
brush (Group)	ce paramètre est la couleur de brosse à utiliser pour ce *Pattern Blt*. Voir 8.16.22 pour plus d'information sur les brosses.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.6 Screen Blt

Une ASCE enverra un ordre *Screen Blt* là où les capacités négociées indique que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Screen Blt* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.ScreenBltLevel* est différente de zéro).

Une ASCE recevant l'ordre *Screen Blt* effectue le traitement de points de paramètre ROP3 sur le rectangle destination sur le moniteur virtuel, soumis éventuellement à troncature de frontières.

- Le rectangle source est défini par les paramètres *SourceX* et *SourceY* et les paramètres *destWidth* et *destHeight*.
- Le rectangle destination est défini par les paramètres *DestLeft* et *DestTop* et les paramètres *destWidth* et *destHeight*.
- Les paramètres *DestWidth* et *DestHeight* définissent la largeur et la hauteur des rectangles destination et source. Ceci exclut tout étirement.
- La source peut recouvrir la destination.

Voir le Tableau 8-67.

Tableau 8-67/T.128 – Ordre Screen Blt

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de ce *Screen Blt*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haut de moniteur virtuel du rectangle destination de ce *Screen Blt*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du rectangle destination de ce *Screen Blt*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du rectangle destination de ce *Screen Blt*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour ce *Pattern Blt*. Pour ce qui concerne les ordres *Screen Blt*, le *three-way ROP* référencera une destination et ne référencera pas une source ou un motif. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
sourceX (Coordinate)	ce paramètre est la coordonnée *Source X* du moniteur virtuel pour ce *Screen Blt*.
sourceY (Coordinate)	ce paramètre est la coordonnée *Source Y* du moniteur virtuel pour ce *Screen Blt*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.7 Mise en antémémoire de phototrame (*Cache Bitmap*)

Une ASCE enverra un ordre *Cache Bitmap* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où les ordres *Memory Blt* ou *Memory Three Way Blt* sont supportés (c'est-à-dire dans le mode hérité, l'entrée négociée correspondante *Order.OrderSupport* est différente de zéro; en mode de base, l'entrée des capacités négociées *Order.MemoryBltLevel* ou *Order.MemoryThreeWayBltLevel* est différente de zéro). De plus, dans le mode de base, la capacité négociée *Order.CacheBitmapLevel* sera différente de zéro.

L'ASCE émettrice s'assurera ultérieurement que les paramètres de l'ordre *Cache Bitmap* reflètent la conclusion de la négociation des capacités *Bitmap* et *Bitmap Cache* (voir 8.2.4 et 8.2.7), c'est-à-dire que:

- elle enverra un ordre *Cache Bitmap (Compressed)* seulement là où les valeurs du jeu de capacités négocié *Bitmap* indiquent que la compression de phototrame est supportée;
- le *bitmapBitsPerPixel* est égal à la valeur négociée *sendingBitsPerPixel*;
- la valeur *cacheID* référence une zone d'antémémoire de phototrame où la valeur négociée indique au moins une entrée;
- la valeur *cacheIndex* dans cette zone d'antémémoire de phototrame est inférieure au nombre d'entrées;
- la taille de la phototrame (après compression) rentre dans la taille maximale de cellules négociée pour cette zone d'antémémoire de phototrame.

L'ASCE émettrice est responsable de l'allocation des paramètres *cacheID* et *cacheIndex* et donc, du peuplement et de la mise à jour d'antémémoire de phototrame d'ASCE réceptrice. Ceci requiert que les ASCE hôtes pistent l'usage d'antémémoire en se basant sur les ordres *Cache Bitmap* envoyés précédemment.

Une ASCE recevant un ordre *Cache Bitmap* d'une ASCE hôte particulière place la phototrame fournie dans la zone d'antémémoire de phototrame *CacheID*, à l'entrée *cacheIndex* correspondant à cette ASCE hôte. La phototrame fournie remplace toute phototrame existante mise en antémémoire à cet emplacement. Ceci requiert que toutes les ASCE actives maintiennent une antémémoire de phototrame séparée pour chaque ASCE hôte. Voir le Tableau 8-68.

Tableau 8-68/T.128 – Ordre Cache Bitmap

Paramètre	Description
Secondary Order Header	le *Secondary Order Header* est décrit au 8.16.2.
cacheID	ce paramètre indique quelle antémémoire de phototrame devrait être utilisée pour cette phototrame. Les valeurs autorisées, qui dépendent de la négociation de *Bitmap Cache*, sont dans l'intervalle 0..2. Voir 8.2.7 pour plus d'information sur la négociation de *Bitmap Cache*.
bitmapWidth	ce paramètre est la largeur en pixels de la phototrame.
bitmapHeight	ce paramètre est la hauteur en pixels de la phototrame.
bitmapBitsPerPixel	ce paramètre est le nombre de bits par pixel de la donnée phototrame. Ce paramètre sera identique à la valeur de capacité négociée *Bitmap.sendingBitsPerPixel*. Voir 8.2.4 pour plus d'information.
cacheIndex	ce paramètre indique quelle entrée d'antémémoire utiliser dans l'antémémoire particulière indiquée par le paramètre *cacheID* (voir ci-dessus). Les valeurs autorisées dépendent pour chaque antémémoire de l'issue de la négociation des tailles de cellules d'antémémoire des capacités *Bitmap Cache*. Voir 8.2.7 pour plus d'information sur la négociation de *Bitmap Cache*.
bitmapData	ce paramètre est la donnée de phototrame à cacher. La phototrame peut être non compressée (voir 8.17.1) ou compressée (voir 8.17.2).
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.8 Mise en antémémoire de tableau de couleurs (*Cache ColorTable*)

Une ASCE enverra un ordre *Cache ColorTable* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où les ordres *Memory Blt* ou *Memory Three Way Blt* sont supportés (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.MemoryBltLevel* ou *Order.MemoryThreeWayBltLevel* est différente de zéro; dans le mode de base, la capacité négociée *Order.CacheColorTableLevel* est différente de zéro).

L'ASCE émettrice s'assurera ultérieurement que les paramètres de l'ordre *Cache ColorTable* reflètent la conclusion de la négociation des capacités *Bitmap* et *ColorTable Cache* (voir 8.2.4 et 8.2.8), c'est-à-dire que:

- le nombre d'entrées dans la table de couleurs est fonction de la valeur *sendingBitsPerPixel* comme suit:

  nombre d'entrées de table de couleurs = 2 à la puissance *sendingBitsPerPixel*

  (ceci donne un nombre autorisé d'entrées de table de couleurs de 16 ou 256);
- la valeur *cacheIndex* dans la table de couleurs est inférieure au nombre d'entrées négocié.

Une ASCE n'enverra pas l'ordre *Cache ColorTable* là où le *Bitmap.sendingBitsPerPixel* vaut 1. Pour ce cas, le protocole AS définit les indices de table de couleurs 0 et 1 comme respectivement les couleurs noire et blanche. Voir 8.2.4 pour plus d'information sur le jeu de capacités *Bitmap*.

L'ASCE émettrice est responsable de l'allocation du paramètre *cacheIndex* et donc, du peuplement et de la mise à jour d'antémémoire de table de couleurs d'ASCE réceptrice. Ceci requiert que les ASCE hôtes pistent l'usage d'antémémoire de tables de couleurs en se basant sur les ordres *Cache ColorTable* envoyés précédemment.

Une ASCE recevant un ordre *Cache ColorTable* place la table de couleurs fournie dans la zone d'antémémoire de table de couleurs, à l'entrée *cacheIndex*. La table de couleurs fournie remplace toute table de couleurs existante mise en antémémoire à cet emplacement. Ceci requiert que toutes les ASCE actives maintiennent une antémémoire de table de couleurs séparée pour chaque ASCE hôte.

Une table de couleurs contient 16 ou 256 couleurs RVB. L'arrangement des valeurs de couleurs dans la table de couleurs est significatif et représente une séquence d'indices de table de couleurs allant de 0..15 ou de 0..255, ceci dépendant de *sendingBitsPerPixel*. Dans le mode de base du protocole AS, la table de couleurs contient aussi une information optionnelle d'exactitude de couleur. Voir le Tableau 8-69.

Tableau 8-69/T.128 – Ordre Cache ColorTable

Paramètre	Description
Secondary Order Header	le *Secondary Order Header* est décrit au 8.16.2.
cacheIndex	ce paramètre indique quelle entrée d'antémémoire utiliser dans l'antémémoire de table de couleurs. Les valeurs autorisées dépendent pour chaque antémémoire de l'issue de la négociation des tailles de cellules d'antémémoire des capacités *ColorTable Cache*. Voir 8.2.8 pour plus d'information sur la négociation de *ColorTable Cache*
colorTable	ce paramètre est une liste de valeurs de couleurs comprenant la table de couleur à mettre en antémémoire. Dans le mode de base du protocole AS, la table de couleurs peut aussi contenir une information optionnelle d'exactitude de couleur.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.9 Memory Blt

Une ASCE enverra un ordre *Memory Blt* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (à savoir, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Memory Blt* est supporté (à savoir dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *OrderMemoryBltLevel* est différente de zéro).

L'ASCE émettrice s'assurera que les paramètres *bitmapCacheID*, *bitmapCacheIndex* et *colorTableCacheIndex* font référence à une phototrame et une table de couleurs précédemment mises en antémémoire en utilisant les ordres *Cache Bitmap* et *Cache ColorTable*.

Une ASCE recevant l'ordre *Memory Blt* effectue le traitement de points de paramètre ROP3 en utilisant le rectangle source de la phototrame mise en antémémoire et le rectangle destination sur le moniteur virtuel, soumis éventuellement à troncature de frontières.

- Le rectangle source est défini par les paramètres *SourceX* et *SourceY* et les paramètres *destWidth* et *destHeight*.
- Le rectangle destination est défini par les paramètres *destleft* et *destTop* et les paramètres *destWidth* et *destHeight*.
- Les paramètres *destWidth* et *destHeight* définissent la largeur et la hauteur des rectangles destination et source. Ceci exclut tout étirement.
- Le rectangle source est complètement à l'intérieur des dimensions de la phototrame mise en antémémoire. L'ASCE réceptrice n'a pas besoin d'effectuer de troncature du rectangle source par rapport aux dimensions de la phototrame mise en antémémoire.
- La phototrame mise en antémémoire sera interprétée en utilisant la table de couleurs référencée mise en antémémoire.

Voir le Tableau 8-70.

Tableau 8-70/T.128 – Ordre Memory Blt

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
colorTableCacheIndex	ce paramètre spécifie la table de couleurs mise en antémémoire à utiliser pour ce *Memory Blt*. Voir 8.16.8 pour plus d'information sur la mise en antémémoire de *ColorTable*. pour les ordres *Memory Blt* dans le mode hérité, ce paramètre et le paramètre *bitmapCacheID* (voir ci-dessous) sont tous les deux présents là où l'indicateur binaire approprié est positionné dans le paramètre *encodingFlags* du *Primary Order Header* (voir la Note).
bitmapCacheID	lorsqu'il est utilisé en association avec le paramètre *bitmapCacheIndex* (voir ci-dessous), il spécifie la phototrame mise en antémémoire à utiliser pour ce *Memory Blt*. Voir 8.16.7 pour plus d'information sur la mise en antémémoire de phototrame. pour les ordres *Memory Blt* dans le mode hérité, ce paramètre et le paramètre *colorTableCacheIndex* (voir ci-dessus) sont tous les deux présents là où l'indicateur binaire approprié est positionné dans le paramètre *encodingFlags* du *Primary Order Header* (voir la Note).

Tableau 8-70/T.128 – Ordre Memory Blt *(fin)*

Paramètre	Description
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche du moniteur virtuel du rectangle destination de ce *Memory Blt*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haute du moniteur virtuel du rectangle destination de ce *Memory Blt*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du moniteur virtuel du rectangle destination de ce *Memory Blt*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du moniteur virtuel du rectangle destination de ce *Memory Blt*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour ce *Memory Blt*. Pour ce qui concerne les ordres *Memory Blt*, le *three-way ROP* référencera une destination et ne référencera pas une source ou un motif. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
sourceX (Coordinate)	ce paramètre est la coordonnée source X dans la phototrame référencée mise en antémémoire de ce *Memory Blt.*
sourceY (Coordinate)	ce paramètre est la coordonnée source Y dans la phototrame référencée mise en antémémoire de ce *Memory Blt.*
bitmapCacheIndex	lorsqu'il est utilisé en association avec le paramètre *bitmapCacheID* (voir ci-dessus), il spécifie la phototrame mise en antémémoire à utiliser pour ce *Memory Blt*. Voir 8.16.7 pour plus d'information sur la mise en antémémoire de phototrame.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.
NOTE – Pour les ordres *Memory Blt* du mode hérité, les paramètres *colorTableCacheIndex* et *bitmapCacheID* sont comptés comme un seul paramètre dans les *Primary Order Header encodingFlag*s. Voir 8.16.3.3 pour plus d'information.	

8.16.10 Memory Three Way Blt

Une ASCE enverra un ordre *Memory Three Way Blt* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Memory Three Way Blt* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *OrderMemoryThreeWayBltLevel* est différente de zéro).

L'ASCE émettrice s'assurera que les paramètres *bitmapCacheID*, *bitmapCacheIndex* et *colorTableCacheIndex* font référence à une phototrame et une table de couleurs précédemment mises en antémémoire en utilisant les ordres *Cache Bitmap* et *Cache ColorTable*.

Une ASCE recevant l'ordre *Memory Three Way Blt* effectue le traitement de points de paramètre ROP3 en utilisant la brosse, le rectangle source de la phototrame mise en antémémoire et le rectangle destination sur le moniteur virtuel, soumis éventuellement à troncature de frontières.

- Le rectangle source est défini par les paramètres *SourceX* et *SourceY* et les paramètres *destWidth* et *destHeight*.

- Le rectangle destination est défini par les paramètres *destleft* et *destTop* et les paramètres *destWidth* et *destHeight*.
- Les paramètres *destWidth* et *destHeight* définissent la largeur et la hauteur des rectangles destination et source. Ceci exclut tout étirement.
- Le rectangle source est complètement à l'intérieur des dimensions de la phototrame mise en antémémoire. L'ASCE réceptrice n'a pas besoin d'effectuer de troncature du rectangle source par rapport aux dimensions de la phototrame mise en antémémoire.
- La phototrame mise en antémémoire sera interprétée en utilisant la table de couleurs référencée mise en antémémoire.

Voir le Tableau 8-71.

Tableau 8-71/T.128 – Ordre Memory Three Way Blt

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
colorTableCacheIndex	ce paramètre spécifie la Table de couleurs mise en antémémoire à utiliser pour ce *Memory Three Way Blt*. Voir 8.16.8 pour plus d'information sur la mise en antémémoire de *ColorTable*. pour les ordres *Memory Three Way Blt* dans le mode hérité, ce paramètre et le paramètre *bitmapCacheID* (voir ci-dessous) sont tous les deux présents là où l'indicateur binaire approprié est positionné dans le paramètre *encodingFlags* du *Primary Order Header* (voir la Note).
bitmapCacheID	lorsqu'il est utilisé en association avec le paramètre *bitmapCacheIndex* (voir ci-dessous), il spécifie la phototrame mise en antémémoire à utiliser pour ce *Memory Three Way Blt*. Voir 8.16.7 pour plus d'information sur la mise en antémémoire de phototrame. pour les ordres *Memory Three Way Blt* dans le mode hérité, ce paramètre et le paramètre *colorTableCacheIndex* (voir ci-dessus) sont tous les deux présents là où l'indicateur binaire approprié est positionné dans le paramètre *encodingFlags* du *Primary Order Header* (Voir la Note).
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche du moniteur virtuel du rectangle destination de ce *Memory Three Way Blt*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haute du moniteur virtuel du rectangle destination de ce *Memory Three Way Blt*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du moniteur virtuel du rectangle destination de ce *Memory Three Way Blt*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du moniteur virtuel du rectangle destination de ce *Memory Three Way Blt*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour ce *Memory Three Way Blt*. Pour ce qui concerne les ordres *Memory Blt*, le *three-way ROP* référencera une destination et ne référencera pas une source ou un motif. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
sourceX (Coordinate)	ce paramètre est la coordonnée sourceX dans la phototrame référencée mise en antémémoire de ce *Memory Three Way Blt.*
sourceY (Coordinate)	ce paramètre est la coordonnée sourceY dans la phototrame référencée mise en antémémoire de ce *Memory Three Way Blt.*
backgroundColor	ce paramètre est la couleur de fond à utiliser pour ce *Memory Three Way Blt*.

Tableau 8-71/T.128 – Ordre Memory Three Way Blt *(fin)*

Paramètre	Description
foregroundColor	ce paramètre est la couleur de surface à utiliser pour ce *Memory Three Way Blt*.
brush (Group)	ce paramètre est la brosse à utiliser pour ce *Memory Three Way Blt*. Voir 8.16.22 pour plus d'information sur les brosses.
bitmapCacheIndex	lorsqu'il est utilisé en association avec le paramètre *bitmapCacheID* (voir ci-dessus), il spécifie la phototrame mise en antémémoire à utiliser pour ce *Memory Three Way Blt*. Voir 8.16.7 pour plus d'information sur la mise en antémémoire de phototrame.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.
NOTE – Pour les ordres *Memory Three Way Blt* du mode hérité, les paramètres *colorTableCacheIndex* et *bitmapCacheID* sont comptés comme un seul paramètre dans les *Primary Order Header encodingFlag*s. Voir 8.16.3.3 pour plus d'information.	

8.16.11 Texte (*Text*)

Une ASCE enverra un ordre *Text* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Text* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.TextLevel* est différente de zéro), la police particulière a été mise en correspondance et les codes correspondent à l'intervalle spécifié des codes de police.

L'ordre *Text* permet à une ASCE de spécifier la position de départ en Y du texte en termes de base de premier caractère ou sommet de cellule de premier caractère. Il est recommandé que les ASCE utilisent le positionnement par la base à tout moment où les capacités l'autorisent comme il peut être difficile sur certains terminaux de positionner de manière exacte le texte correspondant à des polices variables sur la seule base de la position du coin haut gauche de cellule du premier caractère.

Une ASCE enverra des ordres *Text* en utilisant le positionnement par la base là où les capacités négociées indiquent que le positionnement de départ de texte par la base est autorisé (c'est-à-dire, dans le mode hérité l'indicateur binaire *Baseline Start* de la capacité négociée *Order.textFlags* est positionné; dans le mode de base la capacité négociée *Order.baselineStartFlag* vaut TRUE). Là où c'est le cas, elle positionnera l'indicateur binaire *Baseline Start* de la capacité négociée *Order.textFlags* et positionnera les ordres *startX* et *startY* comme étant la position du premier pixel bas gauche de la cellule du premier caractère. Sinon, là où les capacités négociées indiquent que le positionnement de départ de texte par la base n'est pas autorisé, elle remettra à zéro l'indicateur binaire *Baseline Start* de la capacité négociée *Order.textFlags* et positionnera les ordres *startX* et *startY* comme étant la position du premier pixel haut gauche de la cellule du premier caractère.

Certains types de terminaux permettent aux applications de spécifier dynamiquement des attributs de texte de police au moment du tracé, en plus des attributs propres à la police. Par exemple, si un terminal local supporte la police *Courier Bold*, alors elle peut autoriser ses applications à tracer du texte en utilisant cette police en association avec l'attribut *Italic* pour émuler la police *Courier Bold Italic* (qui peut présenter un aspect différent de la police réelle *Courier Bold Italic* correspondant à une police *Courier Bold* disponible). Là où le terminal local supporte les attributs de texte dynamiques de cette manière, une ASCE émettrice peut utiliser les indicateurs binaires *Italic*,

Underline et *StrikeOut* des paramètres *textFlags* et le paramètre *fontWeight* (seule ou en association) pour indiquer que les ASCE réceptrices devraient appliquer les attributs correspondants au texte tracé.

Une ASCE recevant un ordre *Text* trace les codes de caractères sur le moniteur virtuel, soumis éventuellement à troncature de frontières, comme suit:

- les caractères sont tracés dans *foregroundColor*. Si le *backMixMode* est *Opaque*, les fonds de cellules de caractère sont tracés dans *backgroundColor*;
- là où l'indicateur binaire *Baseline Start* du paramètre *textFlags* de l'ordre est positionné, *startX* et *startY* sont la position du premier pixel bas gauche de la cellule du premier caractère. Sinon, *startX* et *startY* sont la position du premier pixel haut gauche de la cellule du premier caractère;
- les caractères sont positionnés sans *extraSpacing* appliqué à tous les caractères et sans *break spacing* (égal à *totalBreakSpacing* divisé par *breakCount*) appliqué au caractère *Break*;
- les caractères sont tracés dans la police locale correspondant au *FontID* de l'ASCE émettrice, soumis à l'ajout éventuel de tout attribut indiqué dans les paramètres *textFlags* et/ou *fontWeight*. Voir 8.8.2 pour plus d'information sur les polices;
- si la police locale est variable, les caractères sont tracés selon les *fontWidth* et *fontHeight* fournies. Sinon, il sont tracés selon la largeur et la hauteur propres à la police;
- les caractères sont tracés selon la *fontWeight* et pour tout indicateur binaire *Italic*, *Underline* et *StrikeOut* fourni selon le paramètre *textFlags*.

Voir le Tableau 8-72.

Tableau 8-72/T.128 – Text Order

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
backMixMode	ce paramètre est le mode de fond mixte utilisé pour cet ordre *Text*. Voir 8.16.24 pour plus d'information sur les modes de fond mixtes.
startX (Coordinate)	ce paramètre est la coordonnée de moniteur virtuel *start X* pour cet ordre *Text*.
startY (Coordinate)	ce paramètre est la coordonnée de moniteur virtuel *start Y* pour cet ordre *Text*. Là où l'indicateur binaire *Baseline Start* est positionné dans le paramètre *textFlags*, celle-ci correspond au point de base de la cellule du premier caractère. Là où l'indicateur binaire *Baseline Start* n'est pas positionné dans le paramètre *textFlags*, celle-ci correspond au sommet de la cellule du premier caractère.
backgroundColor	ce paramètre est la couleur d'arrière-plan à utiliser pour cet ordre *Text*.
foregroundColor	ce paramètre est la couleur d'avant-plan à utiliser pour cet ordre *Text*.
extraSpacing	ce paramètre spécifie l'espacement supplémentaire en pixels devant être appliqué individuellement aux caractères pour cet ordre *Text*. Une valeur égale à zéro indique qu'aucun espacement supplémentaire n'est appliqué.
totalBreakSpacing	ce paramètre spécifie le nombre total d'espacements supplémentaires en pixels devant être appliqué aux caractères *break* pour cet ordre *Text*. Une valeur égale à zéro indique qu'aucun espacement supplémentaire n'est appliqué.

Tableau 8-72/T.128 – Text Order *(fin)*

Paramètre	Description
breakCount	ce paramètre spécifie le nombre total caractères *break* pour cet ordre *Text*. Une valeur égale à zéro indique qu'aucun espacement *break* doit être appliqué ou qu'il n'existe aucun caractère *break* dans cet ordre.
fontHeight	pour les polices variables, ce paramètre est la hauteur en pixels de la police utilisée pour cet ordre *Text*. Voir 8.8 pour plus d'information.
fontWidth	pour les polices variables, ce paramètre est la largeur moyenne en pixels de la police utilisée pour cet ordre *Text*. Voir 8.8 pour plus d'information.
fontWeight	ce paramètre indique le poids de la police à utiliser pour cet ordre *Text*. Les valeurs autorisées sont dans l'intervalle 0..1000, qui sont interprétées comme: *light* ≤ 400< *normal* ≤ 700< *bold*. Voir 8.8 pour plus d'information.
textFlags	ce paramètre est un jeu d'indicateurs binaires indiquant les caractéristiques de la police à utiliser pour cet ordre *Text*. Les valeurs d'indicateurs binaires définies sont les suivantes: • *Italic* • *Underline* • *StrikeOut* • *Baseline Start*
fontID	ce paramètre est le *FontID* de l'ASCE émettrice, déterminé au cours de la négociation de police, à utiliser pour cet ordre *Text*. Voir 8.8 pour plus d'information sur la négociation et le mappage de police.
codePointList	ce paramètre est une liste de codes à utiliser pour cet ordre *Text*. Les codes seront dans la page de code du protocole AS. Voir 8.8.1 pour plus d'information sur les codes et les pages de codes.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.12 Texte étendu (*Extended Text*)

Une ASCE enverra un ordre *Extended Text* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Extended Text* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *OrderExtendedTextLevel* est différente de zéro), la police particulière a été mise en correspondance et les codes correspondent à l'intervalle spécifié des codes de police.

- Une ASCE n'enverra pas d'ordre *Extended Text* sans les codes pour tracer les rectangles opaques. Elle devrait à la place utiliser l'ordre Opaque Rectangle (voir 8.16.15).
- Une ASCE peut envoyer un ordre *Extended Text* avec des ajustements de position *DeltaX* pour les polices qui correspondent approximativement à la position en X.

L'ordre *Extended Text* permet à une ASCE de spécifier la position de départ en Y du texte en termes de base de premier caractère ou sommet de cellule de premier caractère. Il est recommandé que les ASCE utilisent le positionnement par la base à tout moment où les capacités l'autorisent comme il

peut être difficile sur certains terminaux de positionner de manière exacte le texte correspondant à des polices variables sur la seule base de la position du coin haut gauche de cellule du premier caractère.

Une ASCE enverra des ordres *Extended Text* en utilisant le positionnement par la base seulement là où les capacités négociées indiquent que le positionnement de départ de texte par la base est autorisé (c'est-à-dire, dans le mode hérité l'indicateur binaire *Baseline Start* de la capacité négociée *Order.textFlags* est positionné; dans le mode de base la capacité négociée *Order.baselineStartFlag* vaut TRUE). Là où c'est le cas, elle positionnera l'indicateur binaire *Baseline Start* de la capacité négociée *Order.textFlags* et positionnera les ordres *startX* et *startY* comme étant la position du premier pixel bas gauche de la cellule du premier caractère. Sinon, là où les capacités négociées indiquent que le positionnement de départ de texte par la base n'est pas autorisé, elle remettra à zéro l'indicateur binaire *Baseline Start* de la capacité négociée *order textFlags1* et positionnera les ordres *startX* et *startY* comme étant la position du premier pixel haut gauche de la cellule du premier caractère.

Certains types de terminaux permettent aux applications de spécifier dynamiquement des attributs de texte de police au moment du tracé, en plus des attributs propres à la police. Par exemple, si un terminal local supporte la police *Courier Bold*, alors elle peut autoriser ses applications à tracer du texte en utilisant cette police en association avec l'attribut *Italic* pour émuler la police *Courier Bold Italic* (qui peut présenter un aspect différent de la police réelle *Courier Bold Italic* correspondant à une police *Courier Bold* disponible). Là où le terminal local supporte les attributs de texte dynamiques de cette manière, une ASCE émettrice peut utiliser les indicateurs binaires *Italic*, *Underline* et *StrikeOut* des paramètres *textFlags1* et le paramètre *fontWeight* (seul ou en association) pour indiquer que les ASCE réceptrices devraient appliquer les attributs correspondants au texte tracé.

Une ASCE recevant un ordre *Extended Text* trace les codes de caractères sur le moniteur virtuel, soumis éventuellement à troncature de frontières, comme suit:

- les caractères sont tracés dans *foregroundColor*. Si l'indicateur binaire *Opaque Rectangle* de l'ordre *textFlags2* est positionné, le rectangle de troncature de l'ordre est tracé dans *backgroundColor*. Si l'indicateur binaire *Opaque Rectangle* de l'ordre *textFlags2* n'est pas positionné et que le *backMixMode* est *Opaque*, les fonds de cellules de caractères sont tracés dans *backgroundColor*;
- là où l'indicateur binaire *Baseline Start* du paramètre *textFlags1* de l'ordre est positionné, *startX* et *startY* sont la position du premier pixel bas gauche de la cellule du premier caractère. Sinon, *startX* et *startY* sont la position du premier pixel haut gauche de la cellule du premier caractère;
- si l'indicateur binaire *DeltaXPresent* du paramètre *textFlag2* de l'ordre est positionné, les caractères sont positionnés en utilisant les valeurs *Delta X* fournies dans le paramètre *deltaXList*. Sinon, les caractères sont positionnés sans appliquer *d'extraSpacing* à tous les caractères et sans espacement de *break* (égal à *breakTotalSpacing* divisé par *breakCount*) appliqué au caractère *break*;
- les caractères sont tracés dans la police locale correspondant au *FontID* de l'ASCE émettrice, soumis à l'ajout éventuel de tout attribut indiqué dans les paramètres *textFlags1* et/ou *fontWeight*. Voir 8.8.2 pour plus d'information sur les polices;
- si la police locale est variable, les caractères sont tracés selon les *fontWidth* et *fontHeight* fournies. Sinon, ils sont tracés selon la largeur et la hauteur propres à la police;

- les caractères sont tracés selon la *fontWeight* et pour tout indicateur binaire *Italic*, *Underline* et *StrikeOut* fourni selon le paramètre *textFlags1*;
- si l'indicateur binaire *Clip to Rectangle* du *textFlag2* de l'ordre est positionné, les caractères sont tronqués selon le rectangle de troncature de l'ordre.

Voir le Tableau 8-73.

Tableau 8-73/T.128 – Ordre Extended Text

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
backMixMode	ce paramètre est le mode de fond mixte utilisé pour cet ordre *Extended Text*. Voir 8.16.24 pour plus d'information sur les modes de fond mixtes.
startX (Coordinate)	ce paramètre est la coordonnée de moniteur virtuel *start X* pour cet ordre *Extended Text*.
startY (Coordinate)	ce paramètre est la coordonnée de moniteur virtuel *start Y* pour cet ordre *Extended Text*. Là où l'indicateur binaire *Baseline Start* est positionné dans le paramètre *textFlags1*, celle-ci correspond au pixel de base de la cellule du premier caractère. Là où l'indicateur binaire *Baseline Start* n'est pas positionné dans le paramètre *textFlags1*, celle-ci correspond au sommet de la cellule du premier caractère.
backgroundColor	ce paramètre est la couleur de fond à utiliser pour cet ordre *Extended Text*.
foregroundColor	ce paramètre est la couleur de surface à utiliser pour cet ordre *Extended Text*.
extraSpacing	ce paramètre spécifie l'espacement supplémentaire en pixels devant être appliqué individuellement aux caractères pour cet ordre *Extended Text*. Une valeur égale à zéro indique qu'aucun espacement supplémentaire n'est appliqué.
totalBreakSpacing	ce paramètre spécifie le nombre total d'espacements supplémentaires en pixels devant être appliqué aux caractères *break* pour cet ordre *Extended Text*. Une valeur égale à zéro indique qu'aucun espacement supplémentaire n'est appliqué.
breakCount	ce paramètre spécifie le nombre total caractères *break* pour cet ordre *Extended Text*. Une valeur égale à zéro indique qu'aucun espacement *break* doit être appliqué ou qu'il n'existe aucun caractère *break* dans cet ordre.
fontHeight	pour les polices variables, ce paramètre est la hauteur en pixels de la police utilisée pour cet ordre *Extended Text*. Voir 8.8 pour plus d'information.
fontWidth	pour les polices variables, ce paramètre est la largeur moyenne en points de la police utilisée pour cet ordre *Extended Text*. Voir 8.8 pour plus d'information.
fontWeight	ce paramètre indique le poids de la police à utiliser pour cet ordre *Extended Text*. Les valeurs autorisées sont dans l'intervalle 0..1000, qui sont interprétées comme: *light* ≤ 400< *normal* ≤ 700< *bold*. Voir 8.8 pour plus d'information.

Tableau 8-73/T.128 – Ordre Extended Text *(fin)*

Paramètre	Description
textFlags1	ce paramètre est un jeu d'indicateurs binaires indiquant les caractéristiques de la police à utiliser pour cet ordre *Extended Text*. Les valeurs d'indicateurs binaires définies sont les suivantes. • *Italic* • *Underline* • *StrikeOut* • *Baseline Start*
fontID	ce paramètre est le *FontID* de l'ASCE émettrice, déterminé au cours de la négociation de police, à utiliser pour cet ordre *Extended Text*. Voir 8.8 pour plus d'information sur la négociation et la correspondance de police.
textFlags2	ce paramètre est un jeu d'indicateurs binaires spécifiant les options supplémentaires pour cet ordre *Extended Text*. Les valeurs d'indicateurs binaires définies sont les suivantes. • *Opaque Rectangle* • *Clip to Rectangle* • *DeltaXPresent*
clipLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle de troncature pour cet ordre *Extended Text*. Là où les indicateurs binaires *Opaque Rectangle* ou *Clip to Rectangle* du paramètre *textFlags2* ne sont pas positionnés, ce paramètre sera égal à zéro.
clipTop (Coordinate)	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle de troncature pour cet ordre *Extended Text*. Là où les indicateurs binaires *Opaque Rectangle* ou *Clip to Rectangle* du paramètre *textFlags2* ne sont pas positionnés, ce paramètre sera égal à zéro.
clipRight (Coordinate)	ce paramètre est la coordonnée X droite de moniteur virtuel du rectangle de troncature pour cet ordre *Extended Text*. Là où les indicateurs binaires *Opaque Rectangle* ou *Clip to Rectangle* du paramètre *textFlags2* ne sont pas positionnés, ce paramètre sera égal à zéro.
clipBottom (Coordinate)	ce paramètre est la coordonnée Y basse de moniteur virtuel du rectangle de troncature pour cet ordre *Extended Text*. Là où les indicateurs binaires *Opaque Rectangle* ou *Clip to Rectangle* du paramètre *textFlags2* ne sont pas positionnés, ce paramètre sera égal à zéro.
codePointList	ce paramètre est une liste de codes à utiliser pour cet ordre *Extended Text*. Les codes seront dans la page de code du protocole AS. Voir 8.8.1 pour plus d'information sur les codes et les pages de codes.
deltaXList (Coordinate)	ce paramètre est une liste de coordonnées X relatives de moniteur virtuel à utiliser pour cet ordre *Extended Text*. Ce paramètre est présent seulement là où l'indicateur binaire *DeltaXPresent* est positionné dans le paramètre *textFlags2*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.13 Trame (*Frame*)

Une ASCE enverra un ordre *Frame* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Frame* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.FrameLevel* est différente de zéro).

Une ASCE recevant l'ordre *Frame* effectue le traitement de points de paramètre ROP3 en utilisant la brosse et la bordure de trame sur le moniteur virtuel, soumis éventuellement à troncature de frontières.

- Le rectangle de bordure externe est défini par les paramètres *destLeft*, *destTop*, *destRight* and *destBottom*.
- Le rectangle de bordure interne est obtenu par addition ou soustraction des *destWidth* et *destHeight* respectivement des limites horizontales et verticales du rectangle de trame externe.

Voir le Tableau 8-74.

Tableau 8-74/T.128 – Ordre Frame

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de cet ordre *Frame*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle destination de cet ordre *Frame*.
destRight (Coordinate)	ce paramètre est la coordonnée X droite de moniteur virtuel du rectangle destination de cet ordre *Frame*.
destBottom (Coordinate)	ce paramètre est la coordonnée Y basse de moniteur virtuel du rectangle destination de cet ordre *Frame*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels des limites verticales de rectangle pour cet ordre *Frame*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels des limites verticales de rectangle pour cet ordre *Frame*.
ROP3	ce paramètre est le *three-way ROP* à utiliser pour cet ordre *Frame*. Dans le cas des ordres *Frame*, le *three-way ROP* ne référencera pas une source. Voir 8.16.20 pour plus d'information sur les *three-way ROPs*.
backgroundColor	ce paramètre est la couleur d'arrière-plan à utiliser pour cet ordre *Frame*.
foregroundColor	ce paramètre est la couleur d'avant-plan à utiliser pour cet ordre *Frame*.
brush (Group)	ce paramètre est la brosse à utiliser pour cet ordre *Frame*. Voir 8.16.22 pour plus d'information sur les brosses.
nonStandardParameters	Ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.14 Rectangle (*Rectangle*)

Une ASCE n'enverra un ordre *Rectangle* que là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et que l'ordre *Rectangle* est supporté (c'est-à-dire que dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; et que dans le mode de base, la capacité négociée *Order.RectangleLevel* est différente de zéro).

Une ASCE recevant un ordre *Rectangle* effectue deux opérations associées. Elle utilise le traitement de points de paramètre ROP2, la brosse et (en fonction du type de brosse) le mode mixte de fond pour tracer l'intérieur du rectangle sur le moniteur virtuel, avec une éventuelle troncature aux frontières. Elle utilise aussi le traitement de points de paramètre ROP2, le stylet et le mode mixte de fond pour tracer une délimitation autour du rectangle sur le moniteur virtuel, soumis à une éventuelle troncature aux frontières.

- L'intérieur du rectangle est défini par *destLeft* + 1, *destTop* + 1, *destRight* –1 et *destBottom –1*.
- Le pourtour est défini par la séquence de points (*destLeft*, *destTop*), (*destRight*, *destTop*), (*destRight*, *destBottom*), (*destLeft*, *destBottom*), (*destLeft*, *destTop*).

Voir le Tableau 8-75.

Tableau 8-75/T.128 – Ordre Rectangle

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
backMixMode	ce paramètre est le mode mixte de fond à utiliser pour cet ordre *Rectangle*. Voir 8.16.24 pour plus d'information sur les modes mixtes de fond.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de cet ordre *Rectangle*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle destination de cet ordre *Rectangle*.
destRight (Coordinate)	ce paramètre est la coordonnée X droite de moniteur virtuel du rectangle destination de cet ordre *Rectangle*.
destBottom (Coordinate)	ce paramètre est la coordonnée Y basse de moniteur virtuel du rectangle destination de cet ordre *Rectangle*.
backgroundColor	ce paramètre est la couleur d'arrière-plan à utiliser pour cet ordre *Rectangle*.
foregroundColor	ce paramètre est la couleur d'avant-plan à utiliser pour cet ordre *Rectangle*.
brush (Group)	ce paramètre est la brosse à utiliser pour cet ordre *Rectangle*. Voir 8.16.22 pour plus d'information sur les brosses.
ROP2	ce paramètre est le *two-way ROP* à utiliser pour cet ordre *Rectangle*. Voir 8.16.21 pour plus d'information sur les *two-way ROPs*.
pen (Group)	ce paramètre est le stylet à utiliser pour cet ordre *Rectangle*. Voir 8.16.23 pour plus d'information sur les stylets.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.15 Rectangle plein (*Opaque Rectangle*)

Une ASCE enverra un ordre *Opaque Rectangle* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Opaque Rectangle* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.OpaqueRectangleLevel* est différente de zéro).

Une ASCE recevant un ordre *Opaque Rectangle* remplit le rectangle destination sur le moniteur virtuel avec la couleur fournie, soumis à une éventuelle troncature aux frontières. Voir le Tableau 8-76.

Tableau 8-76/T.128 – Ordre Opaque Rectangle

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
destLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de cet ordre *Opaque Rectangle*.
destTop (Coordinate)	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle destination de cet ordre *Opaque Rectangle*.
destWidth (Coordinate)	ce paramètre est la largeur en pixels du rectangle pour cet ordre *Opaque rectangle*.
destHeight (Coordinate)	ce paramètre est la hauteur en pixels du rectangle pour cet ordre *Opaque rectangle*.
color	ce paramètre est la couleur à utiliser pour cet ordre *Opaque Rectangle*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.16 Trait (*Line*)

Une ASCE enverra un ordre *Line* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Line* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.LineLevel* est différente de zéro).

Une ASCE recevant un ordre *Rectangle* utilise le traitement de points de paramètre ROP2, le stylet et le mode mixte de fond pour tracer une ligne à partir de *startX*, *startY* jusqu'à *endX*, *endY* sur le moniteur virtuel, soumise à une éventuelle troncature aux frontières. Il faut noter que la couleur de surface de ligne est spécifiée dans le stylet. Voir le Tableau 8-77.

Tableau 8-77/T.128 – Ordre Line

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
backMixMode	ce paramètre est le mode mixte de fond à utiliser pour cet ordre *Line*. Voir 8.16.24 pour plus d'information sur les modes mixtes de fond.
startX (Coordinate)	ce paramètre est la coordonnée *start X* de moniteur virtuel pour cet ordre *Line*.
startY (Coordinate)	ce paramètre est la coordonnée *start Y* de moniteur virtuel pour cet ordre *Line*.
endX (Coordinate)	ce paramètre est la coordonnée *end X* de moniteur virtuel pour cet ordre *Line*.
endY (Coordinate)	ce paramètre est la coordonnée *end Y* de moniteur virtuel pour cet ordre *Line*.
backgroundColor	ce paramètre est la couleur de fond à utiliser pour cet ordre *Line*.
ROP2	ce paramètre est le *two-way ROP* à utiliser pour cet ordre *Line*. Voir 8.16.21 pour plus d'information sur les *two-way ROPs*.
pen (Group)	ce paramètre est le stylet à utiliser pour cet ordre *Line*. Voir 8.16.23 pour plus d'information sur les stylets.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.17 Sauvegarde de moniteur (*Desktop Save*)

Un gestionnaire de fenêtre de terminal (s'il est présent) peut fournir des opérations locales de sauvegarde et de restauration des zones de moniteur, que ce soit de manière automatisée ou sous le contrôle d'une application locale, pour améliorer la performance de scénarios types d'applications. Cet attribut a pour référence "*save/restore*" ou "*save-under*". Par exemple:

- un utilisateur fait apparaître un menu qui masque une partie de sa propre fenêtre d'application (zone A);
- ensuite l'utilisateur sélectionne un élément de menu ouvrant une boîte de dialogue (et faisant disparaître le menu), cette même boîte masquant une partie différente de la fenêtre d'application (zone B);
- l'utilisateur sélectionne ensuite un bouton de dialogue ouvrant ensuite une boîte de dialogue masquant une partie du premier dialogue et/ou une partie différente de la fenêtre d'application (zone C);
- l'utilisateur désactive ensuite les deux boîtes de dialogue.

Là où le gestionnaire de fenêtre fournit un mécanisme *save/restore*, la zone A est sauvegardée lorsque le menu apparaît et restaurée lorsqu'il disparaît. De la même manière, la zone B est sauvegardée lorsque la première fenêtre de dialogue est créée, la zone C est sauvegardée lorsque la seconde fenêtre de dialogue est créée et ensuite les zones C et B sont restaurées lorsque leurs fenêtres de dialogues respectives sont détruites. Il faut noter que de multiples zones peuvent être sauvegardées en même temps, mais la séquence des sauvegardes et des restaurations est telle qu'elle se comporte comme une pile LIFO (dernier entré, premier sorti).

Le fait d'utiliser des *save/restore* est souvent significativement plus rapide que de forcer un gestionnaire de fenêtre et/ou une application à redessiner les zones masquées. L'ordre *Desktop Save* permet à une ASCE de propager les *save/restore* locaux vers d'autres ASCE et génère souvent une amélioration significative des performances du protocole AS.

Le mécanisme de sauvegarde de moniteur requiert que chaque ASCE le supportant alloue une antémémoire de moniteur pour chacun des autres ASCE hôtes pour recevoir les zones sauvegardées. Là où une sauvegarde modifie les applications hébergées sur une ASCE particulière, cette ASCE enverra un ordre *Desktop Save* à toutes les autres ASCE en spécifiant la zone à sauvegarder, à la suite de quoi les ASCE réceptrices sauvegarderont la zone correspondante du moniteur virtuel sur l'antémémoire de moniteur. Lorsque la restauration correspondante se produit, cette ASCE envoie un ordre *Desktop Save* à toutes les autres ASCE en spécifiant la zone à restaurer, à la suite de quoi les ASCE réceptrices restaureront la zone correspondante de l'antémémoire de moniteur vers le moniteur virtuel.

L'antémémoire de moniteur est organisée logiquement comme un ensemble de fichiers dont la taille est une granularité spécifique en X et en Y. La granularité préférée pour une ASCE particulière dépend de l'efficacité relative des sauvegardes et des restaurations pour des tuiles de taille différentes entre le moniteur virtuel et l'antémémoire de moniteur et le degré de perte de mémoire au fur et à mesure de l'accroissement de la taille de tuile. Cependant, alors qu'une ASCE particulière peut annoncer ses granularités préférées, ces valeurs sont négociées et l'ASCE doit être préparée à recevoir des ordres *Desktop Save* construits selon des valeurs plus grandes (et peut-être moins efficaces). Voir l'Appendice I pour les détails concernant les valeurs informatives des tailles et granularités d'antémémoire de moniteur.

Un ordre *Desktop Save* spécifie une action (qui est soit *save* ou *restore*), la zone à sauvegarder/restaurer (sous la forme d'un rectangle) et un déplacement de points dans l'antémémoire de moniteur. L'ASCE émettrice calcule le déplacement de pixels sur la base de l'utilisation précédente de l'antémémoire de moniteur des autres ASCE et des granularités négociées en X et en Y comme suit:

A l'initialisation, remettre le décalage cumulé de pixels à zéro; pour chaque ordre de sauvegarde de moniteur:

décalage de pixel pour l'ordre = décalage cumulé de pixels
calculer de la manière suivante les pixels requis pour cette sauvegarde/restauration:

largeur zone en pixels = ((largeur zone en pixels + (granularité X négociée −1)) div granularité X négociée) * granularité X négociée;

hauteur zone en pixels = ((hauteur zone en pixels + (granularité Y négociée −1)) div granularité Y négociée) * granularité Y négociée;

Nb pixels requis pour la zone = largeur zone en pixels * hauteur zone en pixels;

s'il s'agit d'une sauvegarde, ajouter le nombre de pixels requis pour cette zone au décalage cumulé de pixels;

s'il s'agit d'une restauration, soustraire le nombre de pixels requis pour cette zone du décalage cumulé de pixels.

Une ASCE enverra un ordre *Desktop Save* là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, dans le mode hérité la capacité négociée *Order.orderFlags* n'a pas positionné l'indicateur binaire *Cannot receive*; dans le mode de base, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et là où l'ordre *Desktop Save* est supporté (c'est-à-dire dans le mode hérité, l'entrée de la capacité négociée *Order.orderSupport* est différente de zéro; dans le mode de base, la capacité négociée *Order.DesktopSaveLevel* est différente de zéro).

Là où l'ASCE émettrice détectera qu'une autre antémémoire de moniteur d'ASCE est complète, alors elle n'enverra plus d'ordres *Desktop Save* jusqu'à ce qu'elle puisse restaurer des zones déjà présentes dans l'antémémoire de moniteur. Ceci peut avoir pour conséquence l'utilisation d'une action localement dépendante sur le terminal pour émuler l'opération de sauvegarde et/ou forcer un retraçage local lors d'une restauration. Voir le Tableau 8-78.

Tableau 8-78/T.128 – Ordre Desktop Save

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
saveOffset	ce paramètre est le déplacement en pixels à l'intérieur de l'antémémoire de moniteur de l'ASCE réceptrice à utiliser pour cet ordre *Desktop Save*.
desktopLeft (Coordinate)	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle de moniteur pour cette action de l'ordre *Desktop Save*.
desktopTop (Coordinate)	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle de moniteur pour cette action de l'ordre *Desktop Save*.
desktopRight (Coordinate)	ce paramètre est la coordonnée X droite de moniteur virtuel du rectangle de moniteur pour cette action de l'ordre *Desktop Save*.
desktopBottom (Coordinate)	ce paramètre est la coordonnée Y basse de moniteur virtuel du rectangle de moniteur pour cette action de l'ordre *Desktop Save*.
action	ce paramètre identifie cette action de l'ordre *Desktop Save*. Les valeurs autorisées *DesktopSave* et *DesktopRestore*.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.18 Origine de moniteur (*Desktop Origin*)

Là où un moniteur local d'ASCE est plus petit que le moniteur virtuel, alors une ASCE peut fournir localement des mécanismes prédéfinis pour faire défiler le moniteur local sur le moniteur virtuel pour faire en sorte qu'un utilisateur final puisse voir et contrôler les fenêtres reflets hébergées sur des ASCE ayant des moniteurs plus grands. Ce mécanisme est appelé défilement de moniteur.

Là où une ASCE implémente le défilement de moniteur, le desktop local se comporte comme un port de visualisation sur le moniteur local et l'origine du moniteur local peut être décalée par rapport à celle du moniteur d'origine.

Là où une ASCE est en train d'héberger des fenêtres et que son origine de moniteur local ne coïncide pas avec l'origine du moniteur virtuel, elle enverra un ordre *Desktop Origin* pour informer les autres ASCE actives de l'origine des moniteurs virtuels dans les coordonnées de moniteur virtuel – à savoir, relativement à l'origine du moniteur virtuel. Ce mécanisme assure une ASCE de toujours connaître l'origine du moniteur s'appliquant à chaque ASPDU entrante et à chaque ordre à l'intérieur d'un *UpdatePDU (Orders)* provenant de chaque ASCE.

Une ASCE s'assurera qu'elle envoie un ordre *Desktop Origin* avec l'envoi de toute ASPDU contenant des coordonnées de moniteur virtuel calculées relativement à l'origine du moniteur.

Une ASCE s'assurera que là où elle envoie un ordre *Desktop Origin*, toute coordonnée de moniteur virtuel envoyée préalablement à cet ordre *Desktop Origin* sera calculée relativement à l'origine de l'ancien moniteur et que toute coordonnée de moniteur virtuel envoyée à la suite de cet ordre *Desktop Origin* sera calculée relativement à la nouvelle origine de moniteur. Il se peut que cette condition requière de la part d'une ASCE de transférer les ordres et/ou les autres ASPDU avant ou après l'envoi d'un ordre *Desktop Origin*.

Une ASCE peut utiliser les ordres *Desktop Origin* en provenance d'une ASCE particulière pour implémenter un mécanisme de défilement de moniteur pour les fenêtres reflets hébergées par cette ASCE, en soustrayant l'origine de moniteur de l'ASCE émettrice des coordonnées de moniteur virtuel des ASPDU entrantes.

Dans le mode hérité du protocole AS, une ASCE enverra l'ordre *Desktop Origin* seulement là où l'entrée de la capacité négociée *Order.orderSupport* pour l'ordre *Screen Blt* est différente de zéro. Dans le mode de base du protocole AS, une ASCE enverra l'ordre *Desktop Origin* seulement là où les capacités négociées indiquent que les autres ASCE peuvent recevoir des ordres (c'est-à-dire la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et l'ordre *Desktop Origin* est supporté (c'est-à-dire la capacité négociée *Order.DesktopOriginLevel* est différente de zéro). Voir le Tableau 8-79.

Tableau 8-79/T.128 – Ordre Desktop Origin

Paramètre	Description
Primary Order Header	le *Primary Order Header* est décrit au 8.16.1.
desktopOriginX (Coordinate)	ce paramètre est la coordonnée X de moniteur virtuel de l'origine de moniteur local de l'ASCE émettrice.
desktopOriginY (Coordinate)	ce paramètre est la coordonnée Y de moniteur virtuel de l'origine de moniteur local de l'ASCE émettrice.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.19 Espace de couleurs (*Color Space*)

Dans le mode de base du protocole AS, une ASCE peut recevoir l'ordre *Color Space* pour spécifier l'espace de couleurs valide pour les ordres à venir et pour fournir des informations optionnelles d'exactitude de couleur. Une ASCE peut aussi utiliser l'ordre *Color Space* pour restaurer l'espace de couleurs par défaut – RVB sans information d'exactitude de couleurs. L'ordre *Color Space* n'est pas supporté dans le mode hérité du protocole AS.

Une ASCE enverra l'ordre *Color Space* dans le mode de base du protocole AS seulement là où les capacités négociées indiquent que toutes les autres ASCE peuvent recevoir des ordres (c'est-à-dire, la capacité négociée *Order.receiveOrdersFlag* vaut TRUE) et l'ordre *Color Space* est supporté (c'est-à-dire la capacité négociée *Order.ColorSpaceLevel* est différente de zéro).

Une ASCE recevant un ordre *Color Space* devrait faire au mieux pour interpréter les informations de couleur d'ordres ultérieurs relativement à l'espace de couleurs spécifié dans l'ordre *Color Space*. Voir le Tableau 8-80.

Tableau 8-80/T.128 – Ordre Color Space

Paramètre	Description
colorSpace	ce paramètre est l'espace de couleurs et l'information d'exactitude de couleur optionnelle à utiliser lors de l'interprétation des ordres suivants.
nonStandardParameters	c'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.16.20 Opérations point-à-point ternaires (*Three-Way ROPs*)

Les *three-way ROPs* sont des opérations point-à-point ternaires utilisées par les ordres suivants.

- *Destination Blt*: voir 8.16.4.
- *Pattern Blt*: voir 8.16.5.
- *Screen Blt*: voir 8.16.6.
- *Memory Blt*: voir 8.16.9.
- *Memory Three Way Blt*: voir 8.16.10.
- *Frame*: voir 8.16.13.

Les codes d'opérations point-à-point ternaires définissent la combinaison des bits dans une phototrame source et une brosse (connue comme une forme) avec les bits d'une phototrame destination. Ce sont des opérations bit-à-bit agissant séparément sur des bits sans aucune interprétation de couleur, où les bits peuvent faire partie d'indices de palettes ou de valeurs directes de couleurs.

Ce qui suit représente les trois opérandes utilisés dans ces opérations.

- D Phototrame destination
- P Brosse sélectionnée (appelée aussi forme)
- S Phototrame source

Ce qui suit représente les opérateurs booléens utilisés dans ces opérations.

- a ET bit-à-bit
- n NON bit-à-bit (inverse)
- o OU bit-à-bit
- x OU exclusif bit-à-bit (XOR)

Toutes les combinaisons d'opérateurs booléens peuvent être représentées en notation Polonaise Inversée. Par exemple, l'opération suivante remplace les bits dans la phototrame destination avec un OU bit-à-bit effectué sur les bits de la source et de la brosse:

PSo

A nouveau, l'opération suivante réalise un OU bit-à-bit exclusif des bits de la source et de la brosse avec les bits de la phototrame destination:

DPSoo

Chaque code d'opération point-à-point est une valeur d'octet simple représentant le résultat de l'opération booléenne sur les valeurs de brosse, de source et destination prédéfinies. Par exemple, les codes d'opération point-à-point des opérations PSo et DPSoo sont décrits dans la liste suivante:

Valeurs sources			Valeurs générées	
P	**S**	**D**	**Pso**	**DPSoo**
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1
			⇒0xFC	⇒0xFE

Dans ce cas, PSo a le code d'opération point-à-point 0xFC (en lisant les valeurs binaires générées comme un octet ayant le bit le moins significatif au sommet). DPSoo a le code d'opération point-à-point 0xFE.

Les codes d'opération point-à-point peuvent être interprétés directement pour déterminer si l'opération point-à-point ternaire fait référence à un opérande source, forme ou destination, comme suit:

- l'opération point-à-point ternaire ne référence pas une forme: bits 0.. 3 = bits 4..7;
- l'opération point-à-point ternaire ne référence pas une source: bits 0, 1, 4, 5 = bits 2, 3, 6, 7;
- l'opération point-à-point ternaire ne référence pas une destination: bits 0, 2, 4, 6 = bits 1, 3, 5, 7.

La liste complète des codes d'opération point-à-point ternaires est décrite dans le Tableau 8-81, avec la définition Polonaise Inversée pour chaque code[5].

Il faut noter que comme les opérations point-à-point agissent directement sur les valeurs binaires, sans interpréter la couleur, les résultats des opérations point-à-point peuvent délivrer des effets de couleurs imprévisibles lors de leur application sur différentes ASCE. Ceci peut être plus particulièrement notable pour les opérations point-à-point qui référencent un opérande destination, telle que l'inversion de destination (c'est-à-dire, le code Dn ou ROP 0x55 – Voir ci-dessous) qui est souvent utilisé pour implémenter des effets de marquage). Si une application hébergée utilise une opération point-à-point de ce type, là où l'ASCE hôte fonctionne sur un terminal utilisant une profondeur de couleur et/ou un modèle de couleur local différent et là où l'opération locale est envoyée comme un ordre, alors le résultat sur la couleur peut être très différent entre celle de l'ASCE hôte et celles des autres ASCE.

Considérons deux ASCE impliquées dans le partage d'application à l'intérieur d'une conférence où l'ASCE A présente des applications hôtes. Le terminal local de l'ASCE A utilise un modèle de couleur RVB palettisé à 8 bits par pixel tandis que le terminal local de l'ASCE B en utilise un de 4 bits par pixels. Alors que le différence de profondeur de couleur affectera la capacité *Bitmap.sendingBitsPerPixel* négociée et donc la profondeur de couleur des informations de palette et de table de couleurs envoyées de l'ASCE A vers l'ASCE B, l'ASCE B aura à faire correspondre

[5] Il existe des épellations alternatives de Polonaise Inversée d'un même code d'opération, si bien qu'une épellation particulière pouvant ne pas être dans le tableau, une forme équivalente sera présente. Par exemple, DSa est équivalente à SDa.

l'information de couleur du protocole, les ordres et la donnée de phototrame sur l'écran de son terminal local, qui présente une résolution de 4 bits par pixels. L'exemple suivant omet de préciser les étapes de mappage intermédiaires (à savoir l'information de protocole envoyée à n'importe quelle antémémoire réceptrice de l'écran du terminal local) et se concentre uniquement sur le mappage de couleur telle qu'elle affecte les opérations point-à-point de la destination.

- Une application sensible à la couleur fonctionnant sur le terminal local de l'ASCE A ajoute des couleurs à une portion de la palette du terminal local, de sorte que sa palette à 8 bits par pixel contienne (entre autres) les entrées suivantes:
  - indice de palette 0x63 ⇒ RVB <204,0,102> ("violet rouge");
  - indice de palette 0x9C ⇒ RVB <204,204,204> ("gris clair").
- Le terminal local de l'ASCE B n'autorise pas la gestion de palette d'application. Sa palette à 4 bits par pixel contient (entre autres) les entrées suivantes:
  - indice de palette 0x1 ⇒ RVB <128,0,0> ("rouge foncé");
  - indice de palette 0xE ⇒ RVB <0,255,255> ("cyan").
- Une application sur le terminal local de l'ASCE remplit une zone de fenêtre hébergée avec RVB <204,0,102> en utilisant une opération graphique locale. Le système graphique du terminal local génère une zone de pixels à l'écran contenant l'indice de palette 0x63 ("violet rouge").
- L'ASCE A envoie un ordre *Pattern Blt* pour cette zone, en utilisant l'opération point-à-point 0xCC (S) et la couleur de fond RVB <204,0,102>. Ceci suppose un mappage un pour un entre l'information de couleur locale et le protocole (ce qui n'est pas nécessairement toujours le cas).
- L'ASCE B exécute l'ordre *Pattern Blt* sur l'écran du terminal local, générant une zone de pixels d'écran contenant l'indice de palette 0x1 ("rouge foncé"). Le mappage de couleur peut être un mappage explicite fait par l'ASCE B ou peut reposer sur un mappage de couleur locale fourni par le terminal local.
- L'application sur le terminal local de l'ASCE A inverse alors la zone remplie en utilisant une opération graphique locale. Ceci a pour résultat d'inverser à la valeur 0x9C ("gris clair") la zone des pixels de l'écran.
- L'ASCE A envoie un ordre *Destination Blt* pour cette zone, en utilisant l'opération point-à-point 0x55 (Dn).
- L'ASCE B exécute l'ordre *Destination Blt* sur l'écran du terminal local. Ceci a pour résultat d'inverser à la valeur 0xE ("cyan") la zone des pixels de l'écran.

Bien que le mappage initial entre les couleurs "violet rouge" et "rouge foncé" est raisonnable, en tenant compte des contraintes liées aux modèles de couleurs du terminal local, le "cyan" n'est pas un bon mappage pour le "gris clair". Cet exemple illustre que l'effet de l'opération point-à-point sur la destination est de déplacer de manière arbitraire des valeurs de couleur (en changeant directement les indices de couleur) sans aucune référence aux mappages de couleurs. En pratique, l'extension des perturbations des opérations point-à-point sur la destination dépend des palettes en vigueur sur le terminal local considéré – là où les terminaux locaux utilisent tous les deux des palettes linéaires (du type de celles typiquement utilisées en tant que palettes de terminal local par défaut), les perturbations sont relativement bénignes, mais là où un ou plusieurs terminaux locaux ou leurs applications locales utilisent des palettes non linéaires ou manipulant de manière explicite les contenus de palette du terminal, les perturbations peuvent être significatives.

Alors que la discussion ci-dessus indique que le partage d'application introduit des problèmes de respects de couleur supplémentaires à travers l'utilisation d'ordres contenant des opérations point-à-point référençant un opérande destination, l'effet de ces émissions dépend du terminal et de l'application. Cependant, le fait d'émettre de telles opérations comme des ordres plutôt que comme des données de phototrame délivre typiquement une performance significativement meilleure. Ainsi donc, on recommande aux ASCE de fournir des mécanismes sur le terminal local par lesquels l'utilisateur peut activer ou désactiver l'envoi d'ordres contenant des opérations point-à-point référençant un opérande destination, pour que les utilisateurs puissent choisir entre le respect de couleur et/ou la performance terminal par terminal et/ou application par application. Voir le Tableau 8-81.

Tableau 8-81/T.128 – Définition de Three-Way ROPs

Code ROP (hexadécimal)	Polonaise inversée
00	0
01	DPSoon
02	DPSona
03	PSon
04	SDPona
05	DPon
06	PDSxnon
07	PDSaon
08	SDPnaa
09	PDSxon
0A	DPna
0B	PSDnaon
0C	SPna
0D	PDSnaon
0E	PDSonon
0F	Pn
10	PDSona
11	DSon
12	SDPxnon
13	SDPaon
14	DPSxnon
15	DPSaon
16	PSDPSanaxx
17	SSPxDSxaxn
18	SPxPDxa
19	SDPSanaxn
1A	PDSPaox
1B	SDPSxaxn

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(suite)*

Code ROP (hexadécimal)	Polonaise inversée
1C	PSDPaox
1D	DSPDxaxn
1E	PDSox
1F	PDSoan
20	DPSnaa
21	SDPxon
22	DSna
23	SPDnaon
24	SPxDSxa
25	PDSPanaxn
26	SDPSaox
27	SDPSxnox
28	DPSxa
29	PSDPSaoxxn
2A	DPSana
2B	SSPxPDxaxn
2C	SPDSoax
2D	PSDnox
2E	PSDPxox
2F	PSDnoan
30	Psna
31	SDPnaon
32	SDPSoox
33	Sn
34	SPDSaox
35	SPDSxnox
36	SDPox
37	SDPoan
38	PSDPoax
39	SPDnox
3A	SPDSxox
3B	SPDnoan
3C	PSx
3D	SPDSonox
3E	SPDSnaox
3F	Psan
40	PSDnaa
41	DPSxon
42	SDxPDxa
43	SPDSanaxn

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(suite)*

Code ROP (hexadécimal)	Polonaise inversée
44	Sdna
45	DPSnaon
46	DSPDaox
47	PSDPxaxn
48	SDPxa
49	PDSPDaoxxn
4A	DPSDoax
4B	PDSnox
4C	SDPana
4D	SSPxDSxoxn
4E	PDSPxox
4F	PDSnoan
50	Pdna
51	DSPnaon
52	DPSDaox
53	SPDSxaxn
54	DPSonon
55	Dn
56	DPSox
57	DPSoan
58	PDSPoax
59	DPSnox
5A	DPx
5B	DPSDonox
5C	DPSDxox
5D	DPSnoan
5E	DPSDnaox
5F	Dpan
60	PDSxa
61	DSPDSaoxxn
62	DSPDoax
63	SDPnox
64	SDPSoax
65	DSPnox
66	DSx
67	SDPSonox
68	DSPDSonoxxn
69	PDSxxn
6A	DPSax
6B	PSDPSoaxxn

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(suite)*

Code ROP (hexadécimal)	Polonaise inversée
6C	SDPax
6D	PDSPDoaxxn
6E	SDPSnoax
6F	PDSxnan
70	PDSana
71	SSDxPDxaxn
72	SDPSxox
73	SDPnoan
74	DSPDxox
75	DSPnoan
76	SDPSnaox
77	DSan
78	PDSax
79	DSPDSoaxxn
7A	DPSDnoax
7B	SDPxnan
7C	SPDSnoax
7D	DPSxnan
7E	SPxDSxo
7F	DPSaan
80	DPSaa
81	SPxDSxon
82	DPSxna
83	SPDSnoaxn
84	SDPxna
85	PDSPnoaxn
86	DSPDSoaxx
87	PDSaxn
88	Dsa
89	SDPSnaoxn
8A	DSPnoa
8B	DSPDxoxn
8C	SDPnoa
8D	SDPSxoxn
8E	SSDxPDxax
8F	PDSanan
90	PDSxna
91	SDPSnoaxn
92	DPSDPoaxx
93	SPDaxn

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(suite)*

Code ROP (hexadécimal)	Polonaise inversée
94	PSDPSoaxx
95	DPSaxn
96	DPSxx
97	PSDPSonoxx
98	SDPSonoxn
99	DSxn
9A	DPSnax
9B	SDPSoaxn
9C	SPDnax
9D	DSPDoaxn
9E	DSPDSaoxx
9F	PDSxan
A0	Dpa
A1	PDSPnaoxn
A2	DPSnoa
A3	DPSDxoxn
A4	PDSPonoxn
A5	PDxn
A6	DSPnax
A7	PDSPoaxn
A8	DPSoa
A9	DPSoxn
AA	D
AB	DPSono
AC	SPDSxax
AD	DPSDaoxn
AE	DSPnao
AF	Dpno
B0	PDSnoa
B1	PDSPxoxn
B2	SSPxDSxox
B3	SDPanan
B4	PSDnax
B5	DPSDoaxn
B6	DPSDPaoxx
B7	SDPxan
B8	PSDPxax
B9	DSPDaoxn
BA	DPSnao
BB	Dsno

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(suite)*

Code ROP (hexadécimal)	Polonaise inversée
BC	SPDSanax
BD	SDxPDxan
BE	DPSxo
BF	DPSano
C0	Psa
C1	SPDSnaoxn
C2	SPDSonoxn
C3	PSxn
C4	SPDnoa
C5	SPDSxoxn
C6	SDPnax
C7	PSDPoaxn
C8	SDPoa
C9	SPDoxn
CA	DPSDxax
CB	SPDSaoxn
CC	S
CD	SDPono
CE	SDPnao
CF	Spno
D0	PSDnoa
D1	PSDPxoxn
D2	PDSnax
D3	SPDSoaxn
D4	SSPxPDxax
D5	DPSanan
D6	PSDPSaoxx
D7	DPSxan
D8	PDSPxax
D9	SDPSaoxn
DA	DPSDanax
DB	SPxDSxan
DC	SPDnao
DD	Sdno
DE	SDPxo
DF	SDPano
E0	PDSoa
E1	PDSoxn
E2	DSPDxax
E3	PSDPaoxn

Tableau 8-81/T.128 – Définition de Three-Way ROPs *(fin)*

Code ROP (hexadécimal)	Polonaise inversée
E4	SDPSxax
E5	PDSPaoxn
E6	SDPSanax
E7	SPxPDxan
E8	SSPxDSxax
E9	DSPDSanaxxn
EA	DPSao
EB	DPSxno
EC	SDPao
ED	SDPxno
EE	Dso
EF	SDPnoo
F0	P
F1	PDSono
F2	PDSnao
F3	PSno
F4	PSDnao
F5	PDno
F6	PDSxo
F7	PDSano
F8	PDSao
F9	PDSxno
FA	Dpo
FB	DPSnoo
FC	Pso
FD	PSDnoo
FE	DPSoo
FF	1

8.16.21 Opérations point-à-point binaires (*Two-Way ROPs)*

Les *two-way ROPs* sont des opérations point-à-point binaires utilisées par les ordres suivants:

- *Rectangle* voir 8.16.14.
- *Line* voir 8.16.16.

Les codes d'opérations point-à-point binaires définissent la combinaison des bits dans un stylet ou une brosse avec les bits d'une phototrame destination. Ce sont des opérations bit-à-bit agissant séparément sur des bits sans aucune interprétation de couleur, où les bits peuvent faire partie d'indices de palettes ou de valeurs directes de couleurs.

Ce qui suit représente les deux opérandes utilisés dans ces opérations.

- D Phototrame destination
- P Brosse (ou brosse)

Ce qui suit représente les opérateurs booléens utilisés dans ces opérations.

- a ET bit-à-bit
- n NON bit-à-bit (inverse)
- o OU bit-à-bit
- x OU exclusif bit-à-bit (XOR)

Toutes les combinaisons d'opérateurs booléens peuvent être représentées en notation Polonaise Inversée. Par exemple, l'opération suivante remplace les bits dans la phototrame destination avec un OU bit-à-bit effectué sur les bits de la phototrame destination et de la brosse/stylet:

DPo

Chaque code d'opération point-à-point binaire est une valeur d'octet simple représentant le résultat de l'opération booléenne sur les valeurs de brosse/stylet et destination prédéfinies qui ensuite est incrémenté de 1. Par exemple, les codes d'opération point-à-point des opérations DPo et DPan sont décrits dans la liste suivante:

Valeurs sources		**Valeurs générées**	
P	**D**	**DPo**	**DPan**
0	0	0	1
1	1	1	1
1	0	1	1
1	1	1	0
		⇒0x0E + 1	⇒0x07 + 1

Dans ce cas, DPo a le code d'opération point-à-point 0x0F (en lisant les valeurs binaires générées comme un octet ayant le bit le moins significatif au sommet – et ensuite en incrémentant). DPan a le code d'opération point-à-point 0x08.

Il faut noter que comme les opérations point-à-point agissent directement sur les valeurs binaires, sans interpréter la couleur, les résultats des opérations point-à-point peuvent délivrer des effets de couleurs imprévisibles lors de leur application sur différentes ASCE. Ceci peut être plus particulièrement notable pour les opérations point-à-point qui référencent une opérande destination, telle que l'inversion de destination (c'est-à-dire, le code Dn ou ROP 0x6 – Voir ci-dessous) qui est souvent utilisé pour implémenter des effets de marquage). Si une application hébergée utilise une opération point-à-point de ce type, là où l'ASCE hôte fonctionne sur un terminal utilisant une profondeur de couleur et/ou un modèle de couleur local différent et là où l'opération locale est envoyée comme un ordre, alors le résultat sur la couleur peut être très différent entre celle de l'ASCE hôte et celles des autres ASCE. Voir 8.16.20 pour plus d'information.

La liste complète des 16 codes d'opération point-à-point binaires est décrite dans le Tableau 8-82, avec la définition Polonaise Inversée pour chaque code[6].

[6] Il existe des épellations alternatives de Polonaise Inversée d'un même code d'opération, si bien qu'une épellation particulière pouvant ne pas être dans le tableau, une forme équivalente sera présente. Par exemple, DPx est équivalente à PDx.

Tableau 8-82/T.128 – Définition de Two-Way ROPs

Code ROP (hexadécimal)	Polonaise inversée
01	0
02	DPon
03	DPna
04	Pn
05	PDna
06	Dn
07	DPx
08	DPan
09	DPa
0A	DPxn
0B	D
0C	DPno
0D	P
0E	PDno
0F	DPo
10	1

8.16.22 Brosse (*Brush)*

Les *Brushes* (appelées aussi *patterns*) sont utilisées pour peindre l'intérieur de zones. Elles sont utilisées dans les ordres suivants:

- *Pattern Blt* voir 8.16.5.
- *Memory Three Way Blt* voir 8.16.10.
- *Frame* voir 8.16.13.
- *Rectangle* voir 8.16.14.

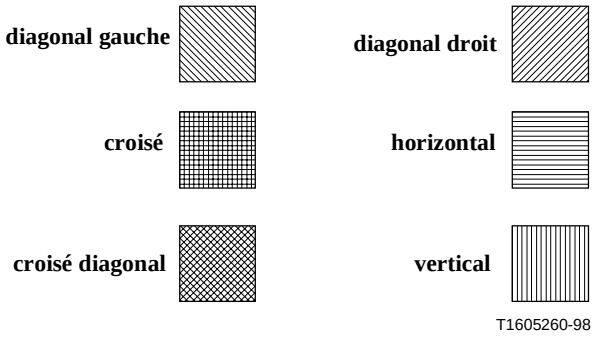
Les paramètres de brosse sont tributaires du codage de paramètre d'ordre. Voir 8.16.3.3 pour plus d'information. Voir le Tableau 8-83.

Tableau 8-83/T.128 – Définition de Brush

Paramètre	Description
originX	ce paramètre est ignoré là où les type de brosse est *solid* ou *null*. Lorsque le type de brosse est *hatch* ou *pattern*, ce paramètre est la coordonnée X de l'origine de la brosse.
originY	ce paramètre est ignoré là où les type de brosse est *solid* ou *null*. Lorsque le type de brosse est *hatch* ou *pattern*, ce paramètre est la coordonnée Y de l'origine de la brosse.
style	ce paramètre indique le style à utiliser pour cette brosse. Les valeurs autorisées sont les suivantes: • *Solid brush* l'intérieur de la zone est peint en utilisant la couleur de fond de l'ordre et les ROP2 et ROP3 dans l'ordre; • *Null brush* la brosse n'a aucun effet sur la teinte de l'intérieur de la zone; • *Hatched brush* l'intérieur de la zone est peint en utilisant le style de hachage de brosse avec les couleurs de fond et de surface relativement à l'origine de la brosse et en utilisant les ROP2 et ROP3 dans l'ordre;
	• *Pattern brush* l'intérieur de la zone est peint en utilisant le modèle de brosse avec les couleurs de fond et de surface relativement à l'origine de la brosse et en utilisant les ROP2 et ROP3 dans l'ordre.
hatch	là où le paramètre de style indique une brosse hachée, ce paramètre indique le style de hachage à utiliser pour cette brosse. Les valeurs autorisées sont les suivantes: • *Horizontal hatch* • *Vertical hatch* • *Forward Diagonal hatch* • *Backward Diagonal hatch* • *Cross hatch* • *Diagonal Cross hatch* là où le paramètre de style indique un modèle de brosse, ce paramètre est le premier octet du modèle.
pattern	là où le paramètre de style indique un modèle de brosse, ce paramètre contient les octets deux à huit du modèle.
nonStandardBrushParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres de brosse hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Le Tableau 8-84 décrit comment les styles de hachage de brosse apparaissent lors de leur utilisation lorsqu'on veut peindre l'intérieur d'un rectangle.

Tableau 8-84/T.128 – Motifs de hachurage de brosse



8.16.22.1 Attributs de brosse

Un attribut de brosse est composé d'une phototrame bicolore de 8 pixels sur 8, où les bits à 1 sont tracés dans la couleur de fond et ceux à 0 dans la couleur de surface.

Les bits du modèle sont envoyés en tant que 8 octets de données de phototrame non compressée, arrangée de telle sorte que le pixel x, y est dans l'octet y + 1 occupant la position de bit (x mod 8). Le premier octet est envoyé dans le paramètre de hachage et les octets deux à huit dans le paramètre modèle.

8.16.22.2 Origine de brosse

Lorsqu'une brosse hachée ou modélisée est utilisée pour peindre l'intérieur d'une zone, la brosse est répliquée pour remplir la zone selon l'origine de la brosse. L'origine de la brosse définit la position du pixel haut gauche de la brosse sur le moniteur virtuel à partir duquel la réplication de la brosse doit commencer. Le processus de réplication de la brosse peut être résumé comme suit:

- positionne le pixel haut gauche de la brosse à l'origine de la brosse sur le moniteur virtuel;
- copie la brosse horizontalement et verticalement sur le moniteur virtuel, avec une troncature correspondant à l'intérieur de la zone.

8.16.23 Stylet *(Pen)*

Les stylets sont utilisés pour tracer des lignes. Ils sont utilisés dans les ordres suivants:

- *Rectangle*: voir 8.16.14.
- *Line*: voir 8.16.16.

Le protocole AS supporte les types de stylets suivants:

- plein
- tireté
- pointillé
- trait-point
- trait-point-point
- transparent.

Les paramètres de stylet sont tributaires du codage de paramètre d'ordre. Voir 8.16.3.3 pour plus d'information. Voir le Tableau 8-85.

Tableau 8-85/T.128 – Définition de Pen

Paramètre	Description
style	ce paramètre indique le style à utiliser pour ce stylet. Les valeurs autorisées sont les suivantes: • *Solid pen* • *Dashed pen* • *Dotted pen* • *Dash-Dot pen* • *Dash-Dot-Dot pen* • *Null pen*
width	ce paramètre indique la taille de stylet à utiliser. La valeur autorisée est 1.
color	ce paramètre est la couleur de stylet à utiliser.
nonStandardPenParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres de stylet hors norme autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Dans le cas d'un stylet plein, tous les pixels de la ligne sont dessinés en utilisant la couleur de stylet et le ROP2 référencés dans l'ordre.

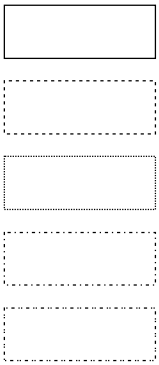
Dans le cas de stylets tireté, pointillé, tiret-point ou tiret-point-point, la ligne est dessinée en utilisant la séquence de pixels fond/surface déterminée par le style de stylet, où:

- les pixels de surface sont dessinés en utilisant la couleur de surface de stylet et le ROP2 référencés dans l'ordre;
- les pixels de fond ne sont pas tracés si le mélange de fond est *Transparent* (voir 8.16.24);
- les pixels de fonds sont tracés en utilisant la couleur de fond de l'ordre et le ROP2 si le mélange de fond est *Opaque* (voir 8.16.24).

Dans le cas d'un stylet *null*, aucune ligne de points ne sera tracée.

Le Tableau 8-86 décrit comment les nombreux styles de stylet apparaissent lorsqu'ils sont utilisés pour tracer une limite de rectangle.

Tableau 8-86/T.128 – Styles de Pen



8.16.24 Mélange de fond

Le mélange de fond est utilisé pour les ordres suivants:

- *Text*: voir 8.16.11.
- *Extended Text*: voir 8.16.12.
- *Rectangle*: voir 8.16.14.
- *Line*: voir 8.16.16.

Dans le cas des ordres *Text* et *Extended Text*, le mélange de fond détermine si les pixels de la couleur de fond sont dessinés ou non dans les cellules des caractères de texte. Là où le mélange de fond est *Transparent*, les pixels de fond de cellule de caractère ne sont pas dessinés. Là où le mélange de fond est *Opaque* (et pour l'ordre *Extended Text*, un rectangle *Opaque* n'est pas requis), les pixels de fond de cellule de caractère sont dessinés dans la couleur de fond de l'ordre.

Dans le cas des ordres *Rectangle* et *Line*, le mélange de fond détermine si les pixels correspondant au stylet levé sont dessinés pour les styles tiretés, pointillés, trait-point, trait-point-point. Là où le mélange de fond est *Transparent*, les pixels correspondant au style levé ne sont pas dessinés. Là où le mélange de fond est *Opaque*, les pixels de fond de cellule de caractère sont dessinés en utilisant la couleur de fond de l'ordre et le paramètre ROP2. Voir 8.16.23 pour plus d'information sur les stylets et 8.16.21 pour plus d'information sur les codes d'opération point-à-point binaires.

Dans le cas de l'ordre *Rectangle*, le mélange de fond détermine si les pixels à l'intérieur du fond sont dessinés lors de l'utilisation d'une brosse hachée. Dans ce cas, là où le mélange de fond est *Transparent*, les pixels de fond ne sont pas dessinés, mais là où le mélange de fond est *Opaque*, les pixels de fond sont dessinés en utilisant la couleur de fond de l'ordre et le paramètre ROP2. Le mélange de fond n'affecte pas l'intérieur des pixels de fond pour d'autres styles de brosse – pour les brosses pleines, les pixels de fond ne sont pas appliqués; pour les brosses modèles, les pixels de fond sont toujours dessinés.

Le protocole AS supporte les types de mélange de fond suivants. Voir le Tableau 8-87.

Tableau 8-87/T.128 – Types de mélange de fond

Type de mélange de fond	Définition
Opaque	les pixels de fond sont dessinés.
Transparent	les pixels de fond ne sont pas dessinés.

8.17 Mises à jour de phototrame

Une ASCE envoie des mises à jour de phototrame à toutes les ASCE dans la conférence en envoyant un *UpdatePDU* contenant des données de phototrame de la manière indiquée dans le Tableau 6-3. Le contenu de *UpdatePDU* contenant la donnée de phototrame est décrit dans le Tableau 8-88.

Une ASCE enverra des données de phototrame non compressées seulement là où la capacité négociée *Bitmap.sendingBitsPerPixel* vaut 1, 4 ou 8. Une ASCE enverra des données de phototrame compressées seulement là où la capacité négociée *Bitmap.sendingBitsPerPixel* vaut 4 ou 8 et la capacité négociée *Bitmap.bitmapCompressionFlag* vaut TRUE. Les données de phototrame compressées ne seront pas envoyées là où la capacité négociée *Bitmap.sendingBitsPerPixel* vaut 1.

Sur réception d'un *UpdatePDU* contenant une donnée de phototrame, l'ASCE réceptrice (après une éventuelle décompression) effectue une copie à partir du coin haut gauche de la phototrame vers les coins haut gauche du rectangle destination sur le moniteur virtuel pour ce qui concerne la largeur et la hauteur de la destination. Là où la largeur et/ou la hauteur de la phototrame sont supérieures à celles de la destination, l'ASCE réceptrice tronquera la phototrame aux dimensions du rectangle destination. L'ASCE interprétera les valeurs de pixels de la phototrame en utilisant le dernier ASPDU *UpdatePDU (Palette)*. Voir 8.15 pour plus d'information sur les palettes.

Tableau 8-88/T.128 – UpdatePDU (Bitmap)

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
destLeft	ce paramètre est la coordonnée X gauche de moniteur virtuel du rectangle destination de la donnée phototrame.
destTop	ce paramètre est la coordonnée Y haute de moniteur virtuel du rectangle destination de la donnée phototrame.
destRight	ce paramètre est la coordonnée X droite de moniteur virtuel du rectangle destination de la donnée phototrame.
destBottom	ce paramètre est la coordonnée Y basse de moniteur virtuel du rectangle destination de la donnée phototrame.
width	ce paramètre est la largeur en pixels de la donnée de phototrame. La largeur de la phototrame sera supérieure ou égale à la largeur du rectangle destination (voir ci-dessus).
height	ce paramètre est la hauteur en pixels de la donnée de phototrame. La hauteur de la phototrame sera supérieure ou égale à la hauteur du rectangle destination (voir ci-dessus).
bitsPerPixel	ce paramètre est le nombre de bits par pixels. Ce paramètre sera identique à la valeur négociée *Bitmap.sendingBitsPerPixel*.Voir 8.2.4 pour plus d'information.
compressedFlag	ce paramètre indique si la donnée de phototrame (voir ci-dessous) est compressée ou non. Une valeur à TRUE indique que la donnée est compressée et une valeur à FALSE qu'elle ne l'est pas.
bitmapData	ce paramètre est la donnée de phototrame. La phototrame peut ne pas être compressée (Voir 8.17.1) ou compressée (Voir 8.17.2).
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.17.1 Données de phototrame non compressées

Les données de phototrame non compressées représentent une phototrame comme une série de rangées où la rangée zéro commence à la coordonnée Y du pixel le plus haut, à savoir en bas à gauche.

A l'intérieur d'une rangée, les valeurs de pixel sont empaquetées dans des octets, en commençant à partir du pixel le plus à gauche.

- Pour les données de phototrame à 1 bit par pixel, chaque octet contient huit pixels, avec le pixel le plus à gauche dans le bit le plus significatif.
- Pour les données de phototrame à 4 bits par pixel, chaque octet contient deux pixels, avec le pixel le plus à gauche dans les 4 bits les plus significatifs.
- Pour les données de phototrame à 8 bits par pixel, chaque octet contient une valeur de pixel.

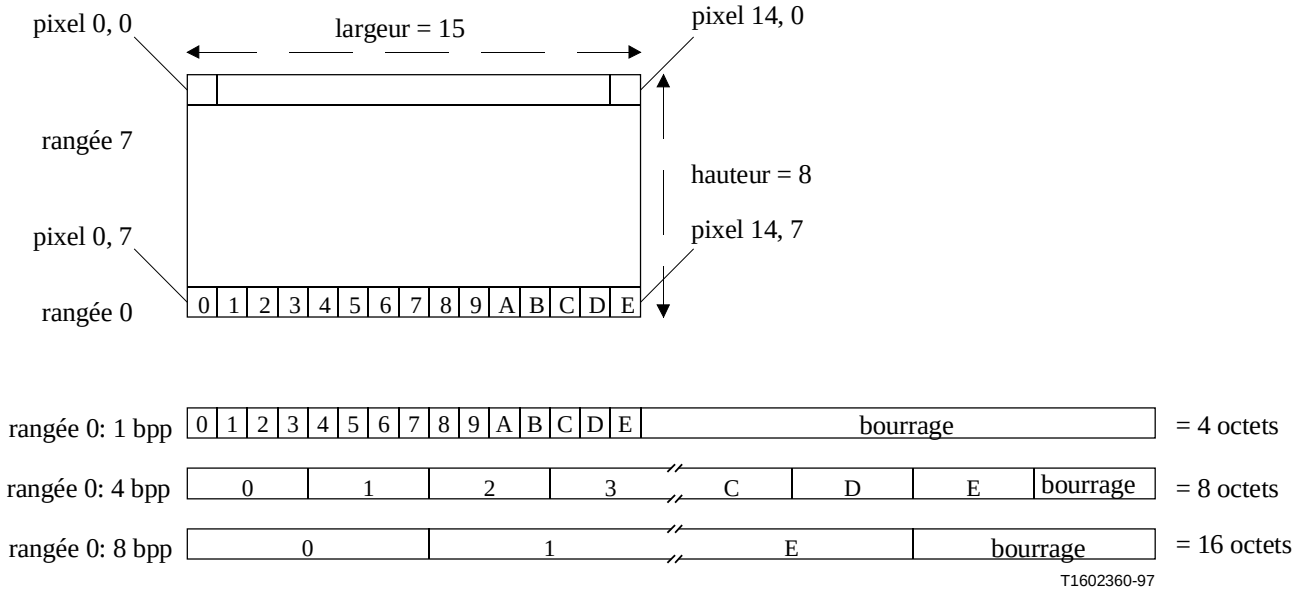
Chaque rangée de donnée de phototrame est remplie selon une frontière de quatre octets. Ceci signifie que la taille d'une rangée est égale à:

((((largeur-phototrame-en-pixels * bit-par-pixel) + 31) div 32) * 4) octets

La taille totale de la phototrame est:

(hauteur-phototrame * octets-par-ligne) octets

Le diagramme suivant illustre le diagramme des pixels dans une donnée de phototrame non compressée pour la rangée 0 d'une phototrame de 15 sur 8, respectivement à 1, 4 et 8 bits par point.



8.17.2 Données de phototrame compressées

Dans le cas des données de phototrame compressées, l'information réelle de la phototrame compressée est précédée par l'en-tête de la phototrame compressée décrite dans le Tableau 8-89.

Tableau 8-89/T.128 – En-tête de phototrame compressée

Paramètre	Description
mainBodySize	ce paramètre est la longueur en octets de la donnée de phototrame compressée (après compression).
rowSize	ce paramètre est la longueur en octets d'une simple rangée de la donnée de phototrame non compressée.
uncompressedSize	ce paramètre est la longueur en octets de la donnée de phototrame non compressée.
compressedBitmapData	ce paramètre est la donnée de phototrame compressée.

La compression de phototrame du protocole AS est orientée vers la compression de données de gestionnaire de fenêtre, où les plages peuvent apparaître à la fois en X et en Y. Le protocole AS représente une phototrame compressée sous la forme d'une séquence de codes de compression, où une phototrame particulière peut être représentée par des séquences différentes de codes valides.

Les codes de compression sont de deux types:

- les codes de différence sont des plages bi-dimensionnelles où une série de valeurs de pixels est déterminée par le code de compression et les valeurs correspondantes de pixels dans les rangées précédentes. Les codes de compression *Fill, Mix, FillOrMix, SetMix_Mix, SetMix_FillOrMix, FillOrMix_1* et *FillOrMix_2* (voir le Tableau 8-90) sont des codes de différence définis comme une ou plusieurs opérations *Fill* et/ou *Mix*;
  - l'opération *Fill* affecte le pixel courant dans la rangée courante au pixel correspondant dans la rangée précédente. C'est une opération de copie directe;
  - l'opération *Mix* affecte au pixel courant dans la rangée courante le résultat du OU exclusif entre la valeur *Mix* courante et la valeur du pixel correspondant de la rangée précédente. La valeur *Mix* est mise par défaut à 0xFF (qui est affecté au départ d'une passe de compression/décompression dans chaque nouvelle phototrame) et peut être mise à jour par les codes *SetMix_Mix* et *SetMix_FillOrMix*;

    Puisque les codes de différence référencent la rangée de phototrame précédente, des opérations spéciales ont été définies pour la première rangée (voir les définitions individuelles de code d'opération dans le Tableau 8-90);
- les codes de couleur sont à passes linéaires où une série de valeurs de pixel est déterminée par référence au seul code de compression. Les codes de compression *Color, Copy, CopyPacked, Bicolor, Black* et *White* (voir le Tableau 8-90) sont des codes de couleurs.

Un code de compression est composé d'un identificateur de code, suivi d'un champ de longueur optionnel, suivi lui-même d'une donnée associée optionnelle dépendante du code. Un code de compression particulier peut avoir de multiples codages, ceci dépend de la taille du champ longueur. Voir le Tableau 8-90 pour plus d'information sur les codages en rapport avec la longueur.

Le champ longueur peut être ensuite codé d'une manière dépendante du code. Voir le Tableau 8-91 pour plus d'information sur les codages de longueur. Lorsqu'elle est présente, la longueur (après un éventuel décodage) peut spécifier le nombre d'octets de données associées et le nombre de répétitions requises.

Une ASCE réceptrice interprète une séquence de codes de compression de phototrame comprise à l'intérieur d'un *Update PDU (Bitmap)* pour générer une phototrame décompressée (selon le format décrit au 8.17.1) comme suit:

1) affecter la position de pixel initiale à (bas, gauche); affecter la valeur *Mix* initiale à 0xFF;
2) décoder le prochain code et calculer sa valeur de longueur (n);

3) effectuer l'opération de code définie selon la répétition définie (en rebouclant sur la prochaine rangée si nécessaire);

4) mettre à jour la position de pixel (en rebouclant sur la prochaine rangée si nécessaire);

5) répéter l'étape 2 à 4 jusqu'au traitement de tous les codes.

Par exemple si la position courante de pixel est le pixel 60 dans la rangée 6 et que le prochain code est *Fill*, avec pour codage 3b<0x0>,5b<len = 8>, l'ASCE réceptrice affecte la valeur des pixels 60..67 avec les valeurs dans les pixels 60..67 de la rangée 5 et met à jour la position de point à la valeur 68.

La où la phototrame d'origine (d'avant la compression) contient du bourrage (voir 8.17.1), l'ASCE émettrice traitera le bourrage comme des données de pixel rentrant dans la compression. Ceci permet à l'ASCE réceptrice de traiter la séquence de code de compression en faisant référence à des séries linéaires de pixels (indépendamment de toute considération de bourrage).

- Le Tableau 8-90 résume les codages, types de longueur de codage, opérations et répétitions pour chaque code défini.
- Le Tableau 8-91 résume le codage pour chaque type de longueur.
- Le Tableau 8-92 résume la notation utilisée dans les Tableaux 8-90 et 8-91.

Les données de phototrame compressées sont empaquetées dans le paramètre *compressedBitmapData* en remplissant les bits successivement dans chaque octet en commençant par le bit le plus significatif de chaque champ et en remplissant vers le bit le moins significatif. Là où les champs sont inférieurs à un octet, les champs suivants sont remplis en commençant au prochain bit le plus significatif. Là où un champ simple occupe deux octets, les octets sont remplis par ordre d'importance croissante, avec l'octet d'ordre le plus élevé, ou le plus significatif, placé dans le second octet[7].

Tableau 8-90/T.128 – Codes de compression de phototrame

Code	Codage	Lon-gueur Type	Opération	Répéter
Fill 9 (Note 1)	3b<0x0>,5b<len> 8b<0x00>,8b<len> 8b<0xF0>,16b<len>	A	rangée 1: $pixel^i \Leftarrow 0$ rangée 2..H: $pixel^i \Leftarrow pixel^{i,r-1}$	i = 0..n–1
Mix	3b<0x1>,5b<len> 8b<0x20>,8b<len> 8b<0xF1>,16b<len>	A	rangée 1: $pixel^i \Leftarrow mix$ rangée 2..H: $pixel^i \Leftarrow pixel^{i,r-1} \wedge mix$	i = 0..n–1
FillOrMix	3b<0x2>,5b<len>,(n+7)/8<mask> 8b<0x40>,8b<len>,(n+7)/8<mask> 8b<0xF2>,16b<len>,(n+7)/8<mask>	A8	rangée 1: $pixel^i \Leftarrow 0$ or $pixel^i \Leftarrow mix$ rangée 2..H: $pixel^i \Leftarrow pixel^{i,r-1}$ or $pixel^i \Leftarrow pixel^{i,r-1} \wedge mix$	i = 0..n–1 pour (n+7)/8 masque
Color	3b<0x3>,5b<len>,<color> 8b<0x60>,8b<len>,<color> 8b<0xF3>,16b<len>,<color>	A	$pixel^i \Leftarrow color$	i = 0..n–1
Copy	3b<0x4>,5b<len>,n<color> 8b<0x80>,8b<len>,n<color> 8b<0xF4>,16b<len>,n<color>	A	$pixel^i \Leftarrow color^i$	i = 0..n–1

[7] Alors que deux champs d'octets ont des octets remplis selon un ordre d'importance croissant, les bits constitutifs dans chaque octet continuent à être remplis du bit le plus significatif de chaque champ vers le moins significatif.

Tableau 8-90/T.128 – Codes de compression de phototrame *(fin)*

Code	Codage	Lon-gueur Type	Opération	Répéter
CopyPacked (Note 2)	3b<0x5>,5b<len>,n/2<packed_color> 8b<0xA0>,8b<len>,n/2<packed_color> 8b<0xF5>,16b<len>,n/2<packed_color>	A	$pixel^{i*2}$ ⇐ $packed\_color^{i}$(high), $pixel^{(i*2)+1}$ ⇐ $packed\_color^{i}$(low)	i = 0..(n/2)–1
SetMix_Mix	4b<0xC>,4b<len>,<color> 8b<0xC0>,8b<len>,<color> 8b<0xF6>,16b<len>,<color>	B	mix ⇐ color, rangée 1: $pixel^{i}$ ⇐ mix rangée 2..H: $pixel^{i}$ ⇐ $pixel^{i,r-1}$∧mix	i = 0..n–1
SetMix_FillOrMix	4b<0xD>,4b<len>,<color>,(n+7)/8<mask> 8b<0xD0>,8b<len>,<color>,(n+7)/8<mask> 8b<0xF7>,16b<len>,<color>,(n+7)/8<mask>	B8	mix ⇐ color, rangée 1: $pixel^{i}$ ⇐ 0 or $pixel^{i}$ ⇐ mix rangée 2..H: $pixel^{i}$ ⇐ $pixel^{i,r-1}$ or $pixel^{i}$ ⇐ $pixel^{i,r-1}$∧mix	i = 0..n–1 pour (n+7)/8 masque
Bicolor	4b<0xE>,4b<len>,<color1>,<color2> 8b<0xE0>,8b<len>,<color1>,<color2> 8b<0xF8>,16b<len>,<color1>,<color2>	B	$pixel^{i*2}$ ⇐ color1, $pixel^{(i*2)+1}$ ⇐ color2	i = 0..n–1
FillOrMix_1	8b<0xF9>{,<mask=0x03>}	C	rangée 1: $pixel^{i}$ ⇐ 0 or $pixel^{i}$ ⇐ mix rangée 2..H: $pixel^{i}$ ⇐ $pixel^{i,r-1}$ or $pixel^{i}$ ⇐ $pixel^{i,r-1}$∧mix	i = 0..7 pour 1 masque
FillOrMix_2	8b<0xFA>{,<mask=0x05>}	C	rangée 1: $pixel^{i}$ ⇐ 0 or $pixel^{i}$ ⇐ mix rangée 2..H: $pixel^{i}$ ⇐ $pixel^{i,r-1}$ or $pixel^{i}$ ⇐ $pixel^{i,r-1}$∧mix	i = 0..7 pour 1 masque
White	8b<0xFD>	C	pixel ⇐ 0xF (4bpp)/0xFF (8bpp)	1
Black	8b<0xFE>	C	pixel ⇐ 0	1

NOTE 1 – Là où un *Fill* suit immédiatement un autre *Fill* (c'est-à-dire qu'il existe des codes *Fill* consécutifs), l'ASCE réceptrice insérera un code unique *Fill* ayant pour codage 3b<0x1>, 5b<len=1>.

NOTE 2 – Une ASCE enverra des données de phototrame compressées contenant le code *CopyPacked* seulement là où la valeur de la capacité négociées *Bitmap.sendingBitsPerPixel* vaut quatre et lorsqu'elle émet des données de phototrame compressées à 4 bits par pixel. Voir 8.2.4 pour plus d'information.

Tableau 8-91/T.128 – Codes de compression de phototrame: codage de longueur

Type de longueur	Intervalle		Calcul
A	5b:	1..31	n = len
	8b:	0..255	n = len + 32
	16b:	288..65535	n = len
B	4b:	1..15	n = len
	8b:	0..255	n = len + 16
	16b:	272..65535	n = len
A8	5b:	1..31	n = len * 8
	8b:	0..254	n = len + 1
	16b:	256..65535	n = len
B8	4b:	1..31	n = len * 8
	8b:	0..254	n = len + 1
	16b:	256..65535	n = len
C	n/a		n = 1

Tableau 8-92/T.128 – Codes de compression de phototrame: légende

Notation	Description
nb<f>	le champ f a une taille de n bits. Par exemple, *Mix* peut être codé avec un champ initial de 3 bits de valeur 0x1.
<len>	le champ contient une longueur du type spécifié dans la colonne *Length*. La valeur de longueur extraite (calculée selon le codage du Tableau 8-91) a pour référence n dans le reste du codage et est typiquement utilisée pour spécifier le nombre d'octets suivants de masque, de couleur ou de champs de couleur paquetisés et pour générer la répétition.
n<f>	il existe n occurrences de type de champ f.

<mask>	le champ contient un masque. Un masque est un octet permettant de coder 8 pixels: un sur chaque bit. Une valeur de bit à 1 indique que la valeur *Mix* devrait être utilisée (voir ci-dessous) et une valeur à 0 indique que la valeur *Fill* devrait être utilisée. La correspondance entre pixels et bits se fait comme suit:
	Pixels (adresse du pixel le plus haut = 7) ⇒ bits (bit le plus significatif = 7)
	`0 1 2 3 4 5 6 7` `7 6 5 4 3 2 1 0`
	`a b c d e f g h` `h f d b g e c a`
	par exemple un masque de valeur 0xAA (c'est-à-dire les bits 10101010) est étendu à la série suivante: *Fill, Fill, Mix, Mix, Fill, Fill, Mix, Mix*.

<color>	le champ contient un octet d'information de couleur. Pour les données de phototrame à 4 bits par pixel, les quatre bits les plus significatifs seront ignorés. Le codage de compression de phototrame n'impose aucune interprétation sur les valeurs de couleur.
<packed_color>	le champ contient un octet paquetisé de valeur de couleur. La valeur de couleur paquetisée est seulement autorisée pour les données de phototrame à 4 bits par pixel, là où elle comprend deux valeurs de couleurs à 4 bits, où la première couleur est dans les quatre bits les plus significatifs et la seconde dans les quatre bits les moins significatifs. Le codage de compression de phototrame n'impose aucune interprétation sur les valeurs de couleur.
<f1>,<f2>	le champ f1 précède le champ f2. Par exemple, *Fill* peut être codé par un champ sur 3 bits de valeur 0x0, suivi d'un champ sur 5 bits donnant la longueur.
{,<f=value>}	le champ f contenant *value* est induit et non pas inclus par l'ASCE émettrice. Par exemple, le code *FillOrMix_1* a un octet initial égal à 0xF9 induisant une valeur de masque de 0x03.
p ⇐ e	l'expression e est assignée au pixel p. Par exemple, l'opération pour *Black* est pixel ⇐ black, remplaçant le pixel courant avec la valeur de couleur Noir (voir ci-dessous).
p ⇐ e1∧e2	le OU exclusif (XOR) point à point des deux expressions est assigné au pixel p.
op1, op2	effectue l'opération op1 et ensuite l'opération op2.
op1 or op2	effectue l'opération op1 pour les bits de masque correspondants égaux à zéro et l'opération op2 pour les bits de masque correspondants égaux à 1 (voir ci-dessus).
rangée 1: op1 or rangée 2..H: op2	effectue l'opération op1 pour les pixels de la première rangée de la phototrame destination et l'opération op2 pour les pixels des autres rangées.
$pixel^i$	le pixel courant dans la rangée courante de la phototrame destination. Là où le code possède une répétition, i prend les valeurs indiquées dans la colonne *repeat*.

Tableau 8-92/T.128 – Codes de compression de phototrame: légende *(fin)*

Notation	Description
$pixel^{i,r-1}$	le pixel correspondant dans la rangée précédente de la phototrame destination. Là où le code définit une répétition, i prend les valeurs indiquées dans la colonne *repeat*. Par exemple, si le pixel courant est le point 35 dans la rangée 6, $pixel^{i,r-1}$ est le point 35 dans la rangée 5.
mix	la valeur *Mix* courante. La valeur *Mix* est réinitialisée à 0xFF au début de la décompression d'un nouvel ordre *UpdatePDU* (*Bitmap*) ou *Cache Bitmap* (voir 8.16.7) et est ensuite affectée par les codes *SetMix_Mix* et/ou *SetMix_FillOrMix*.

8.18 Input (*Entrée*)

La manipulation d'entrée est étroitement intégrée avec le mode présidé (s'il est actif dans la conférence) et avec la politique de contrôle en vigueur dans la conférence. En fonction de la présidence et/ou de l'état de contrôle, l'ASCE peut ou non posséder le droit de fournir une entrée aux fenêtres hébergées ou reflets dans la conférence. Voir 8.19 pour plus d'information sur le mode présidé et 8.12 et 8.13 pour plus d'information sur les mécanismes de contrôle.

Une ASCE peut fournir des entrées aux ASCE homologues dans la conférence en envoyant un *InputPDU* contenant les événements d'entrée de la manière indiquée dans le Tableau 6-3. Le contenu de *InputPDU* est décrit dans le Tableau 8-93. Les événements d'entrée peuvent être de l'un des types suivants:

- événement de dispositif de pointage: voir 8.18.1;
- événement de clavier: voir 8.18.2;
- événement de synchronisation d'entrée: voir 8.18.6.

Tableau 8-93/T.128 – InputPDU

Paramètre	Description
ShareData Header	le *Share Data Header* est décrit au 8.3.
eventList	ce paramètre est une liste d'événements d'entrée (voir ci-dessous).
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.18.1 Evénements de dispositif de pointage

En fonction de la présidence et/ou de l'état de contrôle, l'ASCE peut ou non mettre en file d'attente un événement de pointage dans un *InputPDU* lorsque apparaît un événement local de dispositif de pointage. Une ASCE réceptrice traitera les événements de dispositif de pointage conformément aux politiques et aux états de contrôle locaux courants. L'événement de dispositif de pointage est décrit dans le Tableau 8-94.

Les événements de dispositif de pointage sont modélisés selon un dispositif à trois boutons où:

- le bouton 1 est le bouton logique gauche (utilisé habituellement pour la sélection);
- le bouton 2 est le bouton logique droit;
- le bouton 3 est le bouton logique du milieu.

Certains terminaux permettent de réaffecter le mappage entre boutons logiques et physiques sous le contrôle de l'utilisateur ou d'un programme, de sorte que (par exemple) le bouton logique gauche (ou sélection) soit glissé vers le bouton physique droit. Là où une telle fonction est fournie, l'ASCE s'assurera que les événements de dispositifs de pointage sont envoyés sous la forme d'assignements logiques de boutons.

Les événements de dispositif de pointage contiennent la coordonnée de moniteur virtuel du point correspondant à l'apparition de l'événement. L'ASCE émettrice est responsable de la restriction de l'événement au seul moniteur virtuel. Les ASCE réceptrices ont à leur charge de faire en sorte que tous les événements de dispositif de pointage locaux générés à partir d'événements de dispositif de pointage reçus soient restreints au seul moniteur local.

Tableau 8-94/T.128 – Evénement de dispositif de pointage

Paramètre	Description
eventTime	ce paramètre est le temps local de l'ASCE en millisecondes correspondant à l'apparition de cet événement.
pointingDeviceFlags	ce paramètre est un ensemble d'indicateurs binaires identifiant et qualifiant l'événement de dispositif de pointage. Les valeurs d'indicateurs binaires définies sont les suivantes: • *Move* • *Button 1* • *Button 2* • *Button 3* • *Down* voir ci-dessous pour plus d'information sur l'interprétation des indicateurs d'événement de dispositif de pointage.
pointingDeviceX	ce paramètre est la coordonnée X de moniteur virtuel de l'événement de dispositif de pointage.
pointingDeviceY	ce paramètre est la coordonnée Y de moniteur virtuel de l'événement de dispositif de pointage.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Le Tableau 8-95 décrit l'usage des indicateurs de dispositif de pointage pour un dispositif à trois boutons. L'ASCE émettrice est responsable de l'affectation des indicateurs de dispositif de pointage dans les événements de dispositif de pointage. Les ASCE réceptrices sont responsables de l'interprétation des indicateurs de dispositif de pointage conformément aux caractéristiques locales du terminal.

Le protocole AS ne supporte pas les événements double-clic de dispositif de pointage. Là où un double-clic apparaît sur le terminal local, une ASCE enverra la séquence d'événements correspondante de dispositif de pointage *button down, button up, button down, button up* (par exemple pour un double-clic sur *Button 1*, l'ASCE enverra *Button 1 down, Button 1 up, Button 1 down, Button 1 up*), avec les temps d'événements locaux correspondants. Une ASCE réceptrice responsable du traitement de cette séquence d'événements de dispositif de pointage fera en sorte que là où les temps d'événement tombent dans l'intervalle double-clic du terminal local, ils soient

interprétés comme un double-clic. Ceci signifie que dans une conférence multipoint, là où une ASCE affiche des fenêtres reflets correspondant à des fenêtres hébergées sur de multiples ASCE hôtes et là où elle est *cooperating* et *in control*, alors l'intervalle double-clic sera celui de la fenêtre locale ou reflet active.

Tableau 8-95/T.128 – Indicateurs d'événement de dispositif de pointage à trois boutons

Evénement de dispositif de pointage	Combinaisons valides de *pointingDeviceFlags*
déplacement	déplacement
bouton gauche en bas	bouton 1 + enfoncé
bouton gauche en haut	bouton 1
bouton droit en bas	bouton 2 + enfoncé
bouton droit en haut	bouton 2
bouton milieu en bas	bouton 3 + enfoncé
bouton milieu en haut	bouton 3

8.18.2 Evénements de clavier

En fonction de la présidence et/ou de l'état de contrôle, l'ASCE peut ou non mettre en file d'attente un événement de clavier dans un *InputPDU* lorsque apparaît un événement local de clavier. Une ASCE réceptrice traitera les événements de clavier conformément aux politiques et aux états de contrôle locaux courants.

Les événements de clavier peuvent être représentés dans le protocole AS comme des codes ou des touches virtuelles (voir ci-dessous). L'événement de clavier est décrit dans le Tableau 8-96.

Tableau 8-96/T.128 – Evénement de clavier

Paramètre	Description
eventTime	ce paramètre est le temps local de l'ASCE correspondant à l'apparition de l'événement.
keyboardFlags	ce paramètre est un ensemble d'indicateurs binaires identifiant et qualifiant l'événement de clavier. Les valeurs d'indicateurs binaires sont définies comme suit: • *Right* • *Quiet* • *Down* • *Release* les indicateurs binaires *Right* et *Quiet* sont valables uniquement pour les événements de touches virtuelles de clavier. Les indicateurs binaires *Down* et *Release* sont valables pour les événements de codes et de touches virtuelles de clavier. Voir ci-dessous pour plus d'information sur l'interprétation des indicateurs d'événements de clavier.

Tableau 8-96/T.128 – Evénement de clavier *(fin)*

Paramètre	Description
keyCode	pour les événements de codes de clavier, ce paramètre est un code de table de codage du protocole AS (voir 8.8.1). Pour les événements de touches virtuelles de clavier, ce paramètre est un code de touche virtuelle (voir 8.18.3 ci-dessous).
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

Une ASCE émettrice utilise des combinaisons d'indicateurs binaires *Down* et *Release* dans le paramètre *keyboardFlags* pour indiquer les actions de clavier suivantes.

- Relâchement de clé: les indicateurs binaires *Down* et *Release* sont positionnés.
- Enfoncement de clé: aucun des indicateurs binaires *Down* et *Release* n'est positionné.
- Répétition de clé: l'indicateur binaire *Down* est positionné.

Une ASCE réceptrice:

- traite les événements de touche virtuelle de clavier selon les états de basculement et de modification du clavier précédemment établis, mais n'interprète ni ne traduit la valeur du code de touche virtuelle;
- traite les événements de code de clavier reçus en générant la série appropriée d'actions locales requises pour produire le code sur le terminal local.

Là où une ASCE émettrice a le choix d'utiliser un code ou une touche virtuelle pour une fonction particulière (c'est-à-dire le code 0x41 du protocole AS et le code de touche virtuelle 0x41 correspondent tous les deux à A majuscule), elle devrait envoyer le code plutôt que la code de touche virtuelle. Ceci est préférable puisque les codes des tables de codage du protocole AS sont indépendants des états de basculement et/ou de modification de clavier, tandis que les codes de touches virtuelles ne le sont pas. Par exemple, sur les claviers des Macintosh et PC 101 américains, le caractère @ (Unicode 0x26, AROBA) correspond à shift-2 tandis que sur les claviers PC 101 anglais il correspond à shift-‘ (shift-quote) – et shift-2 est " (Unicode 0x22, GUILLEMET). Ainsi donc, si une ASCE fonctionnant sur un PC américain était dans une conférence où les ASCE homologues étaient un mélange de claviers américains et anglais et avaient envoyé des codes de touches virtuelles, alors le fait de taper Shift-2 localement– qui aurait produit la séquence d'événements de clavier de codes de touches virtuelles *VK_SHIFT-down, VK_2-down, VK_2-up, VK_SHIFT-up* – pourrait produire de manière variable @ et/ou " sur les autres terminaux. Par contraste, si l'ASCE envoyait des codes – qui auraient produit la séquence d'événements de clavier de codes 0x0026-*down*, 0x0026-*up*, chaque ASCE homologue pourrait entreprendre l'action locale appropriée pour produire le caractère @.

8.18.3 Codes de touches virtuelles

Les codes de touches virtuelles fournissent une méthode indépendante du matériel et du langage pour identifier des touches de clavier. Chaque code de touche virtuelle représente une touche unique de clavier et identifie aussi la raison de cette touche. Le Tableau 8-97 définit les codes de touches virtuelles supportés par le protocole AS. Les ASCE enverront seulement des événements de clavier *InputVirtualKey* dans les *InputPDUs* en utilisant les codes de touches virtuelles définis.

Sur certains terminaux, les touches peuvent être dupliquées (par exemple, *shift* est typiquement dupliquée à droite et à gauche) mais ne correspondent qu'à un seul code de touche virtuelle (VK_SHIFT = 0x10). Certaines applications sont sensibles au fait que les utilisateurs ont pressé la variante droite ou gauche de telles touches. Ainsi donc, là où une ASCE peut localement faire la distinction entre les variantes droite et gauche, elle devrait:

- positionner ou remettre à zéro l'indicateur binaire *Right* dans le paramètre *keyboardFlags* lors de l'envoi de tels événements de codes de touches virtuelles;
- vérifier l'indicateur binaire *Right* dans le paramètre *keyboardFlags* lors de la réception d'événements de codes de touches virtuelles et entreprendre l'action locale appropriée pour générer l'indication locale correspondante.

Là où une ASCE ne fait pas de distinction entre les variantes droite et gauche, alors elle devrait:

- ne jamais positionner ou remettre à zéro l'indicateur binaire *Right* dans le paramètre *keyboardFlags* lors de l'envoi de tels événements de codes de touches virtuelles;
- ne jamais vérifier l'indicateur binaire *Right* dans le paramètre *keyboardFlags* lors de la réception d'événements de codes de touches virtuelles et ainsi donc manipuler les événements ayant l'indicateur binaire *Right* positionné comme si l'événement référençait la variante gauche.

Tableau 8-97/T.128 – Codes de clés virtuelles

Nom	Valeur	Commentaire
VK_CANCEL	0x03	*break*
VK_BACK	0x08	
VK_TAB	0x09	
VK_CLEAR	0x0C	
VK_RETURN	0x0D	
VK_SHIFT	0x10	
VK_CONTROL	0x11	
VK_ALT	0x12	connue aussi sous le nom de Menu
VK_PAUSE	0x13	
VK_CAPITAL	0x14	
VK_ESCAPE	0x1B	
VK_SPACE	0x20	
VK_PRIOR	0x21	déplacement de page vers le haut
VK_NEXT	0x22	déplacement de page vers le bas
VK_END	0x23	
VK_HOME	0x24	
VK_LEFT	0x25	
VK_UP	0x26	
VK_RIGHT	0x27	
VK_DOWN	0x28	

Tableau 8-97/T.128 – Codes de clés virtuelles *(suite)*

Nom	Valeur	Commentaire
VK_SELECT	0x29	
VK_SNAPSHOT	0x2C	impression d'écran
VK_INSERT	0x2D	
VK_DELETE	0x2E	
VK_HELP	0x2F	
VK_0	0x30	numérique…
VK_1	0x31	
VK_2	0x32	
VK_3	0x33	
VK_4	0x34	
VK_5	0x35	
VK_6	0x36	
VK_7	0x37	
VK_8	0x38	
VK_9	0x39	
VK_A	0x41	alphabétique…
VK_B	0x42	
VK_C	0x43	
VK_D	0x44	
VK_E	0x45	
VK_F	0x46	
VK_G	0x47	
VK_H	0x48	
VK_I	0x49	
VK_J	0x4A	
VK_K	0x4B	
VK_L	0x4C	
VK_M	0x4D	
VK_N	0x4E	
VK_O	0x4F	
VK_P	0x50	
VK_Q	0x51	
VK_R	0x52	
VK_S	0x53	
VK_T	0x54	
VK_U	0x55	
VK_V	0x56	
VK_W	0x57	

Tableau 8-97/T.128 – Codes de clés virtuelles *(suite)*

Nom	Valeur	Commentaire
VK_X	0x58	
VK_Y	0x59	
VK_Z	0x5A	
VK_LEFT_MENU	0x5B	fonction menu gauche
VK_RIGHT_MENU	0x5C	fonction menu droit
VK_CONTEXT	0x5D	fonction menu contextuel
VK_NUMPAD0	0x60	fonctions clavier numérique…
VK_NUMPAD1	0x61	
VK_NUMPAD2	0x62	
VK_NUMPAD3	0x63	
VK_NUMPAD4	0x64	
VK_NUMPAD5	0x65	
VK_NUMPAD6	0x66	
VK_NUMPAD7	0x67	
VK_NUMPAD8	0x68	
VK_NUMPAD9	0x69	
VK_MULTIPLY	0x6A	
VK_ADD	0x6B	
VK_SUBTRACT	0x6D	
VK_DECIMAL	0x6E	
VK_DIVIDE	0x6F	
VK_F1	0x70	touches fonctions…
VK_F2	0x71	
VK_F3	0x72	
VK_F4	0x73	
VK_F5	0x74	
VK_F6	0x75	
VK_F7	0x76	
VK_F8	0x77	
VK_F9	0x78	
VK_F10	0x79	
VK_F11	0x7A	
VK_F12	0x7B	
VK_F13	0x7C	
VK_F14	0x7D	
VK_F15	0x7E	

Tableau 8-97/T.128 – Codes de clés virtuelles *(fin)*

Nom	Valeur	Commentaire
VK_F16	0x7F	
VK_F17	0x80	
VK_F18	0x81	
VK_F19	0x82	
VK_F20	0x83	
VK_F21	0x84	
VK_F22	0x85	
VK_F23	0x86	
VK_F24	0x87	
VK_NUMLOCK	0x90	verrouillage numérique
VK_SCROLL	0x91	verrouillage défilement
VK_PLUS	0xBB	
VK_COMMA	0xBC	
VK_MINUS	0xBD	
VK_PERIOD	0xBE	
VK_BAR	0xE2	trait plein (hors US)
VK_ATTN	0xF6	
VK_CRSEL	0xF7	
VK_EXSEL	0xF8	
VK_EREOF	0xF9	
VK_PLAY	0xFA	
VK_ZOOM	0xFB	
VK_PA1	0xFD	

8.18.4 Etat de clavier

Pendant une session AS, une ASCE a besoin d'envoyer des événements clavier seulement lors de la fourniture d'entrée à une application hébergée distante. Elle n'a pas besoin d'envoyer des événements clavier pour des touches enfoncées tant qu'elle n'est pas active, ou lorsque la conférence étant en mode présidé elle n'a pas de privilège de président, ou si elle n'est pas en mode in *control* ou *detached*.

Une ASCE émettrice s'assurera que les portions du flux d'entrée local envoyées dans des *InputPDUs* contenant des événements de clavier sont cohérentes en interne. Par exemple, si on suppose que l'utilisateur final local avait la touche *CapsLock* enfoncée avant que l'ASCE ne devienne active dans la session, et ensuite qu'il prenne le contrôle d'une application hébergée à distance et commence à frapper, l'ASCE doit précéder l'envoi d'événements de clavier avec des événements supplémentaires contenant des événements *CapsLock* enfoncé/relâché, et se fier aux ASCE homologues pour préparer leur état de clavier pour faire en sorte que les touches *CapsLock* enfoncées/relâchées prennent effet (ce qui peut être une opération nulle si la touche *CapsLock* distante est elle aussi enfoncée) pour s'assurer que l'entrée sera mise en majuscule sur le système distant.

On demande d'une manière générale à l'ASCE émettrice d'insérer des événements de clavier dans le flux d'entrée courant pour synchroniser le clavier distant avec l'état du clavier local.

Pendant une session AS, une ASCE peut recevoir des entrées en provenance de multiples ASCE homologues. Par exemple, là où une ASCE héberge une application dans une conférence multipoint, plusieurs ASCE homologues peuvent fournir des entrées tour à tour, comme les utilisateurs finaux distants peuvent échanger le contrôle et fournir des entrées à l'application hébergée. Alors qu'une ASCE peut se fier aux ASCE émettrices en insérant n'importe quel état requis (voir ci-dessus), elle est responsable du changement de tout état de clavier local réalisée à la place d'autres ASCE homologues avant le traitement d'entrées en provenance d'une nouvelle ASCE. Par exemple, si l'ASCE A héberge une application, l'ASCE B a la touche *CapsLock* enfoncée et contrôle l'aiguillage vers l'ASCE C qui a déjà la touche *NumLock* enfoncée, alors lorsque l'ASCE A commute une entrée de l'ASCE B vers l'ASCE C, elle peut se baser sur une ASCE C préparant son flux d'entrée avec un *NumLock-down*, *NumLock-up*, mais est elle-même responsable de la défection de l'état de la touche *CapsLock* établi à la place de l'ASCE B. Lorsque l'ASCE A commute ensuite l'entrée de l'ASCE C vers l'ASCE B, elle est alors responsable du rétablissement de l'état enfoncé de la touche *CapsLock* à la place de l'ASCE B.

Il est généralement demandé à une ASCE réceptrice d'assurer la responsabilité de prendre toute action locale nécessaire à la synchronisation de l'état du clavier local avec celle d'un nouveau flux d'entrée.

8.18.5 Touches silencieuses

De nombreux terminaux utilisent des séquences de touches spéciales pour effectuer des actions qui ne devraient pas être reproduites sur les ASCE homologues. Par exemple, ALT-TAB est habituellement utilisée pour faire passer de manière cyclique au premier plan les fenêtres d'un gestionnaire de fenêtres.

De telles séquences devraient avoir un effet seulement local, mais une ASCE émettrice peut être capable de les détecter seulement après l'émission du premier événement dans la séquence. Là où ce serait le cas, l'ASCE émettrice enverrait des événements ultérieurs avec l'indicateur binaire *Quiet* positionné dans le paramètre *keyboardFlags* pour indiquer aux ASCE homologues qu'elles doivent annuler l'effet de la séquence.

Par exemple, une ASCE peut envoyer ALT-TAB comme: *ALT-down*, *TAB-down-quiet*, *TAB-up-quiet*, *ALT-up-quiet*.

Là où une ASCE réceptrice détecte que l'indicateur binaire *Quiet* est positionné pour un événement relâché, cette touche et son événement enfoncé associé peuvent être ignorés. Là où l'ASCE réceptrice détecte que l'indicateur binaire *Quiet* est positionné seulement après le relâchement d'une touche, alors il peut y avoir des événements locaux par lesquels l'effet de la touche peut être annulé en ajoutant des événements de touche supplémentaires avant ou après l'événement enfoncé associé.

Par exemple, une ASCE peut traiter la séquence ALT-TAB envoyée ci-dessus comme: *ALT-down*, ignore *TAB-down-quiet*, écarte *TAB-up-quiet*, insère des événements de suppression dépendants du terminal, *ALT-up*.

8.18.6 Evénement de synchronisation d'entrée

Lorsqu'une ASCE détectera qu'une nouvelle ASCE est devenue active, elle réinitialisera son état de clavier d'émission et mettra en file d'attente un événement de synchronisation d'entrée en vue d'une émission dans l'*InputPDU* suivant. Ensuite, lors de l'émission d'événements d'entrée, elle insérera les événements d'entrée pour mettre les ASCE homologues dans un état de clavier identique à celui du terminal local.

Sur réception d'un événement de synchronisation d'entrée dans un *InputPDU*, une ASCE réinitialisera son état de réception de clavier. Ensuite, avant de recevoir une entrée d'une ASCE homologue particulière, elle pourra s'attendre à recevoir n'importe quel état de clavier requis applicable à cette entrée.

Voir 8.4 pour plus d'information sur l'activation d'ASCE et 8.6 pour plus d'information sur la synchronisation. L'événement de synchronisation d'entrée est décrit dans le Tableau 8-98.

Tableau 8-98/T.128 – Evénement de synchronisation d'entrée

Paramètre	Description
eventTime	ce paramètre est le temps local de l'ASCE en millisecondes du temps d'apparition de l'événement.
nonStandardParameters	ce paramètre est autorisé seulement dans le mode de base du protocole AS. C'est une liste optionnelle de paramètres hors normes autorisés seulement si les capacités hors normes correspondantes sont présentes dans le jeu de capacités négocié.

8.19 Fonctionnement en mode présidé

Lorsqu'une conférence est en mode présidé, les droits d'une ASCE pour héberger des applications et pour fournir des événements d'entrée peuvent être réduits par le nœud présidant.

Lorsqu'une conférence rentre en mode présidé ou lorsque la Présidence se déplace d'un nœud à un autre, chaque ASCE reçoit une indication *GCC-Conductor-Assign*. Sur réception d'une telle indication, aucune ASCE située à un nœud différent du nœud présidant n'a la permission d'héberger des applications ou de fournir des entrées.

La réception d'indications *GCC-Conductor-Permission-Grant* en mode présidé accorde et refuse aux ASCE situés à des nœuds autres que le nœud présidant le droit d'héberger des applications et de fournir des entrées. Le droit est accordé si le paramètre *Permission Flag* est à TRUE et refusé s'il est à FALSE.

Une ASCE situé au nœud présidant d'une conférence en mode présidé peut héberger des applications et fournir des entrées.

En mode non présidé, toutes les ASCE peuvent héberger des applications et fournir des entrées, conformément à la politique de contrôle de la conférence (voir 8.12 et 8.13).

9 Définitions d'ASPDU

La structure des ASPDU pour les modes de base et hérité du protocole AS est spécifiée comme suit en utilisant la notation ASN.1 de la Recommandation X.680.

Les ASPDU du mode hérité sont spécifiés au 9.1 et seront codés en vue de leur transmission en appliquant les règles de codage définies au 9.3.

Les ASPDU du mode de base sont définies au 9.2 et seront codées en vue de leur transmission en appliquant la variante BASIC ALIGNED des Règles de Codage compact de la Recommandation X.691.

Les ASPDU dans les modes de base et hérité seront toutes les deux codées et placées dans le champ donnée des primitives *MCS-Send-Data*, avec la chaîne de bits générée par le codage placée dans la chaîne d'octet utilisée par MCS dans un ordre tel que pour chaque octet, le bit de tête est placé dans le bit le plus significatif et le bit de queue est placé dans la position la moins significative.

9.1 Définition ASN.1 du mode hérité

```
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                              Begin AS Definitions
--
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

AS-PROTOCOL DEFINITIONS ::=

BEGIN

-- NOTE – ==================================================
-- NOTE – All abstract types defined shall be exported
-- NOTE –  ==================================================

-------------------------------------------------------------------------------------------------
-- Constants
-------------------------------------------------------------------------------------------------

maxSourceDescriptor      INTEGER ::= 48
maxTerminalDescriptor    INTEGER ::= 16
maxFonts                 INTEGER ::= 700
maxPassword              INTEGER ::= 9
maxFaceName              INTEGER ::= 32
maxTitleString       INTEGER ::= 50
maxInputEvents           INTEGER ::= 50

-------------------------------------------------------------------------------------------------
-- Base Types
-------------------------------------------------------------------------------------------------

BitString8          ::= BIT STRING (SIZE (0..7))
BitString16         ::= BIT STRING (SIZE (0..15))
BitString32         ::= BIT STRING (SIZE (0..31))
Coordinate8         ::= INTEGER (–128..127)
Coordinate16        ::= INTEGER (–32768..32767)
Integer4            ::= INTEGER (0..15)
Integer8            ::= INTEGER (0..255)
Integer12           ::= INTEGER (0..4095)
Integer16           ::= INTEGER (0..65535)
Integer32           ::= INTEGER (0..4294967295)
Boolean16           ::= Integer16 {false(0), true(1)}

UserID              ::= Integer16
ShareID             ::= Integer32
WindowID            ::= Integer32

T50String           ::= OCTET STRING (SIZE (0..255)) -- -- T.50 String
ASString            ::= OCTET STRING (SIZE (0..255)) -- -- AS Protocol CodePage String
```

```
-----------------------------------------------------------------------------------------------
-- Bit Flag Types
-----------------------------------------------------------------------------------------------

BitmapCompressionCapabilityFlags ::= BitString16
{
   bitmapCompressionSupported          (0)
}

BoundsOrderFlags ::= BitString8
{
   absoluteLeftPresent                 (0),
   absoluteTopPresent                  (1),
   absoluteRightPresent                (2),
   absoluteBottomPresent               (3),
   deltaLeftPresent                    (4),
   deltaTopPresent                     (5),
   deltaRightPresent                   (6),
   deltaBottomPresent                  (7)
}

ControlCapabilityFlags ::= BitString16
{
   allowMediatedControl                (0)
}

ControlOrderFlags ::= BitString8
{
   standard                            (0), -- Mandatory this flag is set
   secondary                           (1),
   bounds                              (2),
   typeChange                          (3),
   deltaCoordinates                    (4)
}

ExtraOrderFlags ::= BitString16
{
   secondary                           (3)
}

ExtraTextFlags ::= BitString16
{
   opaqueRectangle                     (1),
   clipToRectangle                     (2),
   deltaXPresent                       (15)
}

FontAttributeFlags ::= BitString16
{
   fixedPitch                          (0),
   fixedSize                           (1)
}

KeyboardFlags ::= BitString16
{
   right                               (0),
   quiet                               (12),
   down                                (14),
   release                             (15)
}
```

```
OrderCapabilityFlags ::= BitString16
{
   negotiateOrderSupport          (1),   -- Mandatory this flag is set
   cannotReceiveOrders            (2)
}

PointingDeviceFlags ::= BitString16
{
   move                           (11),
   button1                        (12),
   button2                        (13),
   button3                        (14),
   down                           (15)
}

TextAttributeFlags ::= BitString16
{
   italic                         (2),
   underline                      (3),
   strikeout                      (4),
   useBaselineStart               (8)
}

TextCapabilityFlags ::= BitString16
{
   checkFontAspect                (0),
   allowDeltaXSimulation          (5),
   checkFontSignatures            (7),
   useBaselineStart               (9)
}

WindowAttributeFlags ::= BitString32
{
   minimized                      (0),
   taggable                       (1),
   hosted                         (2),
   shadow                         (3),
   local                          (4),
   topmost                        (5),
   windowManagerMinimized         (16),
   windowManagerInvisible         (17)
}

-------------------------------------------------------------------------------------------------
-- General Types
-------------------------------------------------------------------------------------------------

ApplicationAction ::= Integer16
{
   notifyHostedApplications       (1),
   unhostApplication              (2)
}


BackgroundMixMode ::= Integer16
{
   transparent                    (1),
   opaque                         (2)
}
```

```
BitmapData ::= CHOICE
{
   uncompressedBitmapData          OCTET STRING,
   compressedBitmapData            CompressedBitmapData
}

Brush ::= SEQUENCE
{
   originX                         Integer8                    OPTIONAL,
   originY                         Integer8                    OPTIONAL,
   style                           BrushStyle                  OPTIONAL,
   hatch                           BrushHatch                  OPTIONAL,
   pattern                         OCTET STRING (SIZE (7))     OPTIONAL
}

BrushHatch ::= CHOICE
{
   style                           HatchStyle,
   patternZero                     Integer8
}

BrushStyle ::= Integer8
{
   solid                           (0),
   null                            (1),
   hatched                         (2),
   pattern                         (3)
}

Color ::= SEQUENCE
{
   red                             Integer8,
   green                           Integer8,
   blue                            Integer8
}

ColorQuad ::= SEQUENCE
{
   blue                            Integer8,
   green                           Integer8,
   red                             Integer8,
   pad1octet                       Integer8,
}

ColorPointerAttribute ::= SEQUENCE
{
   cacheIndex                      Integer16,
   hotSpot                         Point16,
   width                           Integer16,
   height                          Integer16,
   lengthANDMAsk                   Integer16,
                                   -- length in octets of AND mask in colorPointerData
   lengthXORMAsk                   Integer16,
                                   -- length in octets of XOR mask in colorPointerData
   colorPointerData                OCTET STRING
}

CompressedBitmapData ::= SEQUENCE
{
   pad2octets                      Integer16 (0),
   mainBodySize                    Integer16,
```

```
   rowSize                    Integer16,
   uncompressedSize           Integer16,
   compressedBitmap           OCTET STRING
}

ControlAction ::= Integer16
{
   requestControl             (1),
   detach                     (3),
   grantControl               (2),
   cooperate                  (4)
}

ControlPriority ::= Integer16
{
   always                     (1),
   never                      (2),
   confirm                    (3)
}

Coordinate  ::= CHOICE
{
   absolute                   Coordinate16,
   delta                      Coordinate8
}

DesktopSaveAction ::= Integer8
{
   desktopSave                (0),
   desktopRestore             (1)
}

FontAttribute ::= SEQUENCE
{
   faceName                   T50String (SIZE (1..maxFaceName)),
   fontFlags                  FontAttributeFlags,
   averageWidth               Integer16,
   height                     Integer16,
   aspectX                    Integer16,
   aspectY                    Integer16,
   signature1                 Integer8,
   signature2                 Integer8,
   signature3                 Integer16,
   codePage                   FontCodePage,
   ascent                     Integer16
}

FontCodePage ::= Integer16
{
   allCodePoints              (0),
   coreCodePoints             (255)
}

HatchStyle ::= Integer8
{
   horizontal                 (0),
   vertical                   (1),
   forward                    (2),
   backward                   (3),
```

```
   cross                          (4),
   diagonal                       (5)
}

InputMessageType ::= Integer16
{
   inputSynchronize               (0),
   inputCodePoint                 (1),
   inputVirtualKey                (2),
   inputPointingDevice            ('8001'H)
}

MediatedControlAction ::= Integer16
{
   takeControlRequest             (1),
   passControlRequest             (2),
   detachRequest                  (3)
   confirmTakeResponse            (5),
   denyTakeResponse               (6),
   confirmDetachResponse          (7),
   denyDetachResponse             (8),
   denyPassResponse               (9),
   remoteDetachRequest            (10),
   denyRemoteDetachRequest        (11)
}

MonoPointerAttribute ::= SEQUENCE
{
   hotSpot                        Point16,
   width                          Integer16,
   height                         Integer16,
   lengthPointerData              Integer16,
                                  -- length in octets of monoPointerData
   monoPointerData                OCTET STRING
}

OSMajorType ::= Integer16
{
   unspecified                    (0),
   windows                        (1),
   OS2                            (2),
   macintosh                      (3),
   unix                           (4)
}

OSMinorType ::= Integer16
{
   unspecified                    (0),
   windows_31x                    (1),
   windows_95                     (2),
   windows_NT                     (3),
   OS2_V21                        (4),
   power_pc                       (5),
   macintosh                      (6),
   native_XServer                 (7),
   pseudo_XServer                 (8)

}
```

```
PDUType ::= Integer4
{
   confirmActivePDU          (3),
   dataPDU                   (7),
   deactivateAllPDU          (6),
   deactivateOtherPDU        (4),
   deactivateSelfPDU         (5),
   demandActivePDU           (1),
   requestActivePDU          (2)
}

PDUType2 ::= Integer8
{
   application               (25),
   control                   (20),
   font                      (11),
   input                     (28),
   mediatedControl           (29),
   pointer                   (27),
   remoteShare               (30),
   synchronize               (31),
   update                    (2),
   updateCapability          (32),
   windowActivation          (23),
   windowList                (24)
}

PDUTypeFlow ::= Integer8
{
   flowResponsePDU           (66),
   flowStopPDU               (67),
   flowTestPDU               (65)
}

Pen ::= SEQUENCE
{
   style                     PenStyle          OPTIONAL,
   width                     Integer8 (1)      OPTIONAL,
   color                     Color             OPTIONAL
}

PenStyle ::= ENUMERATED
{
   solid                     (0),
   dashed                    (1),
   dotted                    (2),
   dash-dot                  (3),
   dash-dot-dot              (4),
   null                      (5)
}

Point16 ::= SEQUENCE
{
   x                         Coordinate16,
   y                         Coordinate16
}

PointerMessageType ::= Integer16
{
   cachedPointer             (7),
   colorPointer              (6),
```

```
   monoPointer                    (2),
   pointerPosition                (3),
   systemPointer                  (1)
}

PrimaryOrderType ::= Integer8
{
   destinationBlt                 (0),
   patternBlt                     (1),
   screenBlt                      (2),
   memoryBlt                      (13),
   memoryThreeWayBlt              (14),
   text                           (5),
   extendedText                   (6),
   frame                          (9),
   rectangle                      (7),
   line                           (8),
   opaqueRectangle                (10),
   desktopSave                    (11),
   desktopOrigin                  (32)
}

Rectangle16 ::= SEQUENCE
{
   left                           Coordinate16,
   top                            Coordinate16,
   right                          Coordinate16,
   bottom                         Coordinate16
}

RemoteShareAction ::= Integer16
{
   requestRemoteShare             (1),
   confirmRemoteShare             (2),
   denyRemoteShare                (3)
}

RemoteShareDenial ::= Integer16
{
   incorrectPassword              (1),
   remoteShareNotEnabled          (2),
   remoteShareInOperationIncoming (3),
   remoteShareInOperationOutgoing (4)
}

ROP2 ::= Integer8
{
   r2BLACK                        (1),
   r2DPon                         (2),
   r2DPna                         (3),
   r2Pn                           (4),
   r2PDna                         (5),
   r2Dn                           (6),
   r2DPx                          (7),
   r2DPan                         (8),
   r2DPa                          (9),
   r2DPxn                         (10),
   r2D                            (11),
   r2DPno                         (12),
   r2P                            (13),
   r2PDno                         (14),
```

```
   r2DPo                    (15),
   r2WHITE                  (16)
}

ROP3 ::= Integer8
{
   r3BLACK                  ('00'H),
   r3DPSoon                 ('01'H),
   r3DPSona                 ('02'H),
   r3PSon                   ('03'H),
   r3SDPona                 ('04'H),
   r3DPon                   ('05'H),
   r3PDSxnon                ('06'H),
   r3PDSaon                 ('07'H),
   r3SDPnaa                 ('08'H),
   r3PDSxon                 ('09'H),
   r3DPna                   ('0A'H),
   r3PSDnaon                ('0B'H),
   r3SPna                   ('0C'H),
   r3PDSnaon                ('0D'H),
   r3PDSonon                ('0E'H),
   r3Pn                     ('0F'H),
   r3PDSona                 ('10'H),
   r3DSon                   ('11'H),
   r3SDPxnon                ('12'H),
   r3SDPaon                 ('13'H),
   r3DPSxnon                ('14'H),
   r3DPSaon                 ('15'H),
   r3PSDPSanaxx             ('16'H),
   r3SSPxDSxaxn             ('17'H),
   r3SPxPDxa                ('18'H),
   r3SDPSanaxn              ('19'H),
   r3PDSPaox                ('1A'H),
   r3SDPSxaxn               ('1B'H),
   r3PSDPaox                ('1C'H),
   r3DSPDxaxn               ('1D'H),
   r3PDSox                  ('1E'H),
   r3PDSoan                 ('1F'H),
   r3DPSnaa                 ('20'H),
   r3SDPxon                 ('21'H),
   r3DSna                   ('22'H),
   r3SPDnaon                ('23'H),
   r3SPxDSxa                ('24'H),
   r3PDSPanaxn              ('25'H),
   r3SDPSaox                ('26'H),
   r3SDPSxnox               ('27'H),
   r3DPSxa                  ('28'H),
   r3PSDPSaoxxn             ('29'H),
   r3DPSana                 ('2A'H),
   r3SSPxPDxaxn             ('2B'H),
   r3SPDSoax                ('2C'H),
   r3PSDnox                 ('2D'H),
   r3PSDPxox                ('2E'H),
   r3PSDnoan                ('2F'H),
   r3PSna                   ('30'H),
   r3SDPnaon                ('31'H),
   r3SDPSoox                ('32'H),
   r3Sn                     ('33'H),
   r3SPDSaox                ('34'H),
   r3SPDSxnox               ('35'H),
   r3SDPox                  ('36'H),
```

r3SDPoan ('37'H),
r3PSDPoax ('38'H),
r3SPDnox ('39'H),
r3SPDSxox ('3A'H),
r3SPDnoan ('3B'H),
r3PSx ('3C'H),
r3SPDSonox ('3D'H),
r3SPDSnaox ('3E'H),
r3PSan ('3F'H),
r3PSDnaa ('40'H),
r3DPSxon ('41'H),
r3SDxPDxa ('42'H),
r3SPDSanaxn ('43'H),
r3SDna ('44'H),
r3DPSnaon ('45'H),
r3DSPDaox ('46'H),
r3PSDPxaxn ('47'H),
r3SDPxa ('48'H),
r3PDSPDaoxxn ('49'H),
r3DPSDoax ('4A'H),
r3PDSnox ('4B'H),
r3SDPana ('4C'H),
r3SSPxDSxoxn ('4D'H),
r3PDSPxox ('4E'H),
r3PDSnoan ('4F'H),
r3PDna ('50'H),
r3DSPnaon ('51'H),
r3DPSDaox ('52'H),
r3SPDSxaxn ('53'H),
r3DPSonon ('54'H),
r3Dn ('55'H),
r3DPSox ('56'H),
r3DPSoan ('57'H),
r3PDSPoax ('58'H),
r3DPSnox ('59'H),
r3DPx ('5A'H),
r3DPSDonox ('5B'H),
r3DPSDxox ('5C'H),
r3DPSnoan ('5D'H),
r3DPSDnaox ('5E'H),
r3DPan ('5F'H),
r3PDSxa ('60'H),
r3DSPDSaoxxn ('61'H),
r3DSPDoax ('62'H),
r3SDPnox ('63'H),
r3SDPSoax ('64'H),
r3DSPnox ('65'H),
r3DSx ('66'H),
r3SDPSonox ('67'H),
r3DSPDSonoxxn ('68'H),
r3PDSxxn ('69'H),
r3DPSax ('6A'H),
r3PSDPSoaxxn ('6B'H),
r3SDPax ('6C'H),
r3PDSPDoaxxn ('6D'H),
r3SDPSnoax ('6E'H),
r3PDSxnan ('6F'H),
r3PDSana ('70'H),
r3SSDxPDxaxn ('71'H),
r3SDPSxox ('72'H),
r3SDPnoan ('73'H),

r3DSPDxox ('74'H),
r3DSPnoan ('75'H),
r3SDPSnaox ('76'H),
r3DSan ('77'H),
r3PDSax ('78'H),
r3DSPDSoaxxn ('79'H),
r3DPSDnoax ('7A'H),
r3SDPxnan ('7B'H),
r3SPDSnoax ('7C'H),
r3DPSxnan ('7D'H),
r3SPxDSxo ('7E'H),
r3DPSaan ('7F'H),
r3DPSaa ('80'H),
r3SPxDSxon ('81'H),
r3DPSxna ('82'H),
r3SPDSnoaxn ('83'H),
r3SDPxna ('84'H),
r3PDSPnoaxn ('85'H),
r3DSPDSoaxx ('86'H),
r3PDSaxn ('87'H),
r3DSa ('88'H),
r3SDPSnaoxn ('89'H),
r3DSPnoa ('8A'H),
r3DSPDxoxn ('8B'H),
r3SDPnoa ('8C'H),
r3SDPSxoxn ('8D'H),
r3SSDxPDxax ('8E'H),
r3PDSanan ('8F'H),
r3PDSxna ('90'H),
r3SDPSnoaxn ('91'H),
r3DPSDPoaxx ('92'H),
r3SPDaxn ('93'H),
r3PSDPSoaxx ('94'H),
r3DPSaxn ('95'H),
r3DPSxx ('96'H),
r3PSDPSonoxx ('97'H),
r3SDPSonoxn ('98'H),
r3DSxn ('99'H),
r3DPSnax ('9A'H),
r3SDPSoaxn ('9B'H),
r3SPDnax ('9C'H),
r3DSPDoaxn ('9D'H),
r3DSPDSaoxx ('9E'H),
r3PDSxan ('9F'H),
r3DPa ('A0'H),
r3PDSPnaoxn ('A1'H),
r3DPSnoa ('A2'H),
r3DPSDxoxn ('A3'H),
r3PDSPonoxn ('A4'H),
r3PDxn ('A5'H),
r3DSPnax ('A6'H),
r3PDSPoaxn ('A7'H),
r3DPSoa ('A8'H),
r3DPSoxn ('A9'H),
r3D ('AA'H),
r3DPSono ('AB'H),
r3SPDSxax ('AC'H),
r3DPSDaoxn ('AD'H),
r3DSPnao ('AE'H),
r3DPno ('AF'H),
r3PDSnoa ('B0'H),

r3PDSPxoxn ('B1'H),
r3SSPxDSxox ('B2'H),
r3SDPanan ('B3'H),
r3PSDnax ('B4'H),
r3DPSDoaxn ('B5'H),
r3DPSDPaoxx ('B6'H),
r3SDPxan ('B7'H),
r3PSDPxax ('B8'H),
r3DSPDaoxn ('B9'H),
r3DPSnao ('BA'H),
r3DSno ('BB'H),
r3SPDSanax ('BC'H),
r3SDxPDxan ('BD'H),
r3DPSxo ('BE'H),
r3DPSano ('BF'H),
r3PSa ('C0'H),
r3SPDSnaoxn ('C1'H),
r3SPDSonoxn ('C2'H),
r3PSxn ('C3'H),
r3SPDnoa ('C4'H),
r3SPDSxoxn ('C5'H),
r3SDPnax ('C6'H),
r3PSDPoaxn ('C7'H),
r3SDPoa ('C8'H),
r3SPDoxn ('C9'H),
r3DPSDxax ('CA'H),
r3SPDSaoxn ('CB'H),
r3S ('CC'H),
r3SDPono ('CD'H),
r3SDPnao ('CE'H),
r3SPno ('CF'H),
r3PSDnoa ('D0'H),
r3PSDPxoxn ('D1'H),
r3PDSnax ('D2'H),
r3SPDSoaxn ('D3'H),
r3SSPxPDxax ('D4'H),
r3DPSanan ('D5'H),
r3PSDPSaoxx ('D6'H),
r3DPSxan ('D7'H),
r3PDSPxax ('D8'H),
r3SDPSaoxn ('D9'H),
r3DPSDanax ('DA'H),
r3SPxDSxan ('DB'H),
r3SPDnao ('DC'H),
r3SDno ('DD'H),
r3SDPxo ('DE'H),
r3SDPano ('DF'H),
r3PDSoa ('E0'H),
r3PDSoxn ('E1'H),
r3DSPDxax ('E2'H),
r3PSDPaoxn ('E3'H),
r3SDPSxax ('E4'H),
r3PDSPaoxn ('E5'H),
r3SDPSanax ('E6'H),
r3SPxPDxan ('E7'H),
r3SSPxDSxax ('E8'H),
r3DSPDSanaxxn ('E9'H),
r3DPSao ('EA'H),
r3DPSxno ('EB'H),
r3SDPao ('EC'H),
r3SDPxno ('ED'H),

```
   r3DSo                          ('EE'H),
   r3SDPnoo                       ('EF'H),
   r3P                            ('F0'H),
   r3PDSono                       ('F1'H),
   r3PDSnao                       ('F2'H),
   r3PSno                         ('F3'H),
   r3PSDnao                       ('F4'H),
   r3PDno                         ('F5'H),
   r3PDSxo                        ('F6'H),
   r3PDSano                       ('F7'H),
   r3PDSao                        ('F8'H),
   r3PDSxno                       ('F9'H),
   r3DPo                          ('FA'H),
   r3DPSnoo                       ('FB'H),
   r3PSo                          ('FC'H),
   r3PSDnoo                       ('FD'H),
   r3DPSoo                        ('FE'H),
   r3WHITE                        ('FF'H)
}

SecondaryOrderType ::= Integer8
{
   cacheBitmapUncompressed        (0),
   cacheColorTable                (1),
   cacheBitmapCompressed          (2)
}

StreamID ::= Integer8
{
   streamLowPriority              (1),
   streamMediumPriority           (2),
   streamHighPriority             (4)
}

SynchronizeMessageType ::= Integer16
{
   synchronize                    (1)
}

SystemPointerType ::= Integer32
{
   nullPointer                    (0),
   defaultPointer                 ('00007F00'H)
}

UpdateType ::= Integer16
{
   orders                         (0),
   bitmap                         (1),
   palette                        (2),
   synchronize                    (3)
}

WindowActivationAction ::= Integer16
{
   localWindowActive              (1),
   hostedWindowActive             (2),
   hostedWindowInvisible          (3),
   pointerDeviceCapture           (4),
   activateWindow                 ('8001'H),
   closeWindow                    ('8002'H),
```

```
   restoreWindow                (’8003’H),
   windowManagerMenu            (’8004’H),
   activationHelpKey            (’8011’H),
   activationHelpIndexKey       (’8012’H),
   activationHelpExtendedKey    (’8013’H)
}

WindowAttribute ::= SEQUENCE
{
   windowID                     WindowID,
   windowExtra                  Integer32,
   windowOwner                  WindowID,
   windowFlags                  WindowAttributeFlags,
   windowRectangle              Rectangle16
}

WindowListMessageType ::= Integer16
{
   updateWindowList             (1)
}

WindowTitle ::= CHOICE
{
   noTitle                      Integer8 (255),
   titleString                  T50String (SIZE(1..maxTitleString))
}

------------------------------------------------------------------------------------------------
-- Capability Types
------------------------------------------------------------------------------------------------

CapabilitySetType ::= Integer16
{
   bitmapCacheCapabilitySet     (4),
   bitmapCapabilitySet          (2),
   colorCacheCapabilitySet      (10),
   controlCapabilityset         (5),
   generalCapabilitySet         (1),
   orderCapabilitySet           (3),
   pointerCapabilitySet         (8),
   activationCapabilitySet      (7)
   shareCapabilitySet           (9),
}


GeneralCapabilitySet ::= SEQUENCE
{
   capabilitySetType            CapabilitySetType (generalCapabilitySet),
   lengthCapability             Integer16,
                                -- length of capability set in octets
                                -- (including type and length parameters)
   osMajorType                  OSMajorType,
   osMinorType                  OSMinorType,
   protocolVersion              Integer16 (’0200’H)
   pad2octetsA                  Integer16,
   generalCompressionTypes      Integer16,
   pad2octetsB                  Integer16,
   updatecapabilityFlag         Boolean16,
   remoteUnshareFlag            Boolean16,
```

```
   generalCompressionLevel          Integer16,
   pad2octetsC                      Integer16
}

BitmapCapabilitySet ::= SEQUENCE
{
   capabilitySetType                CapabilitySetType (bitmapCapabilitySet),
   lengthCapability                 Integer16,
                                    -- length of capability set in octets
                                    -- (including type and length parameters)
   preferredBitsPerPixel            Integer16 (1..8),
   receive1BitPerPixelFlag          Boolean16,
   receive4BitsPerPixelFlag         Boolean16,
   receive8BitsPerPixelFlag         Boolean16,
   desktopWidth                     Integer16,
   desktopHeight                    Integer16,
   pad2octetsA                      Integer16,
   desktopResizeFlag                Boolean16,
   bitmapCompressionType            BitmapCompressionCapabilityFlags,
   pad2octetsC                      Integer16
}

OrderCapabilitySet ::= SEQUENCE
{
   capabilitySetType                CapabilitySetType (orderCapabilitySet),
   lengthCapability                 Integer16,
                                    -- length of capability set in octets
                                    -- (including type and length parameters)
   terminalDescriptor               T50String (SIZE (1..maxTerminalDescriptor)),
   pad4octetsA                      Integer32 (0),
   desktopXGranularity              Integer16,
   desktopYGranularity              Integer16,
   pad2octetsA                      Integer16 (0),
   maximumOrderLevel                Integer16,
   numberFonts                      Integer16 (1..maxFonts),
   orderFlags                       OrderCapabilityFlags,
   orderSupport ::= SEQUENCE (SIZE (32)) OF
   {
      destinationBltSupport         Integer8,
      patternBltSupport             Integer8,
      screenBltSupport              Integer8,
      memoryBltSupport              Integer8,
      memoryThreeWayBltSupport      Integer8,
      textSupport                   Integer8,
      extendedTextSupport           Integer8,
      rectangleSupport              Integer8,
      lineSupport                   Integer8,
      frameSupport                  Integer8,
      opaqueRectangleSupport        Integer8,
      desktopSaveSupport            Integer8,
      undefinedOrder12              Integer8 (0),
      undefinedOrder13              Integer8 (0),
      undefinedOrder14              Integer8 (0),
      undefinedOrder15              Integer8 (0),
      undefinedOrder16              Integer8 (0),
      undefinedOrder17              Integer8 (0),
      undefinedOrder18              Integer8 (0),
      undefinedOrder19              Integer8 (0),
      undefinedOrder20              Integer8 (0),
      undefinedOrder21              Integer8 (0),
      undefinedOrder22              Integer8 (0),
```

```
    undefinedOrder23            Integer8 (0),
    undefinedOrder24            Integer8 (0),
    undefinedOrder25            Integer8 (0),
    undefinedOrder26            Integer8 (0),
    undefinedOrder27            Integer8 (0),
    undefinedOrder28            Integer8 (0),
    undefinedOrder29            Integer8 (0),
    undefinedOrder30            Integer8 (0),
    undefinedOrder31            Integer8 (0)
  }
  textFlags                     TextCapabilityFlags,
  pad2octetsB                   Integer16 (0),
  pad4octetsB                   Integer32 (0),
  desktopSaveSize               Integer32,
  pad4octetsC                   Integer32 (0)
}

BitmapCacheCapabilitySet ::= SEQUENCE
{
  capabilitySetType             CapabilitySetType (bitmapCacheCapabilitySet),
  lengthCapability              Integer16,
                                -- length of capability set in octets
                                -- (including type and length parameters)
  pad4octetsA                   Integer32 (0),
  pad4octetsB                   Integer32 (0),
  pad4octetsC                   Integer32 (0),
  pad4octetsD                   Integer32 (0),
  pad4octetsE                   Integer32 (0),
  pad4octetsF                   Integer32 (0),
  cache1Entries                 Integer16,
  cache1MaximumCellSize         Integer16 (256..16384),
  cache2Entries                 Integer16,
  cache2MaximumCellSize         Integer16 (256..16384),
  cache3Entries                 Integer16,
  cache3MaximumCellSize         Integer16 (256..16384)
}

ColorCacheCapabilitySet ::= SEQUENCE
{
  capabilitySetType             CapabilitySetType (colorCacheCapabilitySet),
  lengthCapability              Integer16,
                                -- length of capability set in octets
                                -- (including type and length parameters)
  colorTablecacheSize           Integer16 (1..255),
  pad2octetsA                   Integer16
}

ActivationCapabilitySet ::= SEQUENCE
{
  capabilitySetType             CapabilitySetType (activationCapabilitySet),
  lengthCapability              Integer16,
                                -- length of capability set in octets
                                -- (including type and length parameters)
  helpKeyFlag                   Boolean16,
  helpIndexKeyFlag              Boolean16,
  helpExtendedKeyFlag           Boolean16,
  windowActivateFlag            Boolean16
}
```

```
ControlCapabilitySet ::= SEQUENCE
{
   capabilitySetType          CapabilitySetType (controlCapabilitySet),
   lengthCapability           Integer16,
                              -- length of capability set in octets
                              -- (including type and length parameters)
   controlFlags               ControlCapabilityFlags,
   remoteDetachFlag           Boolean16,
   controlInterest            ControlPriority,
   detachInterest             ControlPriority
}

PointerCapabilitySet ::= SEQUENCE
{
   capabilitySetType          CapabilitySetType (pointerCapabilitySet),
   lengthCapability           Integer16,
                              -- length of capability set in octets
                              -- (including type and length parameters)
   colorPointerFlag           Boolean16,
   pointerCacheSize           Integer16 (1..500)
}


ShareCapabilitySet ::= SEQUENCE
{
   capabilitySetType          CapabilitySetType (shareCapabilitySet),
   lengthCapability           Integer16,
                              -- length of capability set in octets
                              -- (including type and length parameters)
   nodeID                     Integer32
}

NonStandardCapabilitySet ::= SEQUENCE
{
   capabilitySetType          Integer16,
                              -- defined by ASCE
   lengthCapability           Integer16,
                              -- length of capability set in octets
                              -- (including type and length parameters)
   nonStandardParameters      OCTET STRING
}

CombinedCapabilities ::= SEQUENCE
{
   numberCapabilities         Integer16,
                              -- number of capabilities in combinedCapabilities set
   pad2octets                 Integer16 (0),
   combinedCapabilities       SET
   {
      generalCapabilitySet    GeneralCapabilitySet,
      bitmapCapabilitySet     BitMapCapabilitySet,
      orderCapabilitySet      OrderCapabilitySet,
      bitmapCacheCapabilitySet BitmapCacheCapabilitySet,
      colorCacheCapabilitySet ColorCacheCapabilitySet,
      activationCapabilitySet activationCapabilitySet,
      controlCapabilitySet    ControlCapabilitySet,
      pointerCapabilitySet    PointerCapabilitySet,
      shareCapabilitySet      ShareCapabilitySet,
      nonStandardCapabilitySet NonStandardcapabilitySet OPTIONAL
   }
}
```

```
UpdateCapabilitySet ::= CHOICE
{
   bitmapCapabilitySet              BitmapCapabilitySet
}

--------------------------------------------------------------------------------------------------
-- Input Types
--------------------------------------------------------------------------------------------------

InputEvent ::= CHOICE
{
   pointingDeviceEvent              PointingDeviceEvent,
   keyboardEvent                    KeyboardEvent,
   synchronizeEvent                 SynchronizeEvent
}

KeyboardEvent ::= SEQUENCE
{
   eventTime                        Integer32,
   messageType                      InputMessageType (inputCodePoint |
                                                      inputVirtualKey ),
   keyboardFlags                    KeyboardFlags,
   keyCode                          Integer16
                                    -- AS protocol code page codepoint or virtual keycode
}

PointingDeviceEvent ::= SEQUENCE
{
   eventTime                        Integer32,
   messageType                      InputMessageType (inputPointingDevice),
   pointingDeviceFlags              PointingDeviceFlags,
   pointingDeviceX                  Coordinate16,
   pointingDeviceY                  Coordinate16
}

SynchronizeEvent ::= SEQUENCE
{
   eventTime                        Integer32,
   messageType                      InputMessageType (inputSynchronize)
}

--------------------------------------------------------------------------------------------------
-- Common Header Types
--------------------------------------------------------------------------------------------------

PrimaryOrderHeader  ::= SEQUENCE
{
   controlFlags                     ControlOrderFlags,
   orderType                        PrimaryOrderType OPTIONAL,
   encodingFlags                    SEQUENCE (SIZE (1..3)) OF BitString8,
   boundsFlags                      BoundsOrderFlags OPTIONAL,
   boundsLeft                       Coordinate OPTIONAL,
   boundsTop                        Coordinate OPTIONAL,
   boundsRight                      Coordinate OPTIONAL,
   boundsBottom                     Coordinate OPTIONAL
}

SecondaryOrderHeader ::= SEQUENCE
{
   controlFlags                     ControlOrderFlags,
```

```
   orderLength                Integer16,
                              -- length in octets, from and including orderType, minus eight
   extraFlags                 ExtraOrderFlags,
   orderType                  SecondaryOrderType
}

ShareControlHeader  ::= SEQUENCE
{
   totalLength                Integer16 (0..32767),
   protocolVersion            Integer4 (1),
   pduType                    PDUType,
   pad1octet                  Integer8 (0),
   pduSource                  UserID
}

ShareDataHeader ::= SEQUENCE
{
   shareControlHeader         ShareControlHeader, -- PDUType = dataPDU
   shareID                    ShareID,
   pad1octet                  Integer8 (0),
   streamID                   StreamID,
   uncompressedLength         Integer16,
   pduType2                   PDUType2,
   generalCompressedType      Integer8,
   generalCompressedLength    Integer16
}

------------------------------------------------------------------------------------------------------
-- Order Types
------------------------------------------------------------------------------------------------------

DestinationBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader, -- PrimaryOrderType = destinationBlt
   destLeft                   Coordinate          OPTIONAL,
   destTop                    Coordinate          OPTIONAL,
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
   rop3                       ROP3                OPTIONAL
}

PatternBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader, -- PrimaryOrderType = patternBlt
   destLeft                   Coordinate          OPTIONAL,
   destTop                    Coordinate          OPTIONAL,
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
   rop3                       ROP3                OPTIONAL,
   backgroundColor            Color               OPTIONAL,
   foregroundColor            Color               OPTIONAL,
   brush                      Brush               OPTIONAL
}

ScreenBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader, -- PrimaryOrderType = screenBlt
   destLeft                   Coordinate          OPTIONAL,
   destTop                    Coordinate          OPTIONAL,
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
```

```
   rop3                    ROP3                    OPTIONAL,
   sourceX                 Coordinate              OPTIONAL,
   sourceY                 Coordinate              OPTIONAL
}

CacheBitmapOrder ::= SEQUENCE
{
   header                  SecondaryOrderHeader, -- SecondaryOrderType =
                                                 --      cacheBitmapUncompressed |
                                                 --      cacheBitmapCompressed
   cacheId                 Integer8 (0..2),
   pad1octet               Integer8 (0),
   bitmapWidth             Integer8,
   bitmapHeight            Integer8,
   bitmapBitsPerPel        Integer8 (1|4|8),
   bitmapLength            Integer16,
                           -- length of bitmapData in octets (after any compression)
   cacheIndex              Integer16,
   bitmapData              BitmapData
}

CacheColorTableOrder ::= SEQUENCE
{
   header                  SecondaryOrderHeader, -- SecondaryOrderType = cacheColorTable
   cacheIndex              Integer8,
   numberColors            Integer16 (16|256),
   colorTable              SEQUENCE (SIZE (16|256)) OF ColorQuad
}

MemoryBltOrder  ::= SEQUENCE
{
   header                  PrimaryOrderHeader, -- PrimaryOrderType = memoryBlt
   colorTableCacheIndex    Integer8                OPTIONAL,
   bitmapCacheID           Integer8                OPTIONAL,
   destLeft                Coordinate              OPTIONAL,
   destTop                 Coordinate              OPTIONAL,
   destWidth               Coordinate              OPTIONAL,
   destHeight              Coordinate              OPTIONAL,
   rop3                    ROP3                    OPTIONAL,
   sourceX                 Coordinate              OPTIONAL,
   sourceY                 Coordinate              OPTIONAL,
   bitmapCacheIndex        Integer16               OPTIONAL
}

MemoryThreeWayBltOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader, -- PrimaryOrderType = memoryThreeWayBlt
   colorTableCacheIndex    Integer8                OPTIONAL,
   bitmapCacheID           Integer8                OPTIONAL,
   destLeft                Coordinate              OPTIONAL,
   destTop                 Coordinate              OPTIONAL,
   destWidth               Coordinate              OPTIONAL,
   destHeight              Coordinate              OPTIONAL,
   rop3                    ROP3                    OPTIONAL,
   sourceX                 Coordinate              OPTIONAL,
   sourceY                 Coordinate              OPTIONAL,
   backgroundColor         Color                   OPTIONAL,
   foregroundColor         Color                   OPTIONAL,
   brush                   Brush                   OPTIONAL,
   bitmapCacheIndex        Integer16               OPTIONAL
}
```

```
TextOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader, -- PrimaryOrderType = text
   backMixMode             BackgroundMixMode     OPTIONAL,
   startX                  Coordinate            OPTIONAL,
   startY                  Coordinate            OPTIONAL,
   backgroundColor         Color                 OPTIONAL,
   foregroundColor         Color                 OPTIONAL,
   extraSpacing            Integer16             OPTIONAL,
   totalBreakSpacing       Integer16             OPTIONAL,
   breakCount              Integer16             OPTIONAL,
   fontHeight              Integer16             OPTIONAL,
   fontWidth               Integer16             OPTIONAL,
   fontWeight              Integer16             OPTIONAL,
   textFlags               TextAttributeFlags    OPTIONAL,
   fontID                  Integer16             OPTIONAL,
   numberCodePoints        Integer8 (1..255)     OPTIONAL,
                           -- number of codepoints in codePointList
   codePointList           ASString (SIZE (1..255))  OPTIONAL
}

ExtendedTextOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader, -- PrimaryOrderType = extendedText
   backMixMode             BackgroundMixMode     OPTIONAL,
   startX                  Coordinate            OPTIONAL,
   startY                  Coordinate            OPTIONAL,
   backgroundColor         Color                 OPTIONAL,
   foregroundColor         Color                 OPTIONAL,
   extraSpacing            Integer16             OPTIONAL,
   totalBreakSpacing       Integer16             OPTIONAL,
   breakCount              Integer16             OPTIONAL,
   fontHeight              Integer16             OPTIONAL,
   fontWidth               Integer16             OPTIONAL,
   fontWeight              Integer16             OPTIONAL,
   textFlags1              TextAttributeFlags    OPTIONAL,
   fontID                  Integer16             OPTIONAL,
   textFlags2              ExtraTextFlags        OPTIONAL,
   clipLeft                Coordinate            OPTIONAL,
   clipTop                 Coordinate            OPTIONAL,
   clipRight               Coordinate            OPTIONAL,
   clipBottom              Coordinate            OPTIONAL,
   numberCodePoints        Integer8 (1..255)     OPTIONAL,
                           -- number of codepoints in codePointList; where deltaX values
                           -- are present maximum number of codepoints is 127
   codePointList           ASString (SIZE (1..255)) OPTIONAL,
   numberDeltaX            Integer8 (1..127)     OPTIONAL,
                           -- number of deltaX values in deltaXList
   deltaXList              SEQUENCE (SIZE (1..127)) OF Coordinate OPTIONAL
}

FrameOrder  ::= SEQUENCE
{
   header                  PrimaryOrderHeader, -- PrimaryOrderType = frame
   destLeft                Coordinate            OPTIONAL,
   destTop                 Coordinate            OPTIONAL,
   destWidth               Coordinate            OPTIONAL,
   destHeight              Coordinate            OPTIONAL,
   rop3                    ROP3                  OPTIONAL,
   backgroundColor         Color                 OPTIONAL,
```

```
   foregroundColor          Color              OPTIONAL,
   brush                    Brush              OPTIONAL
}

RectangleOrder ::= SEQUENCE
{
   header                   PrimaryOrderHeader, -- PrimaryOrderType = rectangle
   backMixMode              BackgroundMixMode  OPTIONAL,
   destLeft                 Coordinate         OPTIONAL,
   destTop                  Coordinate         OPTIONAL,
   destRight                Coordinate         OPTIONAL,
   destBottom               Coordinate         OPTIONAL,
   backgroundColor          Color              OPTIONAL,
   foregroundColor          Color              OPTIONAL,
   brush                    Brush              OPTIONAL,
   rop2                     ROP2               OPTIONAL,
   pen                      Pen                OPTIONAL
}

OpaqueRectangleOrder ::= SEQUENCE
{
   header                   PrimaryOrderHeader, -- PrimaryOrderType = opaqueRectangle
   destLeft                 Coordinate         OPTIONAL,
   destTop                  Coordinate         OPTIONAL,
   destWidth                Coordinate         OPTIONAL,
   destHeight               Coordinate         OPTIONAL,
   color                    Color              OPTIONAL
}

LineOrder ::= SEQUENCE
{
   header                   PrimaryOrderHeader, -- PrimaryOrderType = line
   backMixMode              BackgroundMixMode  OPTIONAL,
   startX                   Coordinate         OPTIONAL,
   startY                   Coordinate         OPTIONAL,
   endX                     Coordinate         OPTIONAL,
   endY                     Coordinate         OPTIONAL,
   backgroundColor          Color              OPTIONAL,
   rop2                     ROP2               OPTIONAL,
   pen                      Pen                OPTIONAL
}

DesktopSaveOrder ::= SEQUENCE
{
   header                   PrimaryOrderHeader, -- PrimaryOrderType = desktopSave
   saveOffset               Integer32          OPTIONAL,
   destLeft                 Coordinate         OPTIONAL,
   destTop                  Coordinate         OPTIONAL,
   destWidth                Coordinate         OPTIONAL,
   destHeight               Coordinate         OPTIONAL,
   action                   DesktopSaveAction  OPTIONAL
}

DesktopOriginOrder ::= SEQUENCE
{
   header                   PrimaryOrderHeader, -- PrimaryOrderType = desktopOrigin
   desktopLeft              Coordinate         OPTIONAL,
   desktopTop               Coordinate         OPTIONAL
}
```

```
PrimaryOrder ::= CHOICE
{
   destinationBlt                DestinationBltOrder,
   patternBlt                    PatternBltOrder,
   screenBlt                     ScreenBltOrder,
   memoryBlt                     MemoryBltOrder,
   memoryThreeWayBlt             MemoryThreeWayBltOrder,
   text                          TextOrder,
   extendedText                  ExtendedTextOrder,
   frame                         FrameOrder,
   rectangle                     RectangleOrder,
   line                          LineOrder,
   opaqueRectangle               OpaqueRectangleOrder,
   desktopSave                   DesktopSaveOrder,
   desktopOrigin                 DesktopOriginOrder
}

SecondaryOrder ::= CHOICE
{
   cacheBitmap                   CacheBitmapOrder,
   cacheColorTable               CacheColorTableOrder
}

UpdateOrder ::= CHOICE
{
   primaryOrder                  PrimaryOrder,
   secondaryOrder                SecondaryOrder
}

--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                          Begin AS PDU Definitions
--
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

ApplicationPDU ::= SEQUENCE
{
   shareDataHeader               ShareDataHeader, -- PDUType2 = application
   action                        ApplicationAction,
   numberApplications            Integer16,
   windowID                      WindowID
}

ConfirmActivePDU ::= SEQUENCE
{
   shareControlHeader            ShareControlHeader, -- PDUType = confirmActivePDU
   shareID                       ShareID,
   originatorID                  UserID,
   lengthSourceDescriptor        Integer16 (1..maxSourceDescriptor),
                                 -- length of sourceDescriptor in octets
                                 -- (including null terminator)
   lengthCombinedCapabilities    Integer16,
                                 -- length of combinedCapabilities in octets
   sourceDescriptor              T50String (SIZE (1..maxSourceDescriptor)),
   combinedCapabilities          CombinedCapabilities
}
```

```
ControlPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = control
   action                       ControlAction,
   grantID                      UserID
   controlID                    Integer32 (0..2147483647)
}

DeactivateAllPDU ::= SEQUENCE
{
   shareControlHeader           ShareControlHeader, -- PDUType = deactivateAllPDU
   shareID                      ShareID,
   lengthSourceDescriptor       Integer16 (1..maxSourceDescriptor),
                                -- length of sourceDescriptor in octets
                                -- (including null terminator)
   sourceDescriptor             T50String (SIZE (1..maxSourceDescriptor))
}

DeactivateOtherPDU ::= SEQUENCE
{
   shareControlHeader           ShareControlHeader, -- PDUType = deactivateOtherPDU
   shareID                      ShareID,
   deactivateID                 UserID,
   lengthSourceDescriptor       Integer16 (1..maxSourceDescriptor),
                                -- length of sourceDescriptor in octets
                                -- (including null terminator)
   sourceDescriptor             T50String (SIZE (1..maxSourceDescriptor))
}

DeactivateSelfPDU ::= SEQUENCE
{
   shareControlHeader           ShareControlHeader, -- PDUType = deactivateSelfPDU
   shareID                      ShareID
}

DemandActivePDU ::= SEQUENCE
{
   shareControlHeader           ShareControlHeader, -- PDUType = demandActivePDU
   shareID                      ShareID,
   lengthSourceDescriptor       Integer16 (1..maxSourceDescriptor),
                                -- length of sourceDescriptor in octets
                                -- (including null terminator)
   lengthCombinedCapabilities   Integer16,
                                -- length of combinedCapabilities in octets
   sourceDescriptor             T50String (SIZE (1..maxSourceDescriptor)),
   combinedCapabilities         CombinedCapabilities
}

FlowPDU ::= SEQUENCE
{
   flowMarker                   Integer16 ('8000'H),
                                -- distinguishes FlowPDUs from ASPDUs
                                -- containing ShareControlHeaders
   pad8bits                        Integer8 (0),
   pduTypeFlow                        PDUTypeFlow (flowResponsePDU |
                                                   (flowStopPDU    |
                                                   flowTestPDU     ),
   flowIdentifier               Integer8 (0..127),
   flowNumber                   Integer8,
                                -- shall be zero for PDUType FlowStopPDU
```

```
   pduSource                    UserID
                                -- MCS User ID of sending ASCE
}

FontPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = font
   numberFonts                  Integer16 (1..maxFonts),
                                -- number of FontAttributes in fontList
   entrySize                    Integer16,
   fontList                     SEQUENCE (SIZE (1..maxFonts)) OF FontAttribute
}

InputPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = input
   numberEvents                 Integer16,
                                -- number of InputEvents in eventList
   pad2octets                   Integer16 (0),
   eventList                    SEQUENCE (SIZE (1..maxInputEvents)) OF InputEvent
}

MediatedControlPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = mediatedControl
   action                       MediatedControlAction,
   passControlFlag              Boolean16,
   sendingReference             Integer16,
   originatorReference          Integer16,
   originatorID                 UserID
}

PointerPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = pointer
   messageType                  PointerMessageType,
   pad2octets                   Integer16 (0),
   pointerData CHOICE
   {
      systemPointerType         SystemPointerType,
      monoPointerAttribute      MonoPointerAttribute,
      colorPointerAttribute     ColorPointerAttribute,
      cachedPointerIndex        Integer16,
      pointerPosition           Point16
   }
}

RemoteSharePDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = remoteShare
   action                       RemoteShareAction,
   additionalData CHOICE
   {
      requestingID              UserID,
      pad2octets                Integer16 (0),
      denialCode                RemoteShareDenial
   },
   encryptedPassword            OCTET STRING (SIZE (1..maxPassword))
}
```

```
RequestActivePDU ::= SEQUENCE
{
   shareControlHeader              ShareControlHeader, -- PDUType = requestActivePDU
   lengthSourceDescriptor          Integer16 (1..maxSourceDescriptor),
                                   -- length of sourceDescriptor in octets
                                   -- (including null terminator)
   lengthCombinedCapabilities      Integer16,
                                   -- length of combinedCapabilities in octets
   sourceDescriptor                T50String (SIZE (1..maxSourceDescriptor)),
   combinedCapabilities            CombinedCapabilities
}

SynchronizePDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader, -- PDUType2 = synchronize
   messageType                     SynchronizeMessageType,
   targetUser                      UserID
}

UpdateBitmapPDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader, -- PDUType2=update
   updateType                      UpdateType (bitmap),
   pad2octets                      Integer16 (0),
   destLeft                        Coordinate16,
   destTop                         Coordinate16,
   destRight                       Coordinate16,
   destBottom                      Coordinate16,
   width                           Integer16,
   height                          Integer16,
   bitsPerPixel                    Integer16 (1|4|8),
   compressedFlag                  Boolean16,
   bitmapLength                    Integer16,
                                   -- length in octets of bitmapData (after any compression)
   bitmapData                      BitmapData
}

UpdateCapabilityPDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader, -- PDUType2 = updateCapability
   updateCapabilitySet             UpdateCapabilitySet
}

UpdateOrdersPDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader, -- PDUType2 = update
   updateType                      UpdateType (orders),
   pad2octetsA                     Integer16 (0),
   numberOrders                    Integer16,
                                   -- number of UpdateOrders in orderList
   pad2octetsB                     Integer16 (0),
   orderList                       SEQUENCE OF UpdateOrder
}

UpdatePalettePDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader, -- PDUType2 = update
   updateType                      UpdateType (palette),
   pad2octets                      Integer16 (0),
```

```
   numberColors                 Integer32 (16|256),
   palette                      SEQUENCE (SIZE (16|256)) OF Color
}

UpdateSynchronizePDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = update
   updateType                   UpdateType (synchronize),
   pad2octets                   Integer16 (0)
}

WindowActivationPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = windowActivation
   action                       WindowActivationAction,
   activationID                 Integer16,
   activationWindow             WindowID,
   activationPoint              Point16
}

WindowListPDU ::= SEQUENCE
{
   shareDataHeader              ShareDataHeader, -- PDUType2 = windowList
   messageType                  WindowListMessageType,
   pad2octetsA                  Integer16,
   numberWindows                Integer16,
                                -- number of WindowAttributes/Titles in lists
   listTime                     Integer16,
   listID                       Integer16,
   pad2octetsB                  Integer16,
   windowAttributeList          SEQUENCE OF WindowAttribute,
   windowTitleList              SEQUENCE OF WindowTitle
}

SharePDU    ::= CHOICE
{
   applicationPDU               ApplicationPDU,
   confirmActivePDU             ConfirmActivePDU,
   controlPDU                   ControlPDU,
   deactivateAllPDU             DeactivateAllPDU,
   deactivateOtherPDU           DeactivateOtherPDU,
   deactivateSelfPDU            DeactivateSelfPDU,
   demandActivePDU              DemandActivePDU,
   flowPDU                      FlowPDU,
   fontPDU                      FontPDU,
   inputPDU                     InputPDU,
   mediatedControlPDU           MediatedControlPDU,
   pointerPDU                   PointerPDU,
   remoteSharePDU               RemoteSharePDU,
   requestActivePDU             RequestActivePDU,
   synchronizePDU               SynchronizePDU,
   updateCapabilityPDU          UpdateCapabilityPDU,
   updateBitmapPDU              UpdateBitmapPDU,
   updateOrdersPDU              UpdateOrdersPDU,
   updateSynchronizePDU         UpdateSynchronizePDU,
   updatePalettePDU             UpdatePalettePDU,
   windowActivationPDU          WindowActivationPDU,
   windowListPDU                WindowListPDU
}
```

```
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                              End AS Definitions
--
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

END

9.2 Définition ASN.1 du mode de base

```
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                              Begin AS Definitions
--
-- The following base mode ASN.1 definitions are encoded using the BASIC
-- ALIGNED variant of the Packed Encoding Rules of Recommendation
-- X.691.
--
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

AS-PROTOCOL DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

```
IMPORTS         H221NonStandardIdentifier,
                Key,
                NonStandardParameter,
                UserID
FROM GCC-PROTOCOL;

-- NOTE: ======================================================
-- NOTE: All abstract types defined shall be exported
-- NOTE: ======================================================

-------------------------------------------------------------------------------------------------
-- Base Types
-------------------------------------------------------------------------------------------------

Coordinate8     ::= INTEGER (–128..127)
Coordinate16    ::= INTEGER (–32768..32767)
Integer8        ::= INTEGER (0..255)
Integer12       ::= INTEGER (0..4095)
Integer16       ::= INTEGER (0..65535)
Integer32       ::= INTEGER (0..4294967295)
Signed16        ::= INTEGER (–32768..32767)

ShareID         ::= Integer32
WindowID        ::= Integer32

T50String       ::= OCTET STRING (SIZE (0..255)) -- -- T.50 String
ASString        ::= OCTET STRING (SIZE (0..255)) -- -- AS Protocol CodePage String
```

```
--------------------------------------------------------------------------------------------------
-- Bit Flag Types
--------------------------------------------------------------------------------------------------

ExtraTextFlags ::= BIT STRING
{
   opaqueRectangle                (1),
   clipToRectangle                (2),
   deltaXPresent                  (15),
   ...
}

FontAttributeFlags ::= BIT STRING
{
   fixedPitch                     (0),
   fixedSize                      (1),
   ...
}

KeyboardFlags ::= BIT STRING
{
   right                          (0),
   quiet                          (12),
   down                           (14),
   release                        (15),
   ...
}

PointingDeviceFlags ::= BIT STRING
{
   move                           (11),
   button1                        (12),
   button2                        (13),
   button3                        (14),
   down                           (15),
   ...
}

TextAttributeFlags ::= BIT STRING
{
   italic                         (2),
   underline                      (3),
   strikeout                      (4),
   baselineStart                  (8),
   ...
}

WindowAttributeFlags ::= BIT STRING
{
   minimized                      (0),
   taggable                       (1),
   hosted                         (2),
   shadow                         (3),
   local                          (4),
   topmost                        (5),
   windowManagerMinimized         (16),
   windowManagerInvisible         (17),
   ...
}
```

```
-------------------------------------------------------------------------------------------------
-- General Types
-------------------------------------------------------------------------------------------------

ActivateWindowRequest ::= SEQUENCE
{
   activationWindow              WindowID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ActivationHelpKeyRequest ::= SEQUENCE
{
   activationWindow              WindowID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ActivationHelpIndexKeyRequest ::= SEQUENCE
{
   activationWindow              WindowID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ActivationHelpExtendedKeyRequest ::= SEQUENCE
{
   activationWindow              WindowID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

BackgroundMixMode ::= CHOICE
{
   transparent                   [1] NULL,
   opaque                        [2] NULL,
   nonStandardBackgroundMixMode  NonStandardParameter,
                                 -- Subject to capability negotiation.
   ...
}

BitmapData ::= CHOICE
{
   uncompressedBitmapData        [0] OCTET STRING,
   compressedBitmapData          [2] CompressedBitmapData,
   nonStandardBitmapData         NonStandardParameter,
                                 -- Subject to capability negotiation.
   ...
}

Brush ::= SEQUENCE
{
   originX                       Integer8          OPTIONAL,
   originY                       Integer8          OPTIONAL,
   style                         BrushStyle        OPTIONAL,
   hatch                         BrushHatch        OPTIONAL,
```

```
   pattern                          OCTET STRING (SIZE (7)) OPTIONAL,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

BrushHatch ::= CHOICE
{
   style                            HatchStyle,
   patternZero                      Integer8,
   nonStandardBrushHatch            NonStandardParameter,
                                    -- Subject to capability negotiation.
   ...
}

BrushStyle ::= CHOICE
{
   solid                            [0] NULL,
   null                             [1] NULL,
   hatched                          [2] NULL,
   pattern                          [3] NULL,
   nonStandardBrushStyle            NonStandardParameter,
                                    -- Subject to capability negotiation.
   ...
}

CloseWindowRequest ::= SEQUENCE
{
   activationWindow                 WindowID,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

Color ::= SEQUENCE
{
   c1                               Integer8,
                                    -- either R of RGB or subject to capability negotiation.
   c2                               Integer8,
                                    -- either G of RGB or subject to capability negotiation.
   c3                               Integer8
                                    -- either B of RGB or subject to capability negotiation
}

ColorAccuracyEnhancementRGB ::= CHOICE
{
   predefinedRGBSpace CHOICE
   {
      nonStandardRGBSpace           NonStandardParameter,
      ...
   },
   generalRGBParameters SEQUENCE
   {
      gamma                         REAL (0..MAX) OPTIONAL,
                                    -- Gamma value of the color space
      colorTemperature              INTEGER (0..MAX) OPTIONAL,
                                    -- Color temperature of the white point assumed by
                                    -- the color space (in degrees Kelvin)
      primaries SEQUENCE
```

```
      {
        red                             ColorCIExyChromaticity,
                                        -- CIE xy chromaticity coordinate of the red primary
        green                           ColorCIExyChromaticity,
                                        -- CIE xy chromaticity coordinate of the green primary
        blue                            ColorCIExyChromaticity
                                        -- CIE xy chromaticity coordinate of the blue primary
      } OPTIONAL,
      ...
    },
    ...
}

ColorCIExyChromaticity ::= SEQUENCE
{
    x                                   REAL (0..1), -- CIE normalized x component
    y                                   REAL (0..1)  -- CIE normalized y component
}

ColorPalette ::= CHOICE
{
    paletteRGB SEQUENCE
    {
        palette                         SEQUENCE (SIZE (16|256)) OF ColorRGB,
        enhancement                     ColorAccuracyEnhancementRGB OPTIONAL,
        ...
    },
    nonStandardPalette                  NonStandardParameter,
    ...
}

ColorPointerAttribute ::= SEQUENCE
{
    cacheIndex                          Integer16,
    hotSpot                             Point16,
    width                               Integer16,
    height                              Integer16,
    colorPointerData                    OCTET STRING,
    nonStandardParameters               SEQUENCE OF NonStandardParameter OPTIONAL,
                                        -- Subject to capability negotiation.
    ...
}

ColorRGB ::= SEQUENCE
{
    red                                 Integer8,
    green                               Integer8,
    blue                                Integer8
}

ColorSpaceSpecifier ::= CHOICE
{
    colorSpaceDefault                   NULL,
                                        -- Default color space is RGB without accuracy enhancement
    colorSpaceRGB                       ColorAccuracyEnhancementRGB,
    nonStandardColorSpace               NonStandardParameter,
                                        -- Subject to capability negotiation.
    ...
}
```

```
CompressedBitmapData ::= SEQUENCE
{
   mainBodySize                  Integer16,
   rowSize                       Integer16,
   uncompressedSize              Integer16,
   compressedBitmap              OCTET STRING,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ConfirmDetachResponse ::= SEQUENCE
{
   passControlFlag               BOOLEAN,
   sendingReference              Integer16,
   originatorReference           Integer16,
   originatorID                  UserID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ConfirmRemoteShare ::= SEQUENCE
{
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ConfirmTakeResponse ::= SEQUENCE
{
   passControlFlag               BOOLEAN,
   sendingReference              Integer16,
   originatorReference           Integer16,
   originatorID                  UserID,
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

ControlPriority ::= CHOICE
{
   always                        [1] NULL,
   never                         [2] NULL,
   confirm                       [3] NULL,
   nonStandardControlPriority    NonStandardParameter,
                                 -- Subject to capability negotiation.
   ...
}

Cooperate ::= SEQUENCE
{
   nonStandardParameters         SEQUENCE OF NonStandardParameter OPTIONAL,
                                 -- Subject to capability negotiation.
   ...
}

Coordinate  ::= CHOICE
{
   absolute                      Coordinate16,
   delta                         Coordinate8,
```

```
   nonStandardCoordinate          NonStandardParameter,
                                  -- Subject to capability negotiation.
   ...
}

DesktopSaveAction ::= CHOICE
{
   desktopSave                    [0] NULL,
   desktopRestore                 [1] NULL,
   nonStandardDesktopSaveAction   NonStandardParameter,
                                  -- Subject to capability negotiation.
   ...
}

DenyDetachResponse ::= SEQUENCE
{
   passControlFlag                BOOLEAN,
   sendingReference               Integer16,
   originatorReference            Integer16,
   originatorID                   UserID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

DenyPassResponse ::= SEQUENCE
{
   passControlFlag                BOOLEAN,
   sendingReference               Integer16,
   originatorReference            Integer16,
   originatorID                   UserID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

DenyTakeResponse ::= SEQUENCE
{
   passControlFlag                BOOLEAN,
   sendingReference               Integer16,
   originatorReference            Integer16,
   originatorID                   UserID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

DenyRemoteDetachResponse ::= SEQUENCE
{
   passControlFlag                BOOLEAN,
   sendingReference               Integer16,
   originatorReference            Integer16,
   originatorID                   UserID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}
```

```
DenyRemoteShare ::= CHOICE
{
   remoteShareDenial          RemoteShareDenial,
   nonStandardDenial          NonStandardParameter,
                              -- Subject to capability negotiation.
   ...
}

Detach ::= SEQUENCE
{
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

DetachRequest ::= SEQUENCE
{
   passControlFlag            BOOLEAN,
   sendingReference           Integer16,
   originatorID               UserID,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

FontAttribute ::= SEQUENCE
{
   faceName                   T50String,
   fontFlags                  FontAttributeFlags,
   averageWidth               Integer16,
   height                     Integer16,
   aspectX                    Integer16,
   aspectY                    Integer16,
   signature1                 Integer8,
   signature2                 Integer8,
   signature3                 Integer16,
   codePage                   FontCodePage,
   ascent                     Integer16,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

FontCodePage ::= CHOICE
{
   allCodePoints              [0] NULL,
   coreCodePoints             [255] NULL,
   nonStandardFontCodePage    NonStandardParameter,
                              -- Subject to capability negotiation.
   ...
}

GeneralCompressionSpecifier ::= CHOICE
{
   v42bisCompression          V42bisCompression,
   nonStandardCompression     NonStandardParameter,
   ...
}
```

```
GrantControl ::= SEQUENCE
{
   grantID                    UserID,
   controlID                  INTEGER (0..2147483647),
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

HatchStyle ::= CHOICE
{
   horizontal                 [0] NULL,
   vertical                   [1] NULL,
   forward                    [2] NULL,
   backward                   [3] NULL,
   cross                      [4] NULL,
   diagonal                   [5] NULL,
   nonStandardHatchStyle      NonStandardParameter,
                              -- Subject to capability negotiation.
   ...
}

HostedWindowActiveIndication ::= SEQUENCE
{
   activationID               Integer16,
   activationWindow           WindowID,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

HostedWindowInvisibleIndication ::= SEQUENCE
{
   activationID               Integer16,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

LocalWindowActiveIndication ::= SEQUENCE
{
   activationID               Integer16,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

MonoPointerAttribute ::= SEQUENCE
{
   hotSpot                    Point16,
   width                      Integer16,
   height                     Integer16,
   monoPointerData            OCTET STRING,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

NotifyHostedApplications ::= SEQUENCE
{
   numberApplications         Integer16,
```

```
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

PassControlRequest ::= SEQUENCE
{
   passControlFlag                BOOLEAN,
   sendingReference               Integer16,
   originatorID                   UserID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

Pen ::= SEQUENCE
{
   style                          PenStyle          OPTIONAL,
   width                          Integer8 (1)        OPTIONAL,
   color                          Color             OPTIONAL,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

PenStyle ::= CHOICE
{
   solid                          [0] NULL,
   dashed                         [1] NULL,
   dotted                         [2] NULL,
   dash-dot                       [3] NULL,
   dash-dot-dot                   [4] NULL,
   null                           [5] NULL,
   nonStandardPenStyle            NonStandardParameter,
                                  -- Subject to capability negotiation.
   ...
}

Point16 ::= SEQUENCE
{
   x                              Coordinate16,
   y                              Coordinate16
}

PointerDeviceCaptureIndication ::= SEQUENCE
{
   activationID                   Integer16,
   activationWindow               WindowID,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

Rectangle16 ::= SEQUENCE
{
   left                           Coordinate16,
   top                            Coordinate16,
   right                          Coordinate16,
   bottom                         Coordinate16
}
```

```
RemoteDetachRequest ::= SEQUENCE
{
   passControlFlag                  BOOLEAN,
   sendingReference                 Integer16,
   originatorID                     UserID,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

RemoteShareDenial ::= CHOICE
{
   incorrectPassword                [1] NULL,
   remoteShareNotEnabled            [2] NULL,
   remoteShareInOperationIncoming   [3] NULL,
   remoteShareInOperationOutgoing   [4] NULL,
   nonStandardRemoteShareDenial     NonStandardParameter,
                                    -- Subject to capability negotiation.
   ...
}

RequestControl ::= SEQUENCE
{
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

RequestRemoteShare ::= SEQUENCE
{
   requestingID                     UserID,
   encryptedPassword                OCTET STRING,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

RestoreWindowRequest ::= SEQUENCE
{
   activationWindow                 WindowID,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

ROP2 ::= Integer8
{
   r2BLACK                          (1),
   r2DPon                           (2),
   r2DPna                           (3),
   r2Pn                             (4),
   r2PDna                           (5),
   r2Dn                             (6),
   r2DPx                            (7),
   r2DPan                           (8),
   r2DPa                            (9),
   r2DPxn                           (10),
   r2D                              (11),
   r2DPno                           (12),
   r2P                              (13),
   r2PDno                           (14),
```

```
   r2DPo                    (15),
   r2WHITE                  (16),
   ...
}

ROP3 ::= Integer8
{
   r3BLACK                  ('00'H),
   r3DPSoon                 ('01'H),
   r3DPSona                 ('02'H),
   r3PSon                   ('03'H),
   r3SDPona                 ('04'H),
   r3DPon                   ('05'H),
   r3PDSxnon                ('06'H),
   r3PDSaon                 ('07'H),
   r3SDPnaa                 ('08'H),
   r3PDSxon                 ('09'H),
   r3DPna                   ('0A'H),
   r3PSDnaon                ('0B'H),
   r3SPna                   ('0C'H),
   r3PDSnaon                ('0D'H),
   r3PDSonon                ('0E'H),
   r3Pn                     ('0F'H),
   r3PDSona                 ('10'H),
   r3DSon                   ('11'H),
   r3SDPxnon                ('12'H),
   r3SDPaon                 ('13'H),
   r3DPSxnon                ('14'H),
   r3DPSaon                 ('15'H),
   r3PSDPSanaxx             ('16'H),
   r3SSPxDSxaxn             ('17'H),
   r3SPxPDxa                ('18'H),
   r3SDPSanaxn              ('19'H),
   r3PDSPaox                ('1A'H),
   r3SDPSxaxn               ('1B'H),
   r3PSDPaox                ('1C'H),
   r3DSPDxaxn               ('1D'H),
   r3PDSox                  ('1E'H),
   r3PDSoan                 ('1F'H),
   r3DPSnaa                 ('20'H),
   r3SDPxon                 ('21'H),
   r3DSna                   ('22'H),
   r3SPDnaon                ('23'H),
   r3SPxDSxa                ('24'H),
   r3PDSPanaxn              ('25'H),
   r3SDPSaox                ('26'H),
   r3SDPSxnox               ('27'H),
   r3DPSxa                  ('28'H),
   r3PSDPSaoxxn             ('29'H),
   r3DPSana                 ('2A'H),
   r3SSPxPDxaxn             ('2B'H),
   r3SPDSoax                ('2C'H),
   r3PSDnox                 ('2D'H),
   r3PSDPxox                ('2E'H),
   r3PSDnoan                ('2F'H),
   r3PSna                   ('30'H),
   r3SDPnaon                ('31'H),
   r3SDPSoox                ('32'H),
   r3Sn                     ('33'H),
   r3SPDSaox                ('34'H),
   r3SPDSxnox               ('35'H),
```

r3SDPox **('36'H),**
r3SDPoan **('37'H),**
r3PSDPoax **('38'H),**
r3SPDnox **('39'H),**
r3SPDSxox **('3A'H),**
r3SPDnoan **('3B'H),**
r3PSx **('3C'H),**
r3SPDSonox **('3D'H),**
r3SPDSnaox **('3E'H),**
r3PSan **('3F'H),**
r3PSDnaa **('40'H),**
r3DPSxon **('41'H),**
r3SDxPDxa **('42'H),**
r3SPDSanaxn **('43'H),**
r3SDna **('44'H),**
r3DPSnaon **('45'H),**
r3DSPDaox **('46'H),**
r3PSDPxaxn **('47'H),**
r3SDPxa **('48'H),**
r3PDSPDaoxxn **('49'H),**
r3DPSDoax **('4A'H),**
r3PDSnox **('4B'H),**
r3SDPana **('4C'H),**
r3SSPxDSxoxn **('4D'H),**
r3PDSPxox **('4E'H),**
r3PDSnoan **('4F'H),**
r3PDna **('50'H),**
r3DSPnaon **('51'H),**
r3DPSDaox **('52'H),**
r3SPDSxaxn **('53'H),**
r3DPSonon **('54'H),**
r3Dn **('55'H),**
r3DPSox **('56'H),**
r3DPSoan **('57'H),**
r3PDSPoax **('58'H),**
r3DPSnox **('59'H),**
r3DPx **('5A'H),**
r3DPSDonox **('5B'H),**
r3DPSDxox **('5C'H),**
r3DPSnoan **('5D'H),**
r3DPSDnaox **('5E'H),**
r3DPan **('5F'H),**
r3PDSxa **('60'H),**
r3DSPDSaoxxn **('61'H),**
r3DSPDoax **('62'H),**
r3SDPnox **('63'H),**
r3SDPSoax **('64'H),**
r3DSPnox **('65'H),**
r3DSx **('66'H),**
r3SDPSonox **('67'H),**
r3DSPDSonoxxn **('68'H),**
r3PDSxxn **('69'H),**
r3DPSax **('6A'H),**
r3PSDPSoaxxn **('6B'H),**
r3SDPax **('6C'H),**
r3PDSPDoaxxn **('6D'H),**
r3SDPSnoax **('6E'H),**
r3PDSxnan **('6F'H),**
r3PDSana **('70'H),**
r3SSDxPDxaxn **('71'H),**
r3SDPSxox **('72'H),**

r3SDPnoan ('73'H),
r3DSPDxox ('74'H),
r3DSPnoan ('75'H),
r3SDPSnaox ('76'H),
r3DSan ('77'H),
r3PDSax ('78'H),
r3DSPDSoaxxn ('79'H),
r3DPSDnoax ('7A'H),
r3SDPxnan ('7B'H),
r3SPDSnoax ('7C'H),
r3DPSxnan ('7D'H),
r3SPxDSxo ('7E'H),
r3DPSaan ('7F'H),
r3DPSaa ('80'H),
r3SPxDSxon ('81'H),
r3DPSxna ('82'H),
r3SPDSnoaxn ('83'H),
r3SDPxna ('84'H),
r3PDSPnoaxn ('85'H),
r3DSPDSoaxx ('86'H),
r3PDSaxn ('87'H),
r3DSa ('88'H),
r3SDPSnaoxn ('89'H),
r3DSPnoa ('8A'H),
r3DSPDxoxn ('8B'H),
r3SDPnoa ('8C'H),
r3SDPSxoxn ('8D'H),
r3SSDxPDxax ('8E'H),
r3PDSanan ('8F'H),
r3PDSxna ('90'H),
r3SDPSnoaxn ('91'H),
r3DPSDPoaxx ('92'H),
r3SPDaxn ('93'H),
r3PSDPSoaxx ('94'H),
r3DPSaxn ('95'H),
r3DPSxx ('96'H),
r3PSDPSonoxx ('97'H),
r3SDPSonoxn ('98'H),
r3DSxn ('99'H),
r3DPSnax ('9A'H),
r3SDPSoaxn ('9B'H),
r3SPDnax ('9C'H),
r3DSPDoaxn ('9D'H),
r3DSPDSaoxx ('9E'H),
r3PDSxan ('9F'H),
r3DPa ('A0'H),
r3PDSPnaoxn ('A1'H),
r3DPSnoa ('A2'H),
r3DPSDxoxn ('A3'H),
r3PDSPonoxn ('A4'H),
r3PDxn ('A5'H),
r3DSPnax ('A6'H),
r3PDSPoaxn ('A7'H),
r3DPSoa ('A8'H),
r3DPSoxn ('A9'H),
r3D ('AA'H),
r3DPSono ('AB'H),
r3SPDSxax ('AC'H),
r3DPSDaoxn ('AD'H),
r3DSPnao ('AE'H),
r3DPno ('AF'H),

r3PDSnoa ('B0'H),
r3PDSPxoxn ('B1'H),
r3SSPxDSxox ('B2'H),
r3SDPanan ('B3'H),
r3PSDnax ('B4'H),
r3DPSDoaxn ('B5'H),
r3DPSDPaoxx ('B6'H),
r3SDPxan ('B7'H),
r3PSDPxax ('B8'H),
r3DSPDaoxn ('B9'H),
r3DPSnao ('BA'H),
r3DSno ('BB'H),
r3SPDSanax ('BC'H),
r3SDxPDxan ('BD'H),
r3DPSxo ('BE'H),
r3DPSano ('BF'H),
r3PSa ('C0'H),
r3SPDSnaoxn ('C1'H),
r3SPDSonoxn ('C2'H),
r3PSxn ('C3'H),
r3SPDnoa ('C4'H),
r3SPDSxoxn ('C5'H),
r3SDPnax ('C6'H),
r3PSDPoaxn ('C7'H),
r3SDPoa ('C8'H),
r3SPDoxn ('C9'H),
r3DPSDxax ('CA'H),
r3SPDSaoxn ('CB'H),
r3S ('CC'H),
r3SDPono ('CD'H),
r3SDPnao ('CE'H),
r3SPno ('CF'H),
r3PSDnoa ('D0'H),
r3PSDPxoxn ('D1'H),
r3PDSnax ('D2'H),
r3SPDSoaxn ('D3'H),
r3SSPxPDxax ('D4'H),
r3DPSanan ('D5'H),
r3PSDPSaoxx ('D6'H),
r3DPSxan ('D7'H),
r3PDSPxax ('D8'H),
r3SDPSaoxn ('D9'H),
r3DPSDanax ('DA'H),
r3SPxDSxan ('DB'H),
r3SPDnao ('DC'H),
r3SDno ('DD'H),
r3SDPxo ('DE'H),
r3SDPano ('DF'H),
r3PDSoa ('E0'H),
r3PDSoxn ('E1'H),
r3DSPDxax ('E2'H),
r3PSDPaoxn ('E3'H),
r3SDPSxax ('E4'H),
r3PDSPaoxn ('E5'H),
r3SDPSanax ('E6'H),
r3SPxPDxan ('E7'H),
r3SSPxDSxax ('E8'H),
r3DSPDSanaxxn ('E9'H),
r3DPSao ('EA'H),
r3DPSxno ('EB'H),
r3SDPao ('EC'H),

```
   r3SDPxno                        ('ED'H),
   r3DSo                           ('EE'H),
   r3SDPnoo                        ('EF'H),
   r3P                             ('F0'H),
   r3PDSono                        ('F1'H),
   r3PDSnao                        ('F2'H),
   r3PSno                          ('F3'H),
   r3PSDnao                        ('F4'H),
   r3PDno                          ('F5'H),
   r3PDSxo                         ('F6'H),
   r3PDSano                        ('F7'H),
   r3PDSao                         ('F8'H),
   r3PDSxno                        ('F9'H),
   r3DPo                           ('FA'H),
   r3DPSnoo                        ('FB'H),
   r3PSo                           ('FC'H),
   r3PSDnoo                        ('FD'H),
   r3DPSoo                         ('FE'H),
   r3WHITE                         ('FF'H)
}

SystemPointerType ::= CHOICE
{
   null                            [0] NULL,
   default                         [32512] NULL,
   nonStandardSystemPointerValue   NonStandardParameter,
                                   -- Subject to capability negotiation.
   ...
}

TakeControlRequest ::= SEQUENCE
{
   passControlFlag                 BOOLEAN,
   sendingReference                Integer16,
   originatorReference             Integer16,
   originatorID                    UserID,
   nonStandardParameters           SEQUENCE OF NonStandardParameter OPTIONAL,
                                   -- Subject to capability negotiation.
   ...
}

UnhostApplication ::= SEQUENCE
{
   windowID                        WindowID,
   nonStandardParameters           SEQUENCE OF NonStandardParameter OPTIONAL,
                                   -- Subject to capability negotiation.
   ...
}

V42bisCompression ::= SEQUENCE
{
   p1                              INTEGER (512..65535) OPTIONAL,
   p2                              INTEGER (6..250) OPTIONAL,
   ...
}

WindowAttribute ::= SEQUENCE
{
   windowID                        WindowID,
   windowExtra                     Integer32,
   windowOwner                     WindowID,
```

```
   windowFlags                  WindowAttributeFlags,
   windowRectangle              Rectangle16,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

WindowManagerMenuRequest ::= SEQUENCE
{
   activationWindow             WindowID,
   activationPoint              Point16,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

WindowTitle ::= CHOICE
{
   noTitle                      Integer8 (255),
   titleString                  T50String,
   nonStandardWindowTitle       NonStandardParameter,
                                -- Subject to capability negotiation.
   ...
}

-------------------------------------------------------------------------------------------------
-- Input Types
-------------------------------------------------------------------------------------------------

InputEvent ::= CHOICE
{
   pointingDeviceEvent          [32769] PointingDeviceEvent,
   codePointEvent               [1] CodePointEvent,
   virtualKeyEvent              [2] VirtualKeyEvent,
   synchronizeEvent             [0] SynchronizeEvent,
   nonStandardInputEvent        NonStandardParameter,
                                -- Subject to capability negotiation.
   ...
}

CodePointEvent ::= SEQUENCE
{
   eventTime                    Integer32,
   keyboardFlags                KeyboardFlags,
   codePoint                    Integer16,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

VirtualKeyEvent ::= SEQUENCE
{
   eventTime                    Integer32,
   keyboardFlags                KeyboardFlags,
   virtualKey                   Integer16,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}
```

```
PointingDeviceEvent ::= SEQUENCE
{
   eventTime                  Integer32,
   pointingDeviceFlags        PointingDeviceFlags,
   pointingDeviceX            Coordinate16,
   pointingDeviceY            Coordinate16,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

SynchronizeEvent ::= SEQUENCE
{
   eventTime                  Integer32,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

-------------------------------------------------------------------------------------------------
-- Common Header Types
-------------------------------------------------------------------------------------------------

PrimaryOrderHeader  ::= SEQUENCE
{
   boundsLeft                 Coordinate OPTIONAL,
   boundsTop                  Coordinate OPTIONAL,
   boundsRight                Coordinate OPTIONAL,
   boundsBottom               Coordinate OPTIONAL,
   ...
}

ShareDataHeader ::= SEQUENCE
{
   shareID                    ShareID,
   generalCompressionSpecifier GeneralCompressionSpecifier OPTIONAL,
   ...
}

-------------------------------------------------------------------------------------------------
-- Order Types
-------------------------------------------------------------------------------------------------

DestinationBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader,
   destLeft                   Coordinate        OPTIONAL,
   destTop                    Coordinate        OPTIONAL,
   destWidth                  Coordinate        OPTIONAL,
   destHeight                 Coordinate        OPTIONAL,
   rop3                       ROP3              OPTIONAL,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

PatternBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader,
   destLeft                   Coordinate        OPTIONAL,
   destTop                    Coordinate        OPTIONAL,
```

```
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
   rop3                       ROP3                OPTIONAL,
   backgroundColor            Color               OPTIONAL,
   foregroundColor            Color               OPTIONAL,
   brush                      Brush               OPTIONAL,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

ScreenBltOrder ::= SEQUENCE
{
   header                     PrimaryOrderHeader,
   destLeft                   Coordinate          OPTIONAL,
   destTop                    Coordinate          OPTIONAL,
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
   rop3                       ROP3                OPTIONAL,
   sourceX                    Coordinate          OPTIONAL,
   sourceY                    Coordinate          OPTIONAL,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

CacheBitmapOrder ::= SEQUENCE
{
   cacheId                    INTEGER (0..2),
   bitmapWidth                Integer8,
   bitmapHeight               Integer8,
   bitmapBitsPerPel           INTEGER (1|4|8),
   cacheIndex                 Integer16,
   bitmapData                 BitmapData,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

CacheColorTableOrder ::= SEQUENCE
{
   cacheIndex                 Integer8,
   colorTable                 ColorPalette,
   nonStandardParameters      SEQUENCE OF NonStandardParameter OPTIONAL,
                              -- Subject to capability negotiation.
   ...
}

MemoryBltOrder  ::= SEQUENCE
{
   header                     PrimaryOrderHeader,
   colorTableCacheIndex       Integer8            OPTIONAL,
   bitmapCacheID              Integer8            OPTIONAL,
   destLeft                   Coordinate          OPTIONAL,
   destTop                    Coordinate          OPTIONAL,
   destWidth                  Coordinate          OPTIONAL,
   destHeight                 Coordinate          OPTIONAL,
   rop3                       ROP3                OPTIONAL,
   sourceX                    Coordinate          OPTIONAL,
   sourceY                    Coordinate          OPTIONAL,
   bitmapCacheIndex           Integer16           OPTIONAL,
```

```
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

MemoryThreeWayBltOrder ::= SEQUENCE
{
   header                       PrimaryOrderHeader,
   colorTableCacheIndex         Integer8            OPTIONAL,
   bitmapCacheID                Integer8            OPTIONAL,
   destLeft                     Coordinate          OPTIONAL,
   destTop                      Coordinate          OPTIONAL,
   destWidth                    Coordinate          OPTIONAL,
   destHeight                   Coordinate          OPTIONAL,
   rop3                         ROP3                OPTIONAL,
   sourceX                      Coordinate          OPTIONAL,
   sourceY                      Coordinate          OPTIONAL,
   backgroundColor              Color               OPTIONAL,
   foregroundColor              Color               OPTIONAL,
   brush                        Brush               OPTIONAL,
   bitmapCacheIndex             Integer16           OPTIONAL,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

TextOrder ::= SEQUENCE
{
   header                       PrimaryOrderHeader,
   backMixMode                  BackgroundMixMode   OPTIONAL,
   startX                       Coordinate          OPTIONAL,
   startY                       Coordinate          OPTIONAL,
   backgroundColor              Color               OPTIONAL,
   foregroundColor              Color               OPTIONAL,
   extraSpacing                 Integer16           OPTIONAL,
   totalBreakSpacing            Integer16           OPTIONAL,
   breakCount                   Integer16           OPTIONAL,
   fontHeight                   Integer16           OPTIONAL,
   fontWidth                    Integer16           OPTIONAL,
   fontWeight                   Integer16           OPTIONAL,
   textFlags                    TextAttributeFlags  OPTIONAL,
   fontID                       Integer16           OPTIONAL,
   codePointList                ASString (SIZE (1..255)) OPTIONAL,
   nonStandardParameters        SEQUENCE OF NonStandardParameter OPTIONAL,
                                -- Subject to capability negotiation.
   ...
}

ExtendedTextOrder ::= SEQUENCE
{
   header                       PrimaryOrderHeader,
   backMixMode                  BackgroundMixMode   OPTIONAL,
   startX                       Coordinate          OPTIONAL,
   startY                       Coordinate          OPTIONAL,
   backgroundColor              Color               OPTIONAL,
   foregroundColor              Color               OPTIONAL,
   extraSpacing                 Integer16           OPTIONAL,
   totalBreakSpacing            Integer16           OPTIONAL,
   breakCount                   Integer16           OPTIONAL,
   fontHeight                   Integer16           OPTIONAL,
   fontWidth                    Integer16           OPTIONAL,
```

```
   fontWeight                  Integer16                  OPTIONAL,
   textFlags1                  TextAttributeFlags         OPTIONAL,
   fontID                      Integer16                  OPTIONAL,
   textFlags2                  ExtraTextFlags             OPTIONAL,
   clipLeft                    Coordinate                 OPTIONAL,
   clipTop                     Coordinate                 OPTIONAL,
   clipRight                   Coordinate                 OPTIONAL,
   clipBottom                  Coordinate                 OPTIONAL,
   codePointList               ASString (SIZE (1..255))   OPTIONAL,
   deltaXList                  SEQUENCE (SIZE (1..127)) OF Coordinate OPTIONAL,
   nonStandardParameters       SEQUENCE OF NonStandardParameter OPTIONAL,
                               -- Subject to capability negotiation.
   ...
}

FrameOrder  ::= SEQUENCE
{
   header                      PrimaryOrderHeader,
   destLeft                    Coordinate                 OPTIONAL,
   destTop                     Coordinate                 OPTIONAL,
   destWidth                   Coordinate                 OPTIONAL,
   destHeight                  Coordinate                 OPTIONAL,
   rop3                        ROP3                       OPTIONAL,
   backgroundColor             Color                      OPTIONAL,
   foregroundColor             Color                      OPTIONAL,
   brush                       Brush                      OPTIONAL,
   nonStandardParameters       SEQUENCE OF NonStandardParameter OPTIONAL,
                               -- Subject to capability negotiation.
   ...
}

RectangleOrder ::= SEQUENCE
{
   header                      PrimaryOrderHeader,
   backMixMode                 BackgroundMixMode          OPTIONAL,
   destLeft                    Coordinate                 OPTIONAL,
   destTop                     Coordinate                 OPTIONAL,
   destRight                   Coordinate                 OPTIONAL,
   destBottom                  Coordinate                 OPTIONAL,
   backgroundColor             Color                      OPTIONAL,
   foregroundColor             Color                      OPTIONAL,
   brush                       Brush                      OPTIONAL,
   rop2                        ROP2                       OPTIONAL,
   pen                         Pen                        OPTIONAL,
   nonStandardParameters       SEQUENCE OF NonStandardParameter OPTIONAL,
                               -- Subject to capability negotiation.
   ...
}

OpaqueRectangleOrder ::= SEQUENCE
{
   header                      PrimaryOrderHeader,
   destLeft                    Coordinate                 OPTIONAL,
   destTop                     Coordinate                 OPTIONAL,
   destWidth                   Coordinate                 OPTIONAL,
   destHeight                  Coordinate                 OPTIONAL,
   color                       Color                      OPTIONAL,
   nonStandardParameters       SEQUENCE OF NonStandardParameter OPTIONAL,
                               -- Subject to capability negotiation.
   ...
}
```

```
LineOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader,
   backMixMode             BackgroundMixMode     OPTIONAL,
   startX                  Coordinate            OPTIONAL,
   startY                  Coordinate            OPTIONAL,
   endX                    Coordinate            OPTIONAL,
   endY                    Coordinate            OPTIONAL,
   backgroundColor         Color                 OPTIONAL,
   rop2                    ROP2                  OPTIONAL,
   pen                     Pen                   OPTIONAL,
   nonStandardParameters   SEQUENCE OF NonStandardParameter OPTIONAL,
                           -- Subject to capability negotiation.
   ...
}

DesktopSaveOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader,
   saveOffset              Integer32             OPTIONAL,
   destLeft                Coordinate            OPTIONAL,
   destTop                 Coordinate            OPTIONAL,
   destWidth               Coordinate            OPTIONAL,
   destHeight              Coordinate            OPTIONAL,
   action                  DesktopSaveAction     OPTIONAL,
   nonStandardParameters   SEQUENCE OF NonStandardParameter OPTIONAL,
                           -- Subject to capability negotiation.
   ...
}

DesktopOriginOrder ::= SEQUENCE
{
   header                  PrimaryOrderHeader,
   desktopLeft             Coordinate            OPTIONAL,
   desktopTop              Coordinate            OPTIONAL,
   nonStandardParameters   SEQUENCE OF NonStandardParameter OPTIONAL,
                           -- Subject to capability negotiation.
   ...
}

ColorSpaceOrder ::= SEQUENCE
{
   colorSpace              ColorSpaceSpecifier,
   nonStandardParameters   SEQUENCE OF NonStandardParameter OPTIONAL,
                           -- Subject to capability negotiation.
   ...
}

PrimaryOrder ::= CHOICE
{
   destinationBlt          [0] DestinationBltOrder,
   patternBlt              [1] PatternBltOrder,
   screenBlt               [2] ScreenBltOrder,
   memoryBlt               [13] MemoryBltOrder,
   memoryThreeWayBlt       [14] MemoryThreeWayBltOrder,
   text                    [5] TextOrder,
   extendedText            [6] ExtendedTextOrder,
   frame                   [9] FrameOrder,
   rectangle               [7] RectangleOrder,
   line                    [8] LineOrder,
```

```
   opaqueRectangle                 [10] OpaqueRectangleOrder,
   desktopSave                     [11] DesktopSaveOrder,
   desktopOrigin                   [32] DesktopOriginOrder,
   nonStandardPrimaryOrder         NonStandardParameter,
                                   -- Subject to capability negotiation.
   ...
}

SecondaryOrder ::= CHOICE
{
   cacheBitmap                     [0] CacheBitmapOrder,
   cacheColorTable                 [1] CacheColorTableOrder,
   colorSpaceOrder                 ColorSpaceOrder,
   nonStandardSecondaryOrder       NonStandardParameter,
                                   -- Subject to capability negotiation.
   ...
}

UpdateOrder ::= CHOICE
{
   primaryOrder                    PrimaryOrder,
   secondaryOrder                  SecondaryOrder,
   nonStandardOrder                NonStandardParameter,
                                   -- Subject to capability negotiation.
   ...
}

--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                          Begin AS PDU Definitions
--
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

ApplicationPDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader,
   action CHOICE
   {
      notifyHostedApplications     [1] NotifyHostedApplications,
      unhostApplication            [2] UnhostApplication,
      nonStandardAction            NonStandardParameter,
                                   -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters           SEQUENCE OF NonStandardParameter OPTIONAL,
                                   -- Subject to capability negotiation.
   ...
}

ControlPDU ::= SEQUENCE
{
   shareDataHeader                 ShareDataHeader,
   action CHOICE
   {
      requestControl               [1] RequestControl,
      grantControl                 [2] GrantControl,
      detach                       [3] Detach,
      cooperate                    [4] Cooperate,
```

```
      nonStandardAction             NonStandardParameter,
                                    -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

FlowResponsePDU ::= SEQUENCE
{
   flowIdentifier                   INTEGER (0..127),
   flowNumber                       Integer8,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

FlowStopPDU ::= SEQUENCE
{
   flowIdentifier                   INTEGER (0..127),
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

FlowTestPDU ::= SEQUENCE
{
   flowIdentifier                   INTEGER (0..127),
   flowNumber                       Integer8,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

FontPDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   fontList                         SEQUENCE OF FontAttribute,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

InputPDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   eventList                        SEQUENCE OF InputEvent,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

MediatedControlPDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   action CHOICE
   {
      takeControlRequest            [1] TakeControlRequest,
      passControlRequest            [2] PassControlRequest,
      detachRequest                 [3] DetachRequest,
```

```
      confirmTakeResponse           [5] ConfirmTakeResponse,
      denyTakeResponse              [6] DenyTakeResponse,
      confirmDetachResponse         [7] ConfirmDetachResponse,
      denyDetachResponse            [8] DenyDetachResponse,
      denyPassResponse              [9] DenyPassResponse,
      remoteDetachRequest           [10] RemoteDetachRequest,
      denyRemoteDetachResponse      [11] DenyRemoteDetachResponse,
      nonStandardAction             NonStandardParameter,
                                    -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

PointerPDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   pointerData CHOICE
   {
      systemPointerType             [1] SystemPointerType,
      monoPointerAttribute          [2] MonoPointerAttribute,
      colorPointerAttribute         [6] ColorPointerAttribute,
      cachedPointerIndex            [7] Integer16,
      pointerPosition               [3] Point16,
      nonStandardPointer            NonStandardParameter,
                                    -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

RemoteSharePDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   action CHOICE
   {
      requestRemoteShare            [1] RequestRemoteShare,
      confirmRemoteShare            [2] ConfirmRemoteShare,
      denyRemoteShare               [3] DenyRemoteShare,
      nonStandardAction             NonStandardParameter,
                                    -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

SynchronizePDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   targetUser                       UserID,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}
```

```
UpdateBitmapPDU ::= SEQUENCE
{
   shareDataHeader                ShareDataHeader,
   destLeft                       Coordinate16,
   destTop                        Coordinate16,
   destRight                      Coordinate16,
   destBottom                     Coordinate16,
   width                          Integer16,
   height                         Integer16,
   bitsPerPixel                   INTEGER (1|4|8),
   compressedFlag                 BOOLEAN,
   bitmapData                     BitmapData,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

UpdateOrdersPDU ::= SEQUENCE
{
   shareDataHeader                ShareDataHeader,
   orderList                      SEQUENCE OF UpdateOrder,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

UpdatePalettePDU ::= SEQUENCE
{
   shareDataHeader                ShareDataHeader,
   palette                        ColorPalette,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

UpdateSynchronizePDU ::= SEQUENCE
{
   shareDataHeader                ShareDataHeader,
   nonStandardParameters          SEQUENCE OF NonStandardParameter OPTIONAL,
                                  -- Subject to capability negotiation.
   ...
}

WindowActivationPDU ::= SEQUENCE
{
   shareDataHeader                ShareDataHeader,
   action CHOICE
   {
      localWindowActive           [1] LocalWindowActiveIndication,
      hostedWindowActive          [2] HostedWindowActiveIndication,
      hostedWindowInvisible       [3] HostedWindowInvisibleIndication,
      pointerDeviceCapture        [4] PointerDeviceCaptureIndication,
      activateWindow              [32769] ActivateWindowRequest,
      closeWindow                 [32770] CloseWindowRequest,
      restoreWindow               [32771] RestoreWindowRequest,
      windowManagerMenu           [32772] WindowManagerMenuRequest,
      activationHelpKey           [32785] ActivationHelpKeyRequest,
      activationHelpIndexKey      [32786] ActivationHelpIndexKeyRequest,
      activationHelpExtendedKey   [32787] ActivationHelpExtendedKeyRequest,
```

```
      nonStandardAction             NonStandardParameter,
                                    -- Subject to capability negotiation.
      ...
   },
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

WindowListPDU ::= SEQUENCE
{
   shareDataHeader                  ShareDataHeader,
   listTime                         Integer16,
   listID                           Integer16,
   windowAttributeList              SEQUENCE OF WindowAttribute,
   windowTitleList                  SEQUENCE OF WindowTitle,
   nonStandardParameters            SEQUENCE OF NonStandardParameter OPTIONAL,
                                    -- Subject to capability negotiation.
   ...
}

ASNonStandardPDU ::= SEQUENCE
{
   nonStandardParameter             NonStandardParameter,
                                    -- Subject to capability negotiation.
   ...
}

SharePDU    ::= CHOICE
{
   applicationPDU                   [25] ApplicationPDU,
   controlPDU                       [20] ControlPDU,
   flowResponsePDU                  [66] FlowResponsePDU,
   flowStopPDU                      [67] FlowStopPDU,
   flowTestPDU                      [65] FlowTestPDU,
   fontPDU                          [11] FontPDU,
   inputPDU                         [28] InputPDU,
   mediatedControlPDU               [29] MediatedControlPDU,
   pointerPDU                       [27] PointerPDU,
   remoteSharePDU                   [30] RemoteSharePDU,
   synchronizePDU                   [31] SynchronizePDU,
   updateBitmapPDU                  [1] UpdateBitmapPDU,
   updateOrdersPDU                  [0] UpdateOrdersPDU,
   updateSynchronizePDU             [3] UpdateSynchronizePDU,
   updatePalettePDU                 [2] UpdatePalettePDU,
   windowActivationPDU              [23] WindowActivationPDU,
   windowListPDU                    [24] WindowListPDU,
   asNonStandardPDU                 ASNonStandardPDU,
   ...
}


--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--
--                              End AS Definitions
--
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
--||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

END
```

9.3 Règles de codage du mode hérité

Le codage AS des éléments de données ASPDU définis au 9.1 se fait comme suit.

Les bits des combinaisons de bits d'un octet sont identifiés par b_7, b_6, b_5, b_4, b_3, b_2, b_1 et b_0, où b_7 est le bit de rang le plus élevé, ou le plus significatif, et b_0 est le bit de rang le moins élevé ou moins significatif.

- Un octet à l'intérieur d'un élément de donnée est codé comme une séquence de bits, où le bit b_7 de l'octet est codé dans le bit de rang le plus élevé ou le plus significatif de l'octet codé correspondant, b_6 est codé dans le rang le plus élevé qui suit et ainsi de suite en remplissant jusqu'au bit le moins significatif.
- OCTET STRING est codé comme une séquence d'octets selon leur ordre d'apparition dans l'élément de donnée.
- INTEGER (0..15) est codé dans les quatre bits disponibles de rangs les plus élevés, ou plus significatifs, dans un octet.
- INTEGER (0..255) et INTEGER (–128..127) sont codés dans un octet contenant la valeur binaire en complément à deux de l'élément de donnée.
- INTEGER (0..4095) est codé dans les douze bits disponibles de rangs les plus élevés, ou plus significatifs, dans un octet.
- INTEGER (0..65535) et INTEGER (–32768..32767) sont codés dans deux octets contenant la valeur binaire en complément à deux de l'élément de donnée, où l'octet le plus significatif est placé dans le second octet.
- INTEGER (0..4294967295) est codé dans les quatre octets contenant la valeur binaire en complément à deux de l'élément de donnée. Les octets sont rangés par ordre d'importance croissante avec le plus significatif placé dans le quatrième octet.
- BIT STRING (0..7) est codé dans un octet unique.
- BIT STRING (0..15) est codé comme INTEGER (0..65535).
- BIT STRING (0..31) est codé comme INTEGER (0..4294967295).
- Tous les octets sont mis en paquets aux frontières de bits. Des éléments de remplissage sont définis de manière explicite.
- Là où un élément de donnée appartient à un type CHOICE, l'élément de donnée spécifique dépend des autres données comme décrit dans la Spécification de Protocole (voir le pargraphe 8). Aucun bit supplémentaire ne sera codé pour de tels éléments de données.
- Là où un élément de donnée est OPTIONAL, le fait que l'élément de donnée soit codé ou non dépend des autres éléments comme décrit dans la Spécification de Protocole (voir le paragraphe 8). Aucun bit supplémentaire ne sera codé pour de tels éléments de données.
- Là où l'usage des éléments de données n'est pas spécifié dans la Spécification de Protocole (voir le paragraphe 8), l'élément de donnée est codé comme ci-dessus, mais les valeurs de bits ne sont pas définies. Aucun bit supplémentaire ne sera codé pour de tels éléments de données.

9.4 Règles de codage des capacités non réductibles dans le mode de base

Dans le mode de base du protocole AS, une ASCE peut annoncer une capacité particulière via les listes de capacités réductibles ou non dans le répertoire (voir 8.2.2). Là où une ASCE annoncera une capacité particulière via les capacités non réductibles dans le répertoire, alors elle codera la valeur de capacité en utilisant les règles de codage définies dans le présent sous-paragraphe.

Dans le mode de base du protocole AS, les capacités peuvent être une des classes définies dans le Tableau 8-1.

Les classes de capacités définies sont codées en utilisant la variante BASIC ALIGNED des Règles de Codage compact de la Recommandation X.691 basées sur les définitions ASN.1 suivantes.

```
LogicalNonCollapsingCapability ::= BOOLEAN              -- Class L: logical value.
IntegerNonCollapsingCapability ::= INTEGER (MIN..MAX)   -- Class N: signed or unsigned integer value.
```

A la suite du codage, la valeur codée d'une capacité particulière est utilisée comme la valeur de capacité non réductible pour un codage ultérieur par le GCC.

ANNEXE A

Affectation des identificateurs de canaux statiques

Le Tableau A.1 liste l'affectation numérique des identificateurs de canaux statiques pour les canaux statiques alloués dans le cadre de l'utilisation de la présente Recommandation. Il est prévu de centraliser l'affectation numérique des identificateurs de canaux statiques dans la Recommandation T.120. Cependant, celle-ci est incluse ici en attendant la finalisation de la Recommandation T.120.

Tableau A.1/T.128 – Affectation d'identificateur de canal statique

Nom symbolique	Channel ID
AS-CHANNEL-0	11

ANNEXE B

Clé de protocole d'application en mode hérité

Le Tableau B.1 définit les contenus de la clé de protocole d'application utilisée pour identifier le mode hérité du protocole d'application défini par la présente Recommandation.

Tableau B.1/T.128 – Clé de protocole d'application en mode hérité

Clé de protocole d'Application	Description
h221NonStandard: 0xB5, 0x00, 0x53, 0x4C, 0x02	cette séquence définit le contenu de la clé du protocole d'application permettant d'identifier le mode "legs" dans le protocole d'application défini par la présente Recommandation. Noter que pour définir cette clé, l'on fait appel à l'option d'identificateur non normalisé selon la Recommandation H.221. Les valeurs numériques indiquées représentent le contenu de l'identificateur non normalisé H.221 tel qu'il est défini dans la Recommandation T.124.

ANNEXE C

Affectation des identificateurs d'objets

Le Tableau C.1 liste l'affectation des identificateurs d'objets définis pour usage par la présente Recommandation.

Tableau C.1/T.128

Valeur d'identificateur d'objet	Description
{itu-t recommendation t 128 version (0) 1}	cet identificateur d'objet est utilisé pour indiquer la version de la présente Recommandation.

APPENDICE I

Valeurs informatives

Le présent appendice propose des valeurs pour les valeurs diverses décrites dans le corps principal de la présente Recommandation, sur la base de l'expérience du partage d'application sur un nombre de types de terminaux. Ces valeurs ne sont pas obligatoires et les valeurs réelles utilisées par une ASCE spécifique sont laissées à la discrétion de l'implémenteur.

I.1 Contrôle de flux

Les valeurs suivantes et les expressions sont proposées pour usage dans l'algorithme de contrôle de flux décrit au 8.5. Voir les Tableaux I.1, I.2 et I.3.

Tableau I.1/T.128 – Constantes de contrôle de flux

Elément	Haute priorité de MCS	Moyenne priorité de MCS	Basse priorité de MCS
target_round_trip	2 000	pas de contrôle de flux	7 000
target_in_flight	800	pas de contrôle de flux	99 000
max_queued_recv	5	pas de contrôle de flux	5

Tableau I.2/T.128 – Variables de contrôle de flux

Elément	Valeur initiale	Minimum	Maximum
flow_period (millisecondes)	1 000	100	1 000
max_in_flight	8 000	500	256 000

Tableau I.3/T.128 – Opérations de contrôle de flux

Opération	Expression
Decrease max_in_flight	max_in_flight = max_in_flight/2 (mais pas en dessous de la valeur minimale dans le Tableau I.2)
Increase max_in_flight	max_in_flight = max_in_flight*2 (mais pas en dessus de la valeur maximale dans le Tableau I.2)
Decrease flow_period	flow_period = flow_period/2 (mais pas en dessous de la valeur minimale dans le Tableau I.2)
Increase flow_period	flow_period = flow_period*2 (mais pas en dessus de la valeur maximale dans le Tableau I.2)

I.2 Mise en antémémoire de phototrame

Les valeurs suivantes sont proposées pour usage dans le jeu de capacités d'antémémoire de phototrame décrit au 8.2.7.

Elément	Valeur proposée
cache1Entries	600
cache1MaximumCellSize	496
cache2Entries	300
cache2MaximumCellSize	2 032
cache3Entries	150
cache3MaximumCellSize	4 080

I.3 Mise en antémémoire de table de couleur

Les valeurs suivantes sont proposées pour usage dans le jeu de capacités d'antémémoire de table de couleurs décrit au 8.2.7.

Elément	Valeur suggérée
colorTableCacheSize	6

I.4 Mise en antémémoire de curseur

Les valeurs suivantes sont proposées pour usage dans le jeu de capacités d'antémémoire de curseur décrit au 8.2.11.

Elément	Valeur suggérée
colorPointerFlag	TRUE
pointerCacheSize	25

I.5 Antémémoire de sauvegarde de moniteur

Les valeurs suggérées sont proposées pour usage dans les valeurs d'antémémoire de sauvegarde de moniteur dans le jeu de capacités *Order* décrit au 8.2.5 et pour l'algorithme d'antémémoire de moniteur décrit au 8.16.17.

Elément	Valeur suggérée
desktopSaveSize	160,000
desktopSaveXGranularity	1
desktopSaveYGranularity	20

I.6 Compression générale

Les valeurs suivantes sont proposées pour usage dans le mode hérité pour la négociation et l'usage de la compression générale, comme décrit au 8.3.2.

Schéma de compression	generalCompressionTypes (voir le Tableau 8-3)	generalCompressionLevel (voir le Tableau 8-3)	generalCompressedType (voir le Tableau 8-3)
PKWARE PKZIP	indicateur binaire 0 positionné	0 et 1	1

SERIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	**Terminaux des services télématiques**
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux pour données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information
Série Z	Langages de programmation