Groupe de travail Réseau
Request for Comments : 4037
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

A. Rousskov, The Measurement Factory

mars 2005

Cœur du protocole d'invocation (OCP) des services marginaux à connexion libre (OPES)

Statut de ce mémoire

Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

Le présent document spécifie le cœur du protocole d'invocation (OCP, *OPES Callout Protocol*) des services marginaux à connexion libre (OPES, *Open Pluggable Edge Services*). OCP régule les messages d'application provenant d'autres protocoles de communication : un intermédiaire OPES envoie des messages d'application originaux à un serveur d'invocation, celui-ci renvoie les messages d'application adaptés au processeur. OCP est conçu en visant des tâches d'adaptation typiques (par exemple, gestion de virus et de pourriels, traduction de langage et de format, anonymisation de messages, ou manipulation d'annonces). Comme défini dans le présent document, le cœur OCP consiste en mécanismes ignorants des applications qui sont essentiels pour la prise en charge efficace des adaptations typiques.

Table des matières

1. Introduction

L'architecture des services marginaux à connexion libre (OPES, *Open Pluggable Edge Services*) [RFC3835] permet des services d'application coopératifs (services OPES) entre un fournisseur de données, un consommateur de données, et zéro, un ou plusieurs processeurs OPES. Les services d'application considérés analysent et éventuellement transforment les messages de niveau application échangés entre le fournisseur de données et le consommateur des données.

Le processeur OPES peut déléguer la responsabilité de l'exécution du service en communiquant avec des serveurs d'invocation. Comme décrit dans la [RFC3836], un processeur OPES invoque et communique avec les services sur un serveur d'invocation en utilisant un protocole d'invocation OPES, (OCP, *OPES callout protocol*). Le présent document spécifie le cœur de ce protocole ("cœur OCP").

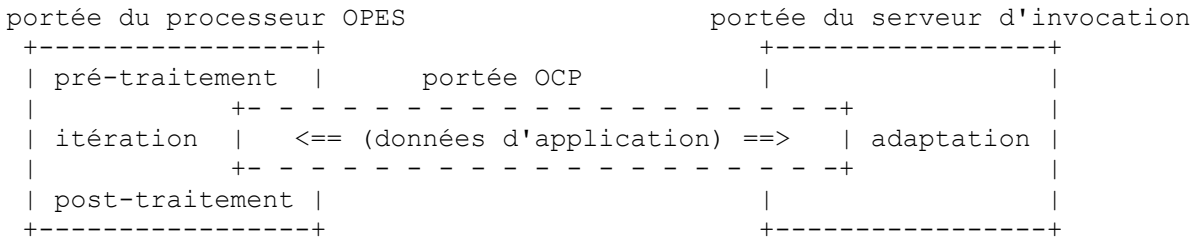
La spécification du cœur d'OCP documente des mécanismes de protocole généraux indépendants de l'application. Une série de documents séparée décrit les aspects spécifiques des applications d'OCP. Par exemple, "Adaptation HTTP avec OPES" [RFC4236] décrit, en partie, comment les messages HTTP et les méta informations HTTP peuvent être communiquées sur OCP.

Le paragraphe 1.2 fait un bref survol de la collection entière des documents OPES, incluant des documents qui décrivent les cas d'utilisation de OPES et les menaces pour la sécurité.

1.1 Domaine d'application

La spécification du cœur d'OCP documente le comportement des agents OCP et les exigences pour les extensions à OCP. Le cœur OCP ne contient pas d'exigences ou de mécanismes spécifiques pour les protocoles d'application qui sont adaptés.

Comme mandataire d'application, le processeur OPES s'attache à un seul protocole d'application ou convertit d'un protocole d'application à un autre. En même temps, le processeur OPES peut être un client OCP, utilisant OCP pour faciliter l'adaptation des messages mandatés aux serveurs d'invocation. Il est donc naturel de supposer qu'un processeur OPES prend les messages d'application qui lui sont mandatés, les surveille sur OCP pour les serveurs d'invocation, et remet ensuite les résultats de l'adaptation sur le réseau. Cependant, cette hypothèse implique que OCP soit appliqué directement aux messages d'application pour lesquels le processeur OPES est mandaté, ce qui peut n'être pas le cas.



Un processeur OPES peut pré traiter (ou post traiter) les messages d'application mandatés avant (ou après) qu'ils ont été adaptés aux serveurs d'invocation. Par exemple, un processeur peut prendre une réponse HTTP qui a été mandatée et la passer telle qu'elle, avec des métadonnées sur la connexion HTTP correspondante. Un autre processeur peut prendre une réponse HTTP, extraire son corps, et passer ce corps avec les métadonnées de codage de contenu. De plus, pour effectuer l'adaptation, le processeur OPES peut exécuter plusieurs services d'invocation, en itérant sur plusieurs serveurs d'invocation. Un tel pré traitement, post traitement, et itérations rendent impossible de s'appuyer sur des relations spécifiques entre les messages d'application qui sont mandatés et les messages d'application qui sont envoyés à un service d'invocation. De façon similaire, les actions spécifiques d'adaptation au serveur d'invocation sortent du domaine d'application du cœur OCP.

Le présente spécification ne définit ni n'exige aucune relation spécifique entre les messages d'application mandatés par un processeur OPES et les messages d'application qui sont échangés entre un processeur OPES et un serveur d'invocation via OCP. Le processeur OPES fournit généralement une transposition entre ces messages d'application, mais les actions spécifiques du processeur sortent du domaine d'application de OCP. En d'autres termes, cette spécification n'est pas concernée par le rôle du processeur OPES comme mandataire d'application ou comme itérateur de services d'invocation. La portée du cœur OCP est la communication entre un seul processeur OPES et un seul serveur d'invocation.

De plus, un processeur OPES peut choisir quels messages d'application mandatés, ou informations sur eux, envoyer sur OCP. Tous les messages mandatés sur toutes les connexions mandatées (si des connexions sont définies pour un certain protocole d'application), tout sur certaines connexions, des messages mandatés choisis, ou rien du tout, peuvent être envoyés sur OCP aux serveurs d'invocation. L'état du processeur OPES et du serveur d'invocation par rapport aux protocoles mandatés peut être relayé sur OCP comme des métadonnées de message d'application.

1.2 Cartographie des documents OPES

Le présent document appartient à un large ensemble de spécifications OPES produites par le groupe de travail OPES de l'IETF. La familiarité avec l'approche globale de OPES et ses scénarios typiques est souvent essentielle quand on essaye de comprendre des documents OPES isolés. Ce paragraphe fournit un index des documents OPES pour aider le lecteur à trouver les informations "manquantes".

- "Cas d'utilisation et scénarios de déploiement de OPES" [RFC3752] décrit un ensemble de services et applications qui sont considérés dans le domaine de OPES et ont été utilisés comme motivation et ligne directrice pour la conception de l'architecture de OPES.

- L'architecture et la terminologie commune de OPES sont décrites dans "Une architecture pour les services marginaux à connexion libre (OPES)" [RFC3835].

- "Exigences de politique, d'autorisation, et de mise en application de OPES" [RFC3838] précise les exigences et hypothèses du cadre de politique, sans spécifier de méthodes concrètes d'autorisation et de mise en application.

- "Menaces pour la sécurité et risques pour les OPES" [RFC3837] fait une analyse des risques pour les OPES, sans recommander de solutions spécifiques.

- "Traitement par l'OPES des considérations de l'IAB" [RFC3914] s'adresse à toutes les considérations de niveau architecture exprimées par le bureau de l'architecture de l'Internet (IAB) quand a été rédigé le mandat du groupe de travail OPES.

- Au cœur de l'architecture des OPES sont le processeur OPES et le serveur d'invocation, deux éléments de réseau qui communiquent l'un avec l'autre via un protocole d'invocation d'OPES (OCP). Les exigences pour ce protocole sont discutées dans "Exigences pour les protocoles d'invocation des OPES" [RFC3836].

- Le présent document spécifie un cœur de protocole ignorant de l'application destiné à être utilisé pour la communication entre un processeur OPES et un serveur d'invocation.

- "Communications des entités OPES et des points d'extrémité" [RFC3897] spécifie les mécanismes génériques de traçage et d'outrepassement pour les OPES.

- Le cœur OCP et les documents de communications sont indépendants du protocole d'application adapté par les entités OPES. Leurs mécanismes génériques doivent être complétés par des profils spécifiques des applications. "Adaptation HTTP avec les OPES" [RFC4236] est un tel profil d'application pour HTTP. Il spécifie comment des mécanismes OPES ignorants des applications vont être utilisés et augmentés afin de prendre en charge l'adaptation de messages HTTP.

- Finalement, "P : langage de traitement de message" [OPES-RULES] définit un langage pour spécifier quelles adaptations OPES (par exemple, traduction) doivent être appliquées à quels messages d'application (par exemple, le message électronique provenant de bob@exemple.com). Le langage P est destiné à configurer les mandataires d'application (processeurs OPES).

1.3 Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119]. Quand ils sont utilisés avec une signification normative, ces mots-clés seront en majuscules. Les occurrences de ces mots en minuscules constituent l'usage normale de la prose, sans implication normative.

Le processeur OPES fonctionne avec des messages provenant des protocoles d'application et peut relayer des informations sur ces messages d'application à un serveur d'invocation. OCP est aussi un protocole d'application. Donc, des éléments de protocole tels que "message", "connexion", ou "transaction" existent dans OCP et d'autres protocoles d'application. Dans la présente spécification, toutes les références aux éléments provenant de protocoles d'application autres que OCP sont utilisés avec un qualificatif explicite "d'application". Les références sans le qualificatif "d'application" se réfèrent aux éléments OCP.

Message OCP : unité de base de communication entre un processeur OPES et un serveur d'invocation. Le message est une séquence d'octets formée conformément aux règles de syntaxe (paragraphe 3.1). La sémantique des messages est définie à la Section 11.

Message d'application : entité définie par la négociation entre processeur OPES et serveur d'invocation. Généralement, la définition négociée va correspondre à la définition provenant d'un protocole d'application (par exemple, [RFC2616] définition d'un message HTTP).

Données de message d'application : séquence d'octets opaque qui représente un message d'application complet ou partiel. Le cœur OCP ne distingue pas les structures de message d'application (si il y en a). Les données de message d'application peuvent être vides.

Données : même chose que données de message d'application.

Original : se réfère à un message d'application qui s'écoule du processeur OPES à un serveur d'invocation.

Adapté : se réfère à un message d'application qui s'écoule d'un serveur d'invocation OPES au processeur OPES.

Adaptation : toute sorte d'accès par un serveur d'invocation, incluant la modification, la génération, et la copie. Par exemple, traduire ou enregistrer un message SMTP est l'adaptation de ce message d'application.

Agent : acteur pour un certain protocole de communication. Le processeur OPES et le serveur d'invocation sont des agents OCP. On peut se référer à un agent comme envoyeur ou receveur, selon ses actions dans un certain contexte.

Immédiat : effectuer l'action spécifiée avant de réagir aux nouveaux messages entrants ou d'envoyer de nouveaux messages sans relation avec l'action spécifiée.

Extension OCP : spécification qui étend ou ajuste le présent document pour l'adaptation d'un protocole d'application (autrement dit, un profil d'application ; par exemple, [RFC4236]), une nouvelle fonction OCP (par exemple, chiffrement et authentification du transport) et/ou une nouvelle version de cœur OCP.

2. Fonctionnement global

Le processeur OPES peut utiliser le protocole d'invocation OPES (OCP) pour communiquer avec des serveurs d'invocation. L'adaptation en utilisant des services d'invocation est parfois appelée une architecture "prise sur le réseau".

2.1 Initialisation

Le processeur OPES établit des connexions de transport avec les serveurs d'invocation pour échanger des messages d'application avec le ou les serveurs d'invocation en utilisant OCP. Après l'établissement d'une connexion de couche transport (généralement TCP/IP) les agents OCP communicants échangent des messages "Début de connexion" (CS, *Connection Start*). Ensuite, les caractéristiques OCP peuvent être négociées entre le processeur et le serveur d'invocation (Section 6). Par exemple, les agents OCP peuvent négocier le chiffrement du transport et la définition du message d'application.

Lorsque suffisamment de réglages sont négociés, les agents OCP peuvent commencer à échanger les messages d'application.

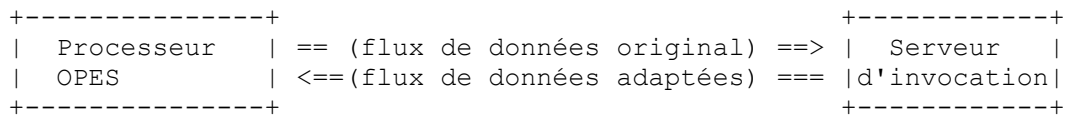
Le cœur OCP fournit la négociation et d'autres mécanismes pour que les agents chiffrent les connexions OCP et s'authentifient les uns les autres. Le cœur OCP n’exige pas le chiffrement de la connexion OCP ni l'authentification de l'agent. Les profils d'application et autres extensions à OCP peuvent documenter et/ou exiger ces mécanismes de sécurité et d'autres. OCP est supposé être utilisé, en partie, dans des environnements clos où la confiance et la confidentialité sont établies par des moyens externes à OCP. Les mises en œuvre sont supposées demander les caractéristiques de sécurité nécessaires via le mécanisme de négociation de cœur OCP, selon la configuration et l'environnement de l'agent.

2.2 Flux de données original

Lorsque le processeur OPES veut adapter un message d'application, il envoie un message "Début de transaction" (TS, *Transaction Start*) pour initier une transaction OCP dédiée à ce message d'application. Le processeur envoie alors un message "Début de message d'application" (AMS, *Application Message Start*) pour préparer le serveur d'invocation pour les données d'application qui vont suivre. Une fois établie la portée du message d'application, les données d'application peuvent être envoyées au serveur d'invocation en utilisant le message "Utiliser mes données" (DUM, *Data Use Mine*) et les messages OCP qui s'y rapportent. Tous ces messages correspondent au flux de données original.

2.3 Flux de données adaptées

Le serveur d'invocation reçoit les données et métadonnées envoyées par le processeur OPES (flux de données original). Le serveur d'invocation analyse les métadonnées et adapte les données à mesure qu'elles arrivent. Le serveur construit généralement sa version des métadonnées et répond au processeur OPES avec un message "Début de message d'application" (AMS, *Application Message Start*). Les données adaptées de message d'application peuvent être envoyées ensuite, en utilisant le ou les messages OCP "Utiliser mes données" (DUM). Le message d'application est alors annoncé comme étant "achevé" ou "fermé" en utilisant un message "Fin de message d'application" (AME, *Application Message End*). La transaction peut aussi être close en utilisant un message "Fin de transaction" (TE, *Transaction End*). Tous ces messages correspondent au flux de données adaptées.



Le processeur OPES reçoit le message d'application adapté envoyé par le serveur d'invocation. Les autres actions de processeur OPES spécifiques du message d'application reçu sortent du domaine d'application de la présente spécification.

2.4 Messages d'application multiples

Le cœur OCP spécifie des interfaces de transaction dédiées à l'échange d'un seul message d'application original et d'un seul message d'application adapté. Certains protocoles d'application peuvent exiger plusieurs versions adaptées pour un seul message d'application original ou même plusieurs messages originaux échangés au titre d'une seule transaction OCP. Par exemple, un seul message électronique original peut devoir être transformé en plusieurs messages électroniques, avec un message personnalisé pour chaque receveur.

Des extensions à OCP PEUVENT documenter des mécanismes pour échanger plusieurs messages d'application originaux et/ou adaptés au sein d'une seule transaction OCP.

2.5 Terminaison

L'un ou l'autre agent OCP peut terminer la livraison, transaction, ou connexion de message d'application par l'envoi d'un message OCP approprié. Généralement, le serveur d'invocation termine la livraison du message d'application adapté et la transaction. Des terminaisons prématurées et anormales à des instants arbitraires sont acceptées. Le message de terminaison inclut une description du résultat.

2.6 Schémas d'échange de messages

En plus des messages qui portent des données d'application, les agents OCP peuvent aussi échanger des messages relatifs à leur configuration, état, connexions de transport, connexions d'application, etc. Un serveur d'invocation peut se supprimer de la boucle de traitement de message d'application. Un seul processeur OPES peut communiquer avec de nombreux serveurs d'invocation et vice versa. Bien que de nombreux schémas d'échange OCP ne suivent pas un modèle classique client-serveur, il est possible de voir un processeur OPES comme un "client OCP" et un serveur d'invocation comme un "serveur OCP". Le document d'architecture OPES [RFC3835] décrit les possibilités de configuration.

Les règles informelles suivantes illustrent les relations entre connexions, transactions, messages OCP, et messages d'application :

- Un agent OCP peut communiquer avec plusieurs agents OCP. Ceci sort du domaine d'application de la présente spécification.
- Un processeur OPES peut avoir plusieurs connexions OCP concurrentes à un serveur d'invocation. La communication sur plusieurs connexions OCP sort du domaine d'application de la présente spécification.
- Une connexion peut porter plusieurs transactions concurrentes. Une transaction est toujours associée à une seule connexion (c'est-à-dire, une transaction ne peut pas s'étendre sur plusieurs connexions concurrentes).
- Une connexion peut porter au plus un message à la fois, incluant les messages de contrôle et ceux relatifs aux transactions. Un message est toujours associé à une seule connexion (c'est-à-dire, un message ne peut pas s'étendre sur plusieurs connexions concurrentes).
- Une transaction est une séquence de messages relatifs à l'application d'un certain ensemble de services d'invocation à un seul message d'application. Une séquence de messages de transaction provenant d'un processeur OPES à destination d'un serveur d'invocation est appelé un flux original. Une séquence de messages de transaction d'un serveur d'invocation à un processeur OPES est appelée un flux adapté. Les deux flux peuvent se chevaucher dans le temps.

- o Dans le cœur OCP, une transaction est associée à un seul message d'application et à un seul message d'application adapté. Des extensions du cœur OCP peuvent étendre la portée d'une transaction à plus de messages d'application.
- o Un message d'application (adapté ou original) est transféré en utilisant une séquence de messages OCP.

2.7 Fins de temporisation

Des violations de OCP, des limites de ressources, des dépendances externes, et d'autres facteurs peuvent conduire à des états dans lesquels un agent OCP ne reçoit pas les messages requis de l'autre agent OCP. Le cœur OCP ne définit pas de message pour de telles situations. En l'absence de tout mécanisme d'extension, les agents OCP doivent mettre en œuvre des temporisations pour les opérations OCP. Un agent OCP DOIT obligatoirement terminer toute connexion, négociation, transaction OCP, etc., qui ne progresse plus. Cette règle couvre les situations de mort et de blocage.

Dans leur mise en œuvre, les agents OCP PEUVENT s'appuyer sur des temporisations de niveau transport ou d'autres temporisations externes si il est garanti que de telles temporisations externes vont se produire pour une certaine opération OCP. Selon l'opération OCP, un agent peut tirer parti de l'envoi d'un "ping" à l'autre côté avec un message "Interrogation de progrès (PQ, *Progress Query*) avant de terminer une transaction ou connexion OCP. Le premier est particulièrement utile pour les adaptations qui peuvent prendre longtemps au serveur d'invocation avant de produire des données adaptées.

2.8 Environnement

La communication OCP est généralement supposée avoir lieu sur des connexions TCP/IP sur l'Internet (bien qu'aucun accès TCP par défaut ne soit alloué à OCP dans la présente spécification). Cela n'empêche pas OCP d'être mis en œuvre par dessus d'autres protocoles de transport, ou sur d'autres réseaux. Des protocoles de transport de haut niveau comme BEEP [RFC3080] peuvent être utilisés. Le cœur OCP exige un transport fiable et qui préserve l'ordre des messages. Tout protocole qui a ces propriétés peut être utilisé ; la transposition des structures de message OCP sur les unités de données de transport du protocole en question sort du domaine d'application de la présente spécification.

Le cœur OCP est neutre par rapport à l'application. Les messages OCP peuvent porter des informations spécifiques d'application comme charge utile ou comme paramètres de message spécifiques de l'application.

Les frais généraux du cœur OCP en termes de trafic supplémentaire sur le réseau sont d'environ 100 à 200 octets par petit message d'application. Le traitement en parallèle *(pipelining)*, la présentation préalable, la préservation des données, et les optimisations de terminaison précoce, ainsi que l'encapsulation telle quelle des données d'application, rendent possible l'échange rapide des messages d'application.

3. Messages

Comme défini au paragraphe 1.3, un message OCP est une unité de base de communication entre un processeur OPES et un serveur d'invocation. Un message est une séquence d'octets formatée en accord avec les règles de la syntaxe (paragraphe 3.1). La sémantique des message est définie à la Section 11. Les messages sont transmis par dessus le transport OCP.

Les messages OCP traitent des échanges de données de transport, de gestion de transaction, et d'application entre un seul processeur OPES et un seul serveur d'invocation. Certains messages ne peuvent être émis que par un processeur OPES, d'autres seulement par un serveur d'invocation, et certains autres que par le processeur OPES et le serveur d'invocation. Certains messages exigent une réponse (on pourrait appeler ces messages des "requêtes") ; certains ne peuvent être utilisés qu'en réponse à d'autres messages ("réponses") ; certains peuvent être envoyés sans sollicitation, et certains peuvent ne pas exiger de réponse.

3.1 Format de message

Un message OCP consiste en un nom de message suivi par des paramètres facultatifs et une charge utile. La syntaxe exacte du message est définie par le format Backus-Naur augmenté (ABNF) [RFC2234] suivant :

```
message = nom [SP paramètres anonymes]
        [CRLF paramètres nommés CRLF]
        [CRLF charge utile CRLF]
        ";" CRLF
```

```
paramètres anonymes = valeur *(SP valeur)                    ; séparées par une espace
paramètres nommés  = valeur nommée *(CRLF valeur nommée)     ; séparées par un CRLF
liste d'éléments = valeur *("," valeur)                      ; séparées par une virgule

charge utile = données

valeur nommée = nom ":" SP valeur

valeur = structure / liste / atome
structure = "{" [paramètres anonymes] [CRLF paramètres nommés CRLF] "}"
liste = "(" [ liste d'éléments ] ")"
atome = valeur nue / valeur citée

nom = ALPHA *OCTET sûr
valeur nue = 1*OCTET sûr
valeur citée = DQUOTE données DQUOTE
données = taille ":" *OCTET                                  ; exactement les octets de taille

OCTET sûr = ALPHA / CHIFFRE / "-" / "_"
taille = nombre décimal                                      ; 0 à 2 147 483 647
nombre décimal = 1* CHIFFRE                                  ; pas de zéros ou signes en tête
```

Plusieurs règles normatives accompagnent l'ABNF ci-dessus:

- Il n'y a pas de règle "espace linéaire implicite" (LWS). Les règles LWS sont courantes dans les grammaires fondées sur MIME mais elles ne sont pas utilisées ici. La syntaxe d'espace est restreinte à ce qui est explicitement permis par l'ABNF ci-dessus.
- Tous les éléments de protocole sont sensibles à la casse sauf spécification contraire. En particulier, les noms de message et de paramètres sont sensibles à la casse.
- Les tailles sont interprétées comme des valeurs décimales et ne peuvent pas porter de zéros en tête.
- Les tailles ne doivent pas excéder 2 147 483 647.
- L'attribut de taille dans un codage de valeur citée spécifie le nombre exact d'octets suivant le séparateur deux-points (':'). Si les octets de taille ne sont pas suivis par un caractère guillemet ("") le codage est syntaxiquement invalide.
- Les valeurs citées vides sont codées comme une séquence de quatre octets "0:".
- Toute valeur nue peut être codée comme une valeur citée. Une valeur citée est interprétée après la suppression du codage. Par exemple, le nombre 1234 peut être codé comme quatre octets 1234 ou comme huit octets "4:1234", donnant exactement la même signification.
- Unicode UTF-8 est le codage par défaut. Noter que ASCII est un sous ensemble de UTF-8, et que la syntaxe interdit les caractères non ASCII en dehors de l'élément "données".

Les messages qui violent les règles de formatage sont, par définition, invalides. Voir à la Section 5 les règles qui gouvernent le traitement des messages invalides.

3.2 Rendu des messages

Les échantillons de message OCP dans la présente spécification et ses extensions peuvent n'être pas composés pour décrire des détails syntaxiques mineurs du format de message OCP. Précisément, les caractères SP et CRLF ne sont pas montrés explicitement. Aucun rendu d'un message OCP ne peut être utilisé pour déduire le format de message. La définition du format de message ci-dessus est la seule source normative pour toutes les mises en œuvre.

À l'occasion, une ligne de message OCP peut excéder la largeur de texte permise par ce format de spécification. Une barre oblique inverse ("\"), un caractère "coupure de ligne", est utilisé pour souligner une coupure de ligne de simple présentation qui viole le protocole. Les barres obliques inverses nues sont interdites par la syntaxe OCP. De même, une chaîne "\r\n" est parfois utilisée pour souligner la présence d'une séquence de CRLF, généralement avant la charge utile du message OCP. Normalement, la fin de ligne visible correspond à la séquence CRLF sur le réseau.

Le paragraphe suivant (3.3) contient des exemples spécifiques de messages OCP, dont certains illustrent les techniques de rendu ci-dessus.

3.3 Exemples de messages

La syntaxe de OCP assure une représentation compacte des messages de contrôle courts et des paramètres exigés tout en permettant des extensions de paramètres. Des exemples de courts messages de contrôle sont présentés ci-dessous. La séquence requise de CRLF à la fin de chaque ligne n'est pas montrée explicitement (voir le paragraphe 3.2).

```
PQ;
TS 1 2;
DWM 22;
DWP 22 16;
x-must "5:xyzzy";
```

Les exemples ci-dessus contiennent des valeurs de paramètres anonymes atomiques, comme des constantes de nombre et de chaîne. Les messages OCP utilisent parfois des paramètres plus compliqués comme des listes d'éléments ou des structures avec des valeurs nommées. Comme l'illustrent les deux messages ci-dessous, des structures et des listes peuvent être incorporées :

```
NO ({"32:http://www.iana.org/assignments/opes/ocp/tls"});
NO ({"54:http://www.iana.org/assignments/opes/ocp/http/response"
Optional-Parts: (request-header)
},{"54:http://www.iana.org/assignments/opes/ocp/http/response"
Optional-Parts: (request-header,request-body)
Transfer-Encodings: (chunked)
});
```

Des paramètres et extensions facultatifs sont possibles avec une approche de paramètres nommés, comme illustré par l'exemple qui suit. Le message DWM (paragraphe 11.17) dans l'exemple a deux paramètres anonymes (le dernier étant une extension) et deux paramètres nommés (le dernier étant une extension).

```
DWM 1 3
Size-Request: 16384
X-Need-Info: "26:twenty six octet extension";
```

Finalement, tout message peut avoir une partie de charge utile. Par exemple, le message "Utiliser mes données (DUM, *Data Use Mine*) ci-dessous porte 8865 octets de données brutes.

```
DUM 1 13
Modp: 75
\r\n
8865:... 8865 octets of raw data ...;
```

3.4 Nom des messages

La plupart des messages OCP définis dans la présente spécification ont des noms abrégés, formés en abrégeant ou compressant un titre plus long lisible par l'homme. Les noms abrégés sans un système d'enregistrement central (comme la présente spécification ou le registre de l'IANA) causeraient probablement des conflits. Des extensions informelles du protocole devraient éviter les noms abrégés. Pour souligner ce qui est déjà défini par la syntaxe de message, les mises en œuvre ne peuvent pas supposer que tous les noms de message sont très courts.

4. Transactions

Une transaction OCP est une séquence logique de messages OCP traitant un seul message d'application original. Le résultat du traitement peut être zéro, un ou plusieurs messages d'application, adaptés de l'original. Une transaction typique consiste en deux flux de messages : un flux du processeur OPES au serveur d'invocation (envoyant le message d'application original) et un flux du serveur d'invocation au processeur OPES (envoyant des messages d'application adaptés). Le nombre de messages d'application produit par le serveur d'invocation et si le serveur d'invocation modifie réellement le message d'application original peut dépendre du service d'invocation demandé et d'autres facteurs. Le processeur OPES ou le serveur d'invocation peut terminer la transaction en envoyant un message correspondant à l'autre côté.

Une transaction OCP commence avec un message "Début de transaction" (TS, *Transaction Start*) envoyé par le processeur OPES. Une transaction se termine avec le premier message "Fin de transaction (TE, *Transaction End*) envoyé ou reçu,

explicite ou implicite. Un message TE peut être envoyé par l'un ou l'autre côté. Zéro, un ou plusieurs messages OCP associés à la transaction peuvent être échangés entre temps. La figure ci-dessous illustre une séquence possible de messages (le préfixe "P" signifie le processeur OPES ; le préfixe "S" signifie le serveur d'invocation). Certains détails de message sont omis.

```
P: TS 10;
P: AMS 10 1;
   ... le processeur envoie des données d'application au serveur d'invocation
S: AMS 10 2;
     ... le serveur d'invocation envoie des données d'application au processeur
     ... le processeur envoie des données d'application au serveur d'invocation
P: AME 10 1 result;
S: AME 10 2 result;
P: TE 10 result;
```

5. Entrée invalide

La présente spécification contient de nombreux critères pour des messages OCP valides et leurs parties, incluant des règles de syntaxe, des exigences sémantiques, et des relations aux états d'agents. Dans ce contexte, "Entrée invalide" signifie que des messages ou parties de message violent au moins une des règles normatives. Un message avec une partie invalide est, par définition, invalide. Si les ressources d'un agent OCP sont épuisées pendant l'analyse ou l'interprétation d'un message, l'agent DOIT traiter le message OCP correspondant comme invalide.

Sauf explicitement permis autrement, un agent OCP DOIT terminer la transaction si il reçoit un message invalide avec la portée de transaction et DOIT terminer la connexion si il reçoit un message invalide avec une portée de connexion. Un agent qui termine DOIT utiliser le code d'état de résultat de 400 et PEUT spécifier les informations de cause de terminaison dans le paramètre de raison d'état de résultat (voir le paragraphe 10.10). Si un agent OCP est dans l'incapacité de déterminer la portée d'un message invalide qu'il a reçu, il DOIT traiter le message comme ayant une portée de connexion.

OCP traite généralement des manipulations facultatives mais invasives de messages d'application pour lesquelles la correction devrait être recherchée plus que la robustesse. Par exemple, un échec d'insertion ou de suppression du contenu facultatif d'une certaine page de la Toile est généralement beaucoup plus perturbant que la corruption (rendant inutilisable) la page d'accueil lors de l'exécution de cette insertion ou suppression. La plupart des adaptations OPES sont par nature de haut niveau, ce qui rend impossible de garantir la correction des adaptations automatiques, en particulier si "des souhaits de robustesse" sont impliqués.

6. Négociation

Le mécanisme de négociation permet aux agents OCP de s'accorder sur l'ensemble de caractéristiques mutuellement acceptables, incluant un comportement facultatif et spécifique de l'application et des extensions à OCP. Par exemple, le chiffrement du transport, le format des données, et la prise en charge d'un nouveau message peuvent être négociés. La négociation implique l'intention d'un changement de comportement. Pour un mécanisme permettant à un agent d'interroger sur les capacités de sa contrepartie sans changer le comportement de la contrepartie, voir les définitions des messages "Interrogation de capacités" (AQ, *Ability Query*) et "Réponse de capacités" (AA, *Ability Answer*).

La plupart des négociations exigent au moins un délai d'aller-retour. Dans de rares cas, lorsque la réponse de l'autre côté n'est pas exigée immédiatement, le délai de négociation peut être éliminé, avec un risque inhérent d'une hypothèse trop optimiste sur la réponse à la négociation.

Une violation détectée des règles de négociation conduit à la terminaison de la connexion OCP. Ce concept réduit le nombre de scénarios de négociation résultant en une impasse lorsque un des agents n'est pas conforme.

Deux primitives cœur de la négociation sont prises en charge : l'offre de négociation et la réponse de négociation. Un message "Offre de négociation" (NO, *Negotiation Offer*) permet à un agent de spécifier un ensemble de caractéristiques à partir duquel le répondeur va choisir la caractéristique qu'il préfère. Le choix est envoyé en utilisant un message "Réponse de négociation" (NR, *Negotiation Response*). Si la réponse est positive, les deux côtés vont supposer que la caractéristique choisie prend effet immédiatement (voir les détails au paragraphe 11.19). Si la réponse est négative, aucun changement de comportement n'est supposé. Dans l'un et l'autre cas, d'autres offres peuvent suivre.

Les agents OCP qui négocient doivent prendre en compte les caractéristiques préalablement négociées (c'est-à-dire, déjà activées). Les agents OCP NE DOIVENT PAS faire et DOIVENT rejeter des offres qui conduiraient à un conflit avec des caractéristique déjà négociées. Par exemple, un agent ne peut pas offrir un profil d'application HTTP pour une connexion qui a déjà activé un profil d'application SMTP, car il n'y aurait aucun moyen de résoudre le conflit pour une transaction. De même, une fois que le chiffrement TLSv1 d'une connexion est négocié, un agent ne doit pas offrir et doit rejeter des offres de chiffrement SSLv2 de la connexion (sauf si une caractéristique négociée permet explicitement de changer un schéma de chiffrement "en vol").

Les messages d'offre de négociation (NO) peuvent être envoyés par l'un ou l'autre agent. Des extensions à OCP documentant la négociation PEUVENT allouer le rôle d'initiateur à un des agents, selon la caractéristique négociée. Par exemple, la négociation d'une caractéristique de sécurité du transport devrait être initiée par les processeurs OPES pour éviter des situations où les deux agents attendent que l'autre fasse une offre.

Comme l'un ou l'autre agent peut faire une offre, deux offres "concurrentes" peuvent être faites en même temps, par les deux agents communicants. Des offres concurrentes non gérées peuvent conduite la négociation à une impasse. En donnant une priorité à un processeur OPES, les règles de traitement des offres (paragraphe 11.18) assurent que seule une offre par connexion OCP est honorée à la fois, et que les autres offres concurrentes sont ignorées par les deux agents.

6.1 Phase de négociation

Une phase de négociation est un mécanisme qui assure que les deux agents ont une chance de négocier toutes les caractéristiques dont ils ont besoin avant de poursuivre le traitement. Les phases de négociation ont une portée de connexion OCP et ne se chevauchent pas. Pour chaque agent OCP, la phase de négociation commence avec le premier message d'offre de négociation (NO) reçu ou le premier message de réponse de négociation (NR) envoyé, pourvu que le message ne fasse pas partie d'une phase existante. Pour chaque agent OCP, la phase de négociation se termine avec le premier message de réponse de négociation (NR) envoyé ou reçu, après quoi l'agent n’attend pas d'autre négociation. Les règles d'attentes des agents sont définies plus loin.

Durant une phase de négociation, un agent OCP NE DOIT PAS envoyer de messages autres que les messages de phase de négociation suivants : "Offre de négociation" (NO), "Réponse de négociation" (NR), "Interrogation de capacités" (AQ), "Réponse de capacités" (AA), "Interrogation de progrès" (PQ, *Progress Query*), "Réponse de progrès" (PA, *Progress Answer*), "Rapport de progrès" (PR, *Progress Report*), et "Fin de connexion" (CE, *Connection End*).

Plusieurs phases de négociation peuvent se produire pendant la durée de vie d'une seule connexion OCP. Un agent peut tenter de commencer une nouvelle phase de négociation immédiatement après la fin de l'ancienne phase, mais il est possible que l'autre agent envoie des messages autres que de négociation de phase avant de recevoir la nouvelle offre de négociation (NO). L'agent qui commence une phase doit être prêt à traiter ces messages pendant que son offre est en train d'atteindre le receveur.

Un processeur OPES DOIT faire une offre de négociation immédiatement après l'envoi d'un message "Début de connexion" (CS, *Connection Start*). Si le processeur OPES n'a rien à négocier, le processeur DOIT envoyer un message "Offre de négociation" (NO) avec une liste de caractéristiques vide. Ces deux règles fixent la première phase de négociation. Les agents sont supposés négocier au moins le profil d'application pour le cœur OCP. Donc, ces exigences de fixation ne vont probablement pas générer de travail supplémentaire.

Une fois qu'une phase de négociation a commencé, un agent DOIT n'attendre de négociations supplémentaires que si et seulement si le dernier NO envoyé sur le dernier NR reçu contenait une vraie valeur de paramètre "Offre en cours". De façon informelle, un agent peut garder la phase ouverte en envoyant des vrais paramètres "Offre en cours" avec les offres ou réponses de négociation. De plus, si il y a une possibilité que l'agent puisse avoir besoin de continuer la phase de négociation, l'agent doit envoyer un vrai paramètre "Offre en cours".

6.2 Exemples de négociation

Voici un exemple de la plus simple négociation possible. Le processeur OPES n'offre rien et on peut prévoir qu'il va recevoir un rejet. Noter que le message NR termine la phase de négociation dans ce cas parce que aucun des messages ne contient une vraie valeur "Offre en cours" :

P: NO ();
S: NR;

L'exemple suivant illustre comment un serveur d'invocation peut forcer la négociation d'une caractéristique qu'un processeur OPES n'a pas négocié. Noter que le serveur règle le paramètre "Offre en cours" à vrai quand il répond au message "Offre de négociation (NO) du processeur. Le processeur choisit d'accepter la caractéristique :

```
P: NO ();
S: NR
  Offer-Pending: vrai
  ;
S: NO ({"22:ocp://feature/example/"})
  Offer-Pending: faux
    ;
P: NR {"22:ocp://feature/example/"};
```

Si le serveur cherche à arrêter les négociations ci-dessus après l'envoi d'une vraie valeur "Offre en cours", sa seule option serait d'envoyer une offre de négociation vide (voir le premier exemple ci-dessus). Si le serveur ne fait rien à la place, le processeur OPES va attendre le serveur et va finalement arriver à la fin de temporisation de la connexion.

L'exemple suivant montre un dialogue avec un serveur d'invocation qui insiste pour activer deux caractéristiques imaginaires : chiffrement fort de transport et mémorisation volatile des réponses. Le serveur est conçu pour ne pas échanger de messages sensibles tant que ces deux caractéristiques ne sont pas activées. Naturellement, la caractéristique de mémorisation volatile doit être négociée de façon sécurisée. Le processeur OPES prend en charge un des mécanismes de chiffrement fort, mais préfère ne pas offrir (d'être volontaire pour prendre en charge) le chiffrement fort, peut-être pour des raisons de performances. Le serveur doit envoyer un vrai paramètre "Offre en cours" pour avoir une chance d'offrir un fort chiffrement (qui est négocié avec succès dans ce cas). Tous les messages envoyés par l'un ou l'autre agent après la (seule) réponse NR de succès sont chiffrés avec le schéma de chiffrement "strongB". Le processeur OPES ne comprend pas la caractéristique de mémorisation volatile, et la dernière négociation échoue (sur une connexion de transport fortement chiffrée).

```
P: NO ({"29:ocp://example/encryption/weak"})
  ;
S: NR
  Offer-Pending: vrai
  ;
S: NO ({"32:ocp://example/encryption/strongA"},\
  {"32:ocp://example/encryption/strongB"})
  Offer-Pending: vrai
  ;
P: NR {"32:ocp://example/encryption/strongB"}
  ;
... tout le trafic suivant est chiffré en utilisant strongB ...
S: NO ({"31:ocp://example/storage/volatile"})
  Offer-Pending: faux
  ;
P: NR
  Unknowns: ({"31:ocp://example/storage/volatile"})
  ;
S: CSE { 400 "33:pas de prise en charge du protocole VolStore" }
  ;
```

L'exemple suivant provenant de [OPES-HTTP] illustre la négociation réussie du profil d'application HTTP :

```
P: NO ({"54:http://www.iana.org/assignments/opes/ocp/http/response"
  Aux-Parts: (request-header,request-body)
  })
  SG: 5;
S: NR {"54:http://www.iana.org/assignments/opes/ocp/http/response"
  Aux-Parts: (request-header)
  Pause-At-Body: 30
  Wont-Send-Body: 2147483647
  Content-Encodings: (gzip)
  }
  SG: 5;
```

7. Optimisation de la préservation des données

De nombreuses adaptations n'exigent aucune modification des données (par exemple, enregistrement ou blocage de message). Certaines adaptations modifient seulement une petite portion du contenu du message d'application (par exemple, filtrage des mouchards HTTP ou insertion de publicités). Donc, dans de nombreux cas, le service d'invocation doit voir les données complètes. Par défaut, les données non modifiées vont d'abord voyager du processeur OPES au serveur d'invocation et retour. L'optimisation de la "préservation des données" dans OCP aide à éliminer le délai d'aller retour si les deux agents OCP coopèrent. Une telle coopération est facultative : les agents OCP PEUVENT prendre en charge l'optimisation de la préservation des données.

Pour éviter de renvoyer des données non modifiées, un service d'invocation doit savoir que le processeur OPES a une copie des données. Comme les données peuvent être de très grande taille, le service d'invocation peut ne pas savoir à l'avance si il va être capable d'utiliser la copie du processeur, il n'est pas possible de demander au processeur de garder une copie des données originales entières. À la place, il est prévu qu'un processeur puisse conserver une certaine portion des données, selon les réglages et l'état du processeur.

Quand un processeur OPES s'engage à garder un tronçon des données, il annonce sa décision et les paramètres du tronçon via un paramètre "Kept" d'un message "Utiliser mes données" (DUM, *Data Use Mine*). Le serveur d'invocation PEUT "utiliser" le tronçon en envoyant un message "Utiliser tes données" (DUY, *Data Use Yours*) qui se réfère au tronçon préservé. Ce message OCP n'a pas de charge utile et donc, le délai d'aller-retour est éliminé.

Comme la transposition entre les données originales et adaptées n'est pas connue du processeur, il DOIT conserver le tronçon tel qu'annoncé jusqu'à la fin de la transaction correspondante, sauf si le serveur d'invocation dit explicitement au processeur que le tronçon n'est pas nécessaire. Comme l'implique l'exigence ci-dessus, le processeur ne peut pas supposer qu'un tronçon de données n'est plus nécessaire juste parce que le serveur d'invocation a envoyé un message DUY ou des données adaptées avec, par exemple, le même décalage que le tronçon préservé.

Pour simplifier, les données préservées sont toujours un tronçon contigu des données originales, décrites par une paire (décalage, taille) utilisant un paramètre "Kept" d'un message DUM. Un processeur OPES peut volontairement augmenter la taille des données conservées. Un processeur OPES peut augmenter le décalage si le serveur d'invocation a indiqué que les données conservées ne sont plus nécessaires.

Les deux agents peuvent bénéficier de la réutilisation des données. Un processeur OPES doit allouer de la mémoire pour prendre en charge cette optimisation, mais pas un serveur d'invocation. Par ailleurs, c'est le serveur d'invocation qui est responsable de libérer le processeur des engagements de préservation des données. Il n'y a pas de façon simple pour résoudre ce conflit d'intérêt au niveau d'un protocole. Certains processeurs OPES peuvent allouer une mémoire tampon relativement petite pour la préservation des données et arrêter la préservation des données quand la mémoire tampon est pleine. Cette technique bénéficierait aux services d'invocation qui peuvent rapidement réutiliser ou éliminer les données conservées. Une autre stratégie de processeur serait de dimensionner la mémoire tampon sur la base d'historiques de statistiques de réutilisation des données. Pour améliorer les chances d'une coopération bénéfique, les serveurs d'invocation sont vivement encouragés à notifier immédiatement aux processeurs OPES les données qui ne sont plus voulues. Le serveur d'invocation qui prend la décision de ne pas envoyer de message DUY (pour une gamme spécifique de données ou pas du tout) DEVRAIT immédiatement informer le processeur OPES de cette décision avec le ou les messages "Data Preservation Interest" (DPI) correspondants ou d'autres mécanismes.

8. Optimisations de terminaison prématurée des flux de données

De nombreux services d'invocation adaptent de petites portions de gros messages et il serait préférable qu'ils ne soient pas dans la boucle quand l'adaptation est finie. Certains services d'invocation peuvent ne pas chercher la modification des données et il serait préférable de ne pas renvoyer les données au processeur OPES, même si le processeur OPES ne prend pas en charge l'optimisation de la préservation des données (Section 7). Selon la conception de OCP, la terminaison unilatérale prématurée du flux de données par un serveur d'invocation conduirait à la terminaison d'une transaction OCP avec une erreur. Donc, les deux agents doivent coopérer pour permettre une terminaison prématurée sans erreur.

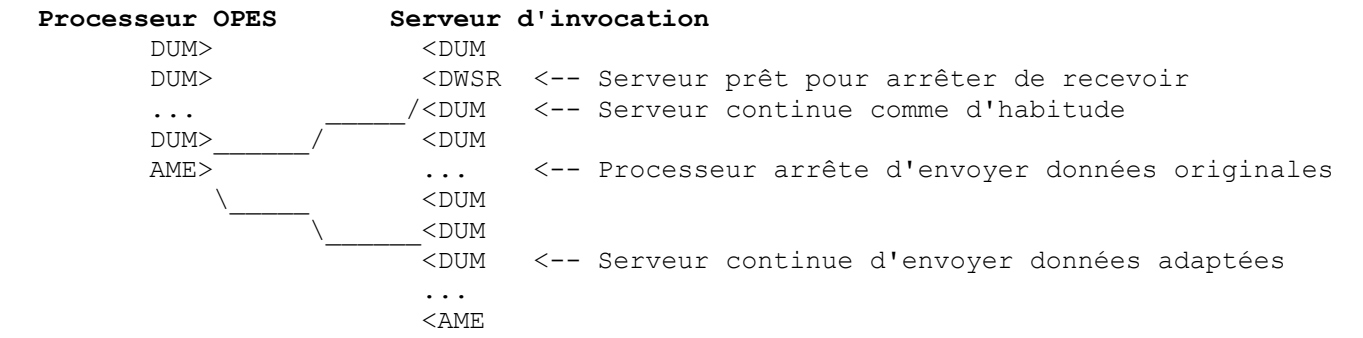
Cette Section précise deux des mécanismes pour la terminaison prématurée d'un flux de données original ou adapté. Combinés, les deux mécanismes permettent au serveur d'invocation de sortir ensemble de la boucle de traitement.

8.1 Flux de données original

Il y a des scénarios où un serveur d'invocation n'est pas intéressé par le reste du flux de données original. Par exemple, un simple blocage d'accès ou service d'invocation "ce site est temporairement hors service" doit envoyer un message d'application adapté (généré) mais de préférence ne va pas recevoir les données originales du processeur OPES.

Le cœur OCP prend en charge la terminaison prématurée du flux de données original via le message "Veux arrêter de recevoir des données" (DWSR, *Want Stop Receiving Data*). Un serveur d'invocation qui ne cherche pas à recevoir des données originales supplémentaires (au delà d'une certaine taille) envoie un message DWSR. Le processeur OPES qui reçoit un message DWSR termine le flux de données original en envoyant un message "Fin de message d'application" (AME, *Application Message End*) avec un code d'état 206 (partiel).

La figure suivante illustre une séquence d'événements typique. Les lignes vers le bas qui connectent les deux flux de données illustrent le délai de transmission qui permet que plus de messages OCP soient envoyés pendant qu'un agent attend la réaction de l'agent opposé.



Le mécanisme décrit dans ce paragraphe n'a pas d'effet sur le flux de données adaptées. Recevoir un message AME avec un code d'état de résultat 206 (partiel) du processeur OPES n'introduit aucune exigence particulière pour la terminaison du flux de données adaptées. Cependant, il n'est pas possible de terminer prématurément le flux de données adaptées après que le flux de données original a été terminé prématurément (voir le paragraphe 8.3).

8.2 Flux de données adapté

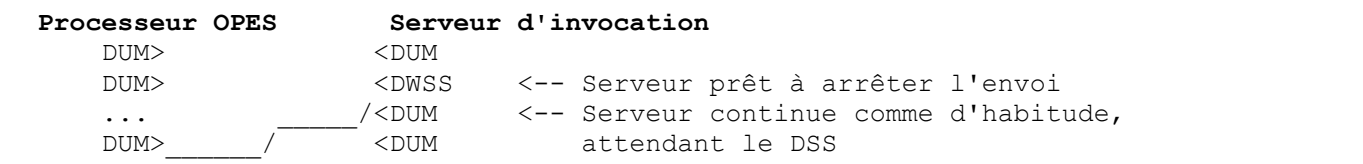
Il y a des scénarios où un service d'invocation peut vouloir arrêter d'envoyer des données adaptées avant l'envoi complet d'un message d'application. Par exemple, un service d'invocation d'enregistrement seul doit recevoir tous les messages d'application mais va préférer ne pas renvoyer de copies au processeur OPES.

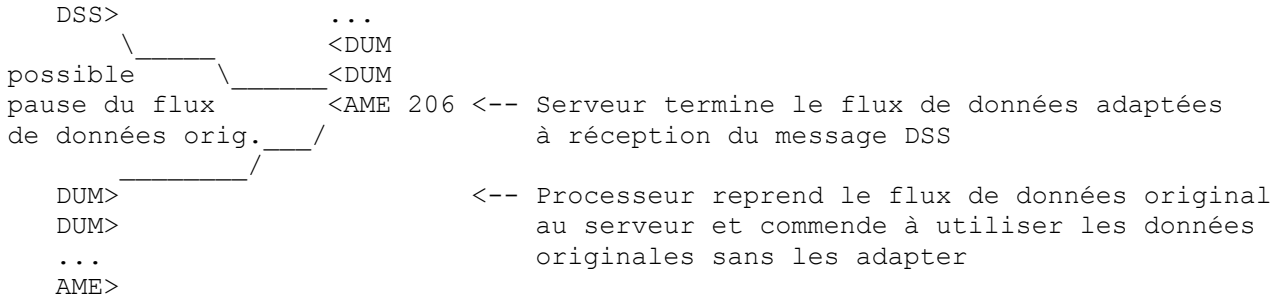
Le cœur OCP prend en charge la terminaison prématurée de flux de données adaptées via une combinaison de messages "Veux arrêter d'envoyer des données" (DWSS, *Want Stop Sending Data*) et "Arrêt d'envoi de données" (DSS, *Stop Sending Data*). Un service d'invocation qui cherche à arrêter d'envoyer des données envoie un message DWSS, sollicitant du processeur OPES la permission d'arrêter. En attendant la permission, le serveur continue sa routine habituelle.

Un processeur OPES qui reçoit un message "Veux arrêter d'envoyer des données" (DWSS) répond avec un message "Arrêt d'envoi de données" (DSS). Le processeur peut alors cesser d'attendre que le serveur d'invocation termine le flux de données adaptées ou peut continuer d'envoyer des données originales tout en en faisant une copie. Une fois que le serveur a terminé le flux de données adaptées, le processeur est responsable de l'utilisation des données originales (envoyées ou en pause après l'envoi DSS) au lieu des données adaptées.

Le serveur d'invocation qui reçoit un message DSS termine le flux de données adaptées (voir la définition du message DSS pour les exigences exactes et les cas particuliers).

La figure qui suit illustre une séquence typique d'événements, incluant une possible pause dans le flux de données original quand le processeur OPES attend la fin du flux de données adaptées. Les lignes vers le bas qui connectent les deux flux de données illustrent le délai de transmission qui permet que plus de messages OCP soient envoyés pendant qu'un agent attend la réaction de l'agent opposé.





La préservation prématurée du flux de données adaptées n'est pas triviale, car le processeur OPES s'appuie sur le serveur d'invocation pour fournir les données adaptées (modifiées ou non) pour construire le message d'application adapté. Si le serveur d'invocation cherche à quitter sa tâche, un soin particulier doit être apporté à s'assurer que le processeur OPES peut construire le message d'application complet. Au niveau logique, ce mécanisme est équivalent à passer d'un serveur d'invocation à un autre serveur d'invocation (sans modification) au milieu d'une transaction OCP.

À part une pause possible dans le flux de données original, le mécanisme décrit dans ce paragraphe n'a pas d'effet sur le flux de données original. Recevoir un message "Fin de message d'application" (AME, *Application Message End*) avec le code d'état de résultat 206 (partiel) du serveur d'invocation n'introduit aucune exigence particulière pour la terminaison du flux de données original.

8.3 Sortie de la boucle

Certains services d'adaptation travaillent sur les préfixes de message d'application et ne cherchent pas à être dans la boucle d'adaptation une fois que leur travail est fait. Par exemple, un service d'insertion d'annonces publicitaires qui a fait son travail en modifiant le premier fragment d'une "page" de la Toile ne va pas chercher à recevoir plus de données originales ou à effectuer d'autres adaptations. L'optimisation "Sortir de la boucle" permet à un serveur d'invocation de sortir complètement de la boucle de traitement de message d'application.

L'optimisation "Sortir de la boucle" est rendue possible en terminant le flux de données adaptées (paragraphe 8.2) et ensuite en terminant le flux de données original (paragraphe 8.1). L'ordre de terminaison est très important.

Si le flux de données original est terminé en premier, le processeur OPES ne va pas permettre que le flux de données adaptées soit terminé prématurément, car le processeur ne serait pas capable de reconstruire la portion restante du message d'application adapté. Le processeur ne saurait pas quel suffixe des données originales restantes doit suivre les parties adaptées. La transposition entre octets originaux et adaptés n'est connue que du service d'invocation.

Un processeur OPES qui a reçu un message DWSS suivi d'un message DWSR NE DOIT PAS envoyer de message AME avec un code d'état 206 (partiel) avant d'envoyer un message DSS. De façon informelle, cette règle signifie qu'un serveur d'invocation qui veut sortir rapidement de la boucle devrait envoyer un message DWSS immédiatement suivi d'un message DWSR ; le serveur n'a pas à attendre la permission du processeur OPES de terminer le flux de données adaptées avant de demander que le processeur OPES termine le flux de données original.

9. Mnémonique de déclaration de type d'élément de protocole

Un type d'élément de protocole est un ensemble nommé de règles de syntaxe et de sémantique. Cette section définit un mnémonique simple de déclaration formelle pour les types d'élément de protocole, étiqueté PETDM *(Protocol Element Type Declaration Mnemonic)* mnémonique de déclaration de type d'élément de protocole. La simplicité du PETDM est destinée à faciliter les déclarations de type dans la présente spécification. La formalité de PETDM est destinée à améliorer l'interopérabilité entre les mises en œuvre. Deux éléments de protocole sont pris en charge par le PETDM : les valeurs de paramètres de message et les messages.

Tous les types de paramètres et messages de cœur OCP sont déclarés en utilisant PETDM. Les extensions à OCP DEVRAIENT utiliser PETDM pour déclarer de nouveaux types.

Les constructions atome, liste, structure, et message sont quatre types de base disponibles. Leurs règles de syntaxe et de sémantique sont définies au paragraphe 3.1. De nouveaux types peuvent être déclarés dans PETDM pour étendre la sémantique du type de base en utilisant les gabarits de déclaration suivants. Les nouvelles règles de sémantique sont destinées à être attachées à chaque déclaration en utilisant un texte en prose.

Les textes entre des crochets angulaires "<>" sont des gabarits fourre tout, à remplacer par les noms de type réels ou des jetons de nom de paramètre. Les crochets rectangulaires "[]" entourent les éléments facultatifs comme des membres de structures ou une charge utile de message.

o Déclaration d'un nouveau type atomique : <nouveau-nom-de-type> : étend l'atome ;

o Déclaration d'une nouvelle liste avec des éléments nom-de-vieux-type : <nouveau-nom-de-type> : étend la liste de <nom-de-vieux-type> ; Sauf notation explicite contraire, les listes vides sont valides et ont la sémantique d'une valeur de paramètre absente.

o Déclaration d'une nouvelle structure avec des membres :
<nouveau-nom-de-type> : étend la structure avec {
<nom-de-vieux-typeA> <nom-de-vieux-typeB> [<nom-de-vieux-typeC>] ...;
<nom-de-membre1>: <nom-de-vieux-type1>;
<nom-de-membre2>: <nom-de-vieux-type2>;
[<nom-de-membre3>: <nom-de-vieux-type3>];
...
};

La nouvelle structure peut avoir des membres anonymes et des membres nommés. L'existence d'aucun des groupes n'est obligatoire. Noter qu'il est toujours possible à des extensions d'ajouter des membres à de vieilles structures sans affecter la sémantique du type parce que les membres non reconnus sont ignorés par les agents conformes.

o Déclaration d'un nouveau message avec paramètres :
<nouveau-nom-de-type> : étend le message avec {
<nom-de-vieux-typeA> <nom-de-vieux-typeB> [<nom-de-vieux-typeC>] ...;
<nom-de-paramètre1>: <nom-de-vieux-type1>;
<nom-de-paramètre2>: <nom-de-vieux-type2>;
[<nom-de-paramètre3>: <nom-de-vieux-type3>];
...
};

Le nom de nouveau type devient le nom de message. Tout comme quand une structure est étendue, le nouveau message peut avoir des paramètres anonymes et des paramètres nommés. Il peut n'en exister aucun. Noter qu'il est toujours possible aux extensions d'ajouter plus de paramètres aux vieux messages sans affecter la sémantique de type parce que les paramètres non reconnus sont ignorés des agents conformes.

o Étendre un type avec plus de détails de sémantique : <nouveau-nom-de-type>: étend <nom-de-vieux-type> ;

o Étendre un type fondé sur la structure ou le message :
<nouveau-nom-de-type> : étend <nom-de-vieux-type> avec {
<nom-de-vieux-typeA> <nom-de-vieux-typeB> [<nom-de-vieux-typeC>] ...;
<nom-de-membre1>: <nom-de-vieux-type1>;
<nom-de-membre2>: <nom-de-vieux-type2>;
[<nom-de-membre3>: <nom-de-vieux-type3>];
...
};
Les nouveaux membres anonymés sont ajoutés aux membres anonymes du vieux type, et les nouveaux membres nommés sont fusionnés avec les membres nommés du vieux type.

o Extension d'un type fondé sur le message avec une sémantique de charge utile :
<nouveau-nom-de-type> : étend <nom-de-vieux-type> avec {
...
} et charge utile ;
Tout message OCP peut avoir une charge utile, mais seuls certains types de message ont une sémantique de charge utile connue. Comme tout paramètre, la charge utile peut être requise ou facultative.

o Extension de la sémantique de type sans renommer le type :
<nom-de-vieux-type> : étend <espace-de-noms>::<nom-de-vieux-type> ;
Le gabarit ci-dessus peut être utilisé par les extensions de cœur OCP qui cherchent à changer la sémantique des types de cœur OCP sans les renommer. Cette technique est essentielle pour étendre les messages OCP parce que le nom de message

est le même que le nom de type de message. Par exemple, un profil SMTP pour OCP pourrait utiliser la déclaration suivante pour étendre un message "début de message d'application" (AMS) avec Am-Id, paramètre défini dans ce profil :

```
AMS : étend Core::AMS avec {
      Am-Id: am-id;
};
```

Tous les types étendus peuvent être utilisés en remplacement des types qu'ils étendent. Par exemple, un message "Offre de négociation" (NO) utilise un paramètre de type "Features" (caractéristiques). "Features" (paragraphe 10.12) est une liste d'éléments de caractéristiques (paragraphe 10.11). Feature est un type fondé sur la structure. Une extension OCP (par exemple, un profil d'application HTTP) peut étendre le type de caractéristique et utiliser une valeur de ce type étendu dans une offre de négociation. Les receveurs qui connaissent l'extension vont reconnaître les membres ajoutés dans les éléments de caractéristiques et vont négocier en conséquence. Les autres receveurs vont les ignorer.

L'étiquette d'espace de noms de cœur OCP est "Core". Les extensions à OCP qui déclarent des types DOIVENT définir leurs étiquettes d'espace de noms (afin que les autres extensions et la documentation puissent les utiliser dans leurs déclarations de PETDM).

9.1 Paramètres facultatifs

Les paramètres anonymes sont de position : la position du paramètre (c'est-à-dire, le nombre de paramètres anonymes à gauche) est son "nom". Donc, quand une structure ou message a plusieurs paramètres anonymes facultatifs, les paramètres sur la droite ne peuvent être utilisés que si tous les paramètres sur la gauche sont présents.

La notation suivante : [nom1] [nom2] [nom3] ... [nomN] est interprétée comme [nom1 [nom2 [nom3 ... [nomN] ...]]]

Quand un paramètre anonyme est ajouté à une structure ou message qui a des paramètres anonymes facultatifs, le nouveau paramètre doit être facultatif et ne peut être utilisé que si tous les vieux paramètres facultatifs sont utilisés. Les paramètres nommés n'ont pas ces limitations, car ils ne sont pas positionnels, mais associatifs ; ils sont identifiés par leur nom explicite et unique.

10. Types de paramètres de message

Cette Section définit les types de valeur de paramètre qui sont utilisés pour les définitions de messages (Section 11). Avant d'utiliser une valeur de paramètre, un agent OCP DOIT vérifier si la valeur a le type attendu (c'est-à-dire, si elle se conforme à toutes les règles provenant de la définition de type). Une seule violation de règle signifie que le paramètre est invalide. Voir à la Section 5 les règles de traitement des entrées invalides.

Les extensions à OCP PEUVENT définir leurs propres types. Si elles le font, elles DOIVENT définir des types avec exactement un format de base et elles DOIVENT spécifier le type de chaque nouvel élément de protocole qu'elles introduisent.

10.1 uri

uri : étend atome ;

Uri (identifiant universel de ressource) est un atome formaté conformément aux règles d'URI de la [RFC2396].

Souvent, un paramètre uri est utilisé comme identifiant unique (au sein d'une certaine portée). La sémantique de uri est incomplète sans la spécification de la portée. De nombreux paramètres uri sont des URL. Sauf notation contraire, les identifiants URL n'impliquent pas l'existence d'une ressource desservable à la localisation qu'ils spécifient. Par exemple, une demande HTTP pour un identifiant URI fondé sur HTTP peut résulter en une réponse 404 (Pas trouvé).

10.2 uni

uni : étend atome ;

Uni (identifiant numérique unique) est un atome formaté en nombre décimal avec une valeur dans la gamme de 0 à 2 147 483 647 inclus.

Un paramètre uni est utilisé comme identifiant univoque (au sein d'une certaine portée). La sémantique de uni est incomplète sans la spécification de portée. Certains messages OCP créent des identifiants (c'est-à-dire, les introduisent dans la portée). Certains messages OCP les détruisent (c'est-à-dire, les retirent de la portée). Un agent OCP NE DOIT PAS créer la même valeur de uni plus d'une fois au sein de la même portée. Lors de la création d'un nouvel identifiant du même type et au sein de la même portée qu'un vieil identifiant, un agent OCP DOIT utiliser une valeur numérique supérieure pour le nouvel identifiant. La première règle fait que les identifiants uni conviennent pour les enregistrement qui font référence l'un à l'autre et autres artefacts. La seconde règle rend efficaces les vérifications de la première règle possible.

Par exemple, un identifiant de transaction précédemment utilisé "xid" ne doit pas être utilisé pour un nouveau message "Début de transaction" (TS) au sein de la même transaction OCP, même si un message "Fin de transaction" (TE) antérieur avait été envoyé pour la même transaction.

Un agent OCP DOIT terminer l'état associé à une portée d'unicité d'uni si toutes les valeurs uniques ont été utilisées.

10.3 size

size : étend atome ;

Size est un atome formaté comme nombre-décimal et avec une valeur dans la gamme de 0 à 2 147 483 647, inclus.

La valeur de Size est le nombre d'octets dans le tronçon de données associé.

Le cœur OCP ne peut pas traiter les messages d'application qui excèdent une taille de 2 147 483 647, qui exigent des tailles supérieures au titre du processus de triage OCP, ou qui utilisent des tailles avec une granularité autre que 8 bits. Cette limitation peut être traitée par des extensions à OCP, comme indiqué au paragraphe 15.1.

10.4 offset

offset : étend atome ;

Offset est un atome formaté comme nombre-décimal et avec une valeur dans la gamme de 0 à 2 147 483 647, inclus.

Offset est une position d'octet exprimée en nombre d'octets par rapport au premier octet du flux de données associé. Le décalage du premier octet de données a une valeur de zéro.

10.5 percent

percent : étend atome ;

Percent est un atome formaté comme nombre-décimal avec une valeur dans la gamme de 0 à 100, inclus.

La sémantique de percent est incomplète si sa valeur n'est pas associée à une déclaration ou assertion booléenne. La valeur de 0 indique une impossibilité absolue. La valeur de 100 indique une certitude absolue. Dans l'un et l'autre cas, la déclaration associée peut être tenue pour certaine comme si elle était exprimée comme booléen plutôt qu'en termes probabilistes. Les valeurs dans l'intervalle [1, 99] inclus indiquent des niveaux correspondants de certitude que la déclaration associée est vraie.

10.6 boolean

boolean : étend atome ;

Le type boolean est un atome avec deux valeurs valides : vrai et faux. Un paramètre booléen exprime la vérité de la déclaration associée.

10.7 xid

xid : étend uni ;

Xid, identifiant de transaction OCP, identifie de façon univoque une transaction OCP au sein d'une connexion OCP.

10.8 sg-id

sg-id : étend uni ;

Sg-id, identifiant de groupe de services, identifie de façon univoque un groupe de services sur une connexion OCP.

10.9 modp

modp : étend percent ;

Modp étend le type percent pour exprimer la confiance de l'envoyeur quand à la modification des données d'application. La déclaration booléenne associée à la valeur de pourcentage est "les données seront modifiées". La modification est définie comme une adaptation qui change la valeur numérique d'au moins un octet de données.

10.10 result

```
result : étend structure avec {
        atome [atome];
  };
```

Le résultat du traitement OCP est exprimé comme une structure avec deux membres documentés : un code d'état uni exigé et une raison facultative. Le membre raison contient des informations textuelles non destinées à un traitement automatique. Par exemple :

```
{ 200 OK }
{ 200 "6:eu" }
{ 200 "27:27 octets en codage UTF-8" }
```

La présente spécification définit les codes d'état de résultat suivants :

Code	Texte	Sémantique
200	OK	Succès global. La présente spécification ne contient aucune action générale pour la réception d'un code d'état 200.
206	partiel	Succès partiel. Ce code d'état est documenté pour les seuls messages de fin d'application (AME). Le code indique que l'agent a terminé prématurément le flux de données correspondant (c'est-à-dire que plus de données seraient nécessaires pour reconstruire un message d'application complet). La terminaison prématurée d'un flux de données n'introduit aucune exigence particulière sur la terminaison des autres flux de données. Voir les cas d'utilisation à la Section 8 sur l'optimisation de la terminaison des flux de données.
400	échec	Erreur, exception, ou problème. Le receveur d'un résultat 400 (échec) d'un message AME, TE, ou CE DOIT détruire tout état ou données associés au flux de données, transaction, ou connexion correspondant. Par exemple, une version adaptée des données de message d'application doit être purgée de l'antémémoire du processeur si le processeur OPES reçoit un message de fin d'application (AME) avec un code de résultat de 400.

Des messages OCP spécifiques peuvent exiger des actions spécifiques du code.

Il est possible d'étendre la sémantique de result par l'ajout de nouveaux membres de structure "result" ou par la négociation de codes de résultat supplémentaires (par exemple, au titre d'un profil négocié). Le receveur d'un code de résultat inconnu (dans le contexte alors en cours) DOIT agir comme si le code 400 (échec) était reçu.

Le receveur d'un message sans le paramètre de résultat réel, mais avec un paramètre de résultat formel facultatif, DOIT agir comme si le code 200 (OK) était reçu.

Les informations de texte (le second paramètre anonyme de la structure de résultat) sont souvent appelées "raison" ou "phrase de raison". Pour aider aux efforts de réparation manuelles, les agents OCP sont invités à inclure des descriptions des raisons avec tout résultat indiquant un échec.

Dans la présente spécification, un message OCP avec un code d'état de résultat de 400 (échec) est appelé "un message indiquant un échec".

10.11 feature

feature : étend structure avec {
uri;
};

Le type feature étend une structure pour relayer un identifiant de caractéristique OCP et pour réserver une "place" pour des paramètres facultatifs spécifiques d'une caractéristique (parfois appelés des attributs de caractéristique). Les valeurs de feature sont utilisées pour déclarer la prise en charge de caractéristiques OCP et en négocier l'utilisation.

La présente spécification ne définit aucune caractéristique.

10.12 features

features : étend la liste de caractéristiques ;

Features est une liste de valeurs de feature. Sauf notation contraire, la liste peut être vide, et les caractéristiques sont énumérées en ordre décroissant de préférence.

10.13 service

service : étend structure avec {
uri;
};

La structure service a un membre anonyme, un identifiant de service OPES de type uri. Les services peuvent avoir des paramètres dépendants du service. Une extension OCP définissant un service à utiliser avec OCP DOIT définir un identifiant de service et des paramètres dépendants du service, si il y en a, comme membres supplémentaires de la structure "service". Par exemple, une valeur de service peut ressembler à :

{"41:http://www.iana.org/assignments/opes/ocp/tls" "8:blowfish"}

10.14 services

services : étend liste de services ;

Services est une liste de valeurs de service. Sauf notation contraire; la liste peut être vide, et l'ordre des valeurs est celui demandé ou l'ordre réel de l'application de service.

10.15 Spécialisations des flux de données

Plusieurs types de paramètres, comme un décalage s'appliquent au flux de données à la fois original et adapté. Il est relativement aisé de confondre l'affiliation de flux de données d'un type, en particulier lorsque des paramètres avec des affiliations différentes sont mêlés dans une déclaration de message. Les déclarations suivantes sont pour de nouveaux types spécifiques de flux de données en utilisant leur version qui ignore le flux de données (noté par un paramètre fictif <type>).

Les nouveaux types suivants se réfèrent seulement aux données originales : org-<type> : étend <type> ;

Les nouveaux types suivants se réfèrent seulement aux données adaptées : adp-<type> : étend <type> ;

Les nouveaux types suivants se réfèrent seulement au flux de données de l'envoyeur : my-<type> : étend <type> ;

Les nouveaux types suivants se réfèrent seulement au flux de données du receveur : your-<type> : étend <type> ;

Le cœur OCP utilise le schéma de désignation de type ci-dessus pour mettre en œuvre une spécialisation de flux de données pour les types suivants : offset, size, et sg-id. Les extensions à OCP DEVRAIENT utiliser le même schéma.

11 Définitions de message

Cette Section décrit les messages spécifiques de OCP. Chaque message reçoit un nom unique et a généralement un ensemble de paramètres anonymes et/ou nommés. L'ordre des paramètres anonymes est spécifié dans la définition du message. Aucun ordre particulier n'est impliqué pour les paramètres nommés. Les extensions à OCP NE DOIVENT PAS introduire de paramètres nommés dépendants de l'ordre. Pas plus d'un paramètre nommé avec un certain nom ne peut apparaître dans le message ; les messages avec plusieurs paramètres de même noms sont invalides.

Un receveur DOIT être capable d'analyser tout message de format valide (voir au paragraphe 3.1) sous réserve des limitations des ressources du receveur.

Les noms de message, paramètres, et charges utiles inconnus ou inattendus peuvent être des extensions valides. Par exemple, un paramètre nommé "extra" peut être utilisé pour un certain message, en plus de ce qui est documenté dans la définition du message ci-dessous. Un receveur DOIT ignorer tout nom valide mais inconnu ou inattendu de paramètre, membre, ou charge utile.

Certaines valeurs de paramètre de message utilisent des identifiants uni pour se référer à divers états OCP (voir le paragraphe 10.2 et l'Appendice B). Ces identifiants sont créés, utilisés, et supprimés par les agents OCP via les messages correspondants. Sauf quand on crée un nouvel identifiant, un agent OCP NE DOIT PAS envoyer d'identifiant uni qui corresponde à un état inactif (c'est-à-dire, qui n'a jamais été créé ou est déjà supprimé). De tels identifiants invalident le message OCP de l'hôte (voir la Section 5). Par exemple, le receveur doit terminer la transaction quand le paramètre xid dans un message "Data Use Mine" (DUM) se réfère à une transaction OCP inconnue ou déjà terminée.

11.1 Début de connexion (CS, *Connection Start*)

CS : étend message ;

Un message "Début de connexion" (CS, *Connection Start*) indique le début d'une connexion OCP. Un agent OCP DOIT envoyer ce message avant d'envoyer aucun autre message sur la connexion. Si le premier message qu'un agent OCP reçoit n'est pas CS, l'agent DOIT terminer la connexion avec un message "Fin de connexion" (CE, *Connection End*) ayant un code d 'état de résultat de 400 (échec). Un agent OCP DOIT envoyer le messages CS exactement une fois. Un agent OCP DOIT ignorer les messages CS répétés.

À tout moment, un serveur d'invocation PEUT refuser tout autre traitement sur une connexion OCP en envoyant un message CE avec le code d'état 400 (échec). Noter que l'exigence d'envoyer en premier un message CS s'applique toujours.

Avec TCP/IP comme transport, les connexions TCP brutes (adresses IP locale et distante d'homologue avec numéros d'accès) identifient une connexion OCP. D'autres transports peuvent fournir des identifiants de connexion OCP pour distinguer les connexions logiques qui partagent le même transport. Par exemple, un seul canal BEEP [RFC3080] peut être désigné comme une seule connexion OCP.

11.2 Fin de connexion (CE, *Connection End*)

```
CE : étend message avec {
        [result];
  };
```

Un message "Fin de connexion" (CE) indique la fin d'une connexion OCP. L'agent qui initie la clôture ou terminaison d'une connexion DOIT envoyer ce message immédiatement avant de clore ou terminer. Le receveur DOIT libérer l'état associé, y compris l'état de transport.

La terminaison de connexion sans un message CE indique que la connexion a été close prématurément, peut-être à l'insu ou sans l'intention de l'agent du côté qui clôt. Quand un agent OCP détecte une connexion close prématurément, l'agent DOIT agir comme si un message CE indiquant une défaillance avait été reçu.

Un message CE implique la fin de toutes les transactions, négociations, et groupes de service ouverts ou actifs sur la connexion qui se termine.

11.3 Groupe de service créé (SGC, *Service Group Created*)

```
SGC : étend message avec {
        my-sg-id services;
  };
```

Un message "Groupe de service créé" (SGC, *Service Group Created*) informe le receveur qu'une liste de services d'adaptation a été associée à l'identifiant de groupe de service ("my-sg-id"). Suite à ce message, l'envoyeur peut se référer au groupe en utilisant l'identifiant. Le receveur DOIT conserver l'association jusqu'à ce qu'un message "Groupe de service détruit" (SGD, *Service Group Destroyed*) correspondant soit reçu ou que la connexion OCP correspondante soit close.

Les groupes de service ont une portée de connexion. Les messages de gestion de transaction n'affectent pas les groupes de service existants.

Conserver les associations de groupe de service exige des ressources (par exemple, la mémorisation pour conserver l'identifiant de groupe et une liste des identifiants de service). Donc, il y a un nombre fini d'associations qu'une mise en œuvre peut conserver. Les serveurs d'invocation DOIVENT être capables de conserver au moins une association pour chaque connexion OCP qu'ils acceptent. Si un receveur d'un message SGC ne crée pas l'association demandée, il DOIT immédiatement terminer la connexion avec un message "Fin de connexion" (CE, *Connection End*) indiquant un échec.

11.4 Groupe de service détruit (SGD, *Service Group Destroyed*)

```
SGD : étend message avec {
        my-sg-id;
  };
```

Un message "Groupe de service détruit" (SGD, *Service Group Destroyed*) donne pour instruction au receveur d'oublier le groupe de service associé à l'identifiant spécifié. Le receveur DOIT détruire l'association de groupe de service identifiée.

11.5 Début de transaction (TS, T*ransaction Start*)

```
TS : étend message avec {
        xid my-sg-id;
  };
```

Envoyé par un processeur OPES, un message "Début de transaction" (TS, *Transaction Start*) indique le début d'une transaction OCP. À réception de ce message, le serveur d'invocation PEUT refuser de poursuivre le traitement de la transaction en répondant avec un message "Fin de transaction" (TE, *Transaction End*) correspondant. Un serveur d'invocation DOIT conserver l'état jusqu'à ce qu'il reçoive un message indiquant la fin de la transaction ou jusqu'à ce qu'il termine lui-même la transaction.

L'identifiant exigé "my-sg-id" se réfère à un groupe de service créé avec un message "Groupe de service créé" (SGC, *Service Group Created*). Le serveur d'invocation DOIT appliquer la liste des services associés au "my-sg-id", dans l'ordre spécifié.

Ce message introduit l'identifiant de transaction (xid).

11.6 Fin de transaction (TE, *Transaction End*)

```
TE : étend message avec {
        xid [result];
  };
```

Un message "'Fin de transaction" (TE, *Transaction End*) indique la fin de la transaction OCP identifiée.

Un agent OCP DOIT envoyer un message TE immédiatement après la prise de décision d'envoyer plus de messages relatifs à la transaction correspondante. Violer cette exigence peut causer, par exemple, des délais inutiles, le rejet de nouvelles transactions, et même des fins de temporisation pour les agents qui s'appuient sur cette condition de fin de fichier pour continuer le traitement.

Ce message termine la vie de l'identifiant de transaction (xid).

11.7 Début de message d'application (AMS, *Application Message Start*)

```
AMS : étend message avec {
        xid;
        [Services: services];
  };
```

Un message "Début de message d'application" (AMS, *Application Message Start*) indique le début du traitement du message d'application original ou adapté et du flux de données. Le receveur PEUT refuser de poursuivre le traitement par l'envoi d'un message "Fin de message d'application" (AME, *Application Message End*) indiquant un échec.

Quand un message AMS est envoyé par le processeur OPES, le serveur d'invocation renvoie généralement un message AMS, annonçant la création d'une version adaptée du message d'application original. Cette annonce peut être retardée. Par exemple, le serveur d'invocation peut attendre d'avoir plus d'informations de la part du processeur OPES.

Quand un message AMS est envoyé par le serveur d'invocation, un paramètre facultatif "Services" décrit les services OPES que le serveur PEUT appliquer au message d'application original. Généralement, la valeur "services" correspond à ce qui était demandé par le processeur OPES. Le serveur d'invocation DEVRAIT envoyer un paramètre "Services" si sa valeur différerait de la liste des services demandés par le processeur OPES. Comme le même service peut être connu sous de nombreux noms, la discordance n'implique pas nécessairement une erreur.

11.8 Fin de message d'application (AME, *Application Message End*)

```
AME : étend message avec {
        xid [result];
  };
```

Un message "Fin de message d'application" (AME, *Application Message End*) indique la fin du traitement du message d'application original ou adapté et du flux de données. Le receveur devrait ne plus attendre d'autres données pour le message d'application correspondant.

Un message AME termine les engagements de préservation des données et tout autre état associé au message d'application correspondant.

Un agent OCP DOIT envoyer un message AME immédiatement après la décision d'arrêter le traitement de son message d'application. Violer cette exigence peut causer, par exemple, des délais inutiles, le rejet de nouvelles transactions, et même des fins de temporisation pour les agents qui s'appuient sur cette condition de fin de fichier pour poursuivre le traitement.

11.9 Utiliser mes données (DUM, *Data Use Mine*)

```
DUM : étend message avec {
        xid my-offset;
        [As-is: org-offset];
        [Kept: org-offset org-size ];
        [Modp: modp];
  } et payload;
```

Un message "Utiliser mes données" (DUM, *Data Use Mine*) porte des données d'application.C'est le seul message de cœur OCP qui ait une charge utile documentée. L'envoyeur NE DOIT PAS faire le moindre trou dans les données fournies par les messages DUM et "Utiliser tes données" (DUV, *Data Use Yours*) (c'est-à-dire, le my-offset du prochain message de données doit être égal au my-offset plus la taille de charge utile du message de données précédent). Les messages avec des trous sont invalides. L'envoyeur DOIT envoyer la charge utile et PEUT utiliser des charges utiles vides (c'est-à-dire, une charge utile avec une taille de zéro). Un message DUM sans charge utile est invalide. Les charges utiles vides sont utiles pour communiquer des méta-informations sur les données (par exemple, des prédictions de modification ou des engagements de préservation) sans envoyer de données.

Un processeur OPES PEUT envoyer un paramètre "Kept" pour indiquer son engagement actuel de préservation de données (Section 7) pour les données originales. Quand un processeur OPES envoie un paramètre "Kept", le processeur DOIT garder une copie des données spécifiées (l'engagement de préservation débute ou se continue). Le paramètre "Kept offset" spécifie le décalage du premier octet des données préservées. Le paramètre "Kept size" est la taille des données préservées.

Noter que les règles de préservation de données permettent (c'est-à-dire, n'interdisent pas) à un processeur OPES de diminuer le décalage et de spécifier une gamme de données non encore pleinement livrées au serveur d'invocation. Le cœur OCP n'exige aucune relation entre la charge utile de DUM et le paramètre "Kept".

Si la valeur du paramètre "Kept" viole les règles de préservation des données mais si le receveur n'a pas encore envoyé de messages "Utiliser vos données" (DUY, *Data Use Yours*) pour la transaction OCP concernée, alors le receveur NE DOIT PAS utiliser de données préservées pour cette transaction (c'est-à-dire, il ne doit pas envoyer de messages DUY)). Si la valeur du paramètre "Kept" viole les règles de préservation des données et si le receveur a déjà envoyé des messages DUY, le message DUM est invalide, et les règles de la section 5 s'appliquent. Ces exigences aident à préserver l'intégrité des données quand l'optimisation "Kept" est utilisée par le processeur OPES.

Un serveur d'invocation DOIT envoyer un paramètre "Modp" si le serveur peut fournir une valeur fiable et n'a pas encore envoyé la même valeur de paramètre pour le message d'application correspondant. La définition de "fiable" relève entièrement du serveur d'invocation. La prédiction de modification des données inclut la charge utile de DUM. C'est-à-dire que si la charge utile jointe a été modifiée, la valeur de modp ne peut pas être 0 %.

Un serveur d'invocation DEVRAIT envoyer un paramètre "As-is" si les données jointes sont identiques à un fragment au décalage spécifié dans le flux de données original. Un paramètre "As-is" qui spécifie un fragment de données qui n'a pas été vu du serveur d'invocation est invalide. Le receveur DOIT ignorer les paramètres "As-is" invalides. Identique signifie que tous les octets adaptés ont la même valeur numérique que les octets originaux correspondants. Ce paramètre est destiné à permettre une optimisation partielle de la préservation des données sans engagement de préservation. Les données préservées traversent toujours deux fois la connexion avec le serveur d'invocation, mais le processeur OPES peut être capable d'optimiser son traitement des données.

Le processeur OPES NE DOIT PAS terminer son engagement de préservation des données (Section 7) en réaction à la réception d'un message DUM.

11.10 Utiliser tes données (DUY, *Data Use Yours*)

```
DUY : étend message avec {
        xid org-offset org-size;
  };
```

Le serveur d'invocation dit au processeur OPES d'utiliser les "size" octets de données originales préservées, en commençant au décalage spécifié, comme si ce tronçon de données venait du serveur d'invocation dans un message DUM.

Le processeur OPES NE DOIT PAS terminer son engagement de préservation des données (Section 7) en réaction à la réception d'un message DUY.

11.11 Intérêt à la préservation des données (DPI, *Data Preservation Interest*)

```
DPI : étend message avec {
        xid org-offset org-size;
  };
```

Le message "Intérêt à la préservation des données" (DPI, *Data Preservation Interest*) décrit un tronçon de données originales en utilisant comme paramètres le décalage de premier octet et la taille. Le tronçon est la seule zone de données originales qui peut intéresser le serveur d'invocation en s'y référant dans de futurs messages DUY. Ce tronçon de données est appelé les "données réutilisables". Le reste des données originales est appelé les "données disponibles". Donc, les données disponibles consistent en les octets en dessous du décalage spécifié et au décalage (offset + size) ou au dessus.

Après l'envoi de ce message, le serveur d'invocation NE DOIT PAS envoyer de messages DUY se référant aux tronçons de données disponibles. Si un processeur OPES ne préserve pas certaines données réutilisables, il PEUT commencer à préserver ces données. Si un processeur OPES préserve des données disponibles, il PEUT arrêter de préserver ces données. Si un processeur OPES ne préserve pas certaines données disponibles, il NE PEUT PAS commencer à préserver ces données.

Un serveur d'invocation NE DOIT PAS indiquer des zones de données réutilisables qui se chevauchent avec des zones de données disponibles indiquées dans des messages DPI précédents. En 'autres termes, les données réutilisables ne doivent pas grossir, et les données disponibles ne doivent pas diminuer. Si un serveur d'invocation viole cette règle, le message DPI est invalide (voir la Section 5).

Le message DPI ne peut pas forcer le processeur OPES à préserver les données. Dans ce contexte, le terme réutilisable est mis pour marquer l'intérêt du serveur d'invocation pour la réutilisation des données à l'avenir, avec la coopération du processeur OPES.

Par exemple, une valeur de décalage de zéro et la valeur de taille de 2 147 483 647 indiquent que le serveur peut vouloir réutiliser toutes les données originales. La valeur de taille de zéro indique que le serveur ne va plus envoyer d'autres messages DUY.

11.12 Veux arrêter de recevoir des données (DWSR, *Want Stop Receiving Data*)

```
DWSR : étend message avec {
        xid org-size;
  };
```

Le message 'Veux arrêter de recevoir des données" (DWSR, *Want Stop Receiving Data*) informe le processeur OPES que le serveur d'invocation veut arrêter de recevoir des données originales à tout moment après avoir reçu au moins une quantité org-size du préfixe d'un message d'application. C'est-à-dire que le serveur demande au processeur de terminer le flux de données original prématurément (voir le paragraphe 8.1) après l'envoi d'au moins org-size octets.

Un processeur OPES qui reçoit un message DWSR DEVRAIT terminer le flux de données original en envoyant un message AME avec un code d'état de 206 (partiel).

Un processeur OPES NE DOIT PAS terminer son engagement de préservation de données (Section 7) en réaction à la réception d'un message DWSR. Tout comme avec tous les autres messages, un processeur OPES peut utiliser les informations fournies par DWSR pour décider des engagements de préservation futurs.

11.13 Veux arrêter d'envoyer des données (DWSS, *Want Stop Sending Data*)

```
DWSS : étend message avec {
        xid;
  };
```

Le message "Veux arrêter d'envoyer des données" (DWSS, *Want Stop Sending Data*) informe le processeur OPES que le serveur d'invocation veut arrêter d'envoyer des données adaptées aussitôt que possible ; le serveur demande au processeur la permission de terminer prématurément le flux de données adaptées (voir au paragraphe 8.2). Le processeur OPES peut accorder cette permission en utilisant un message DSS.

Une fois le message DWSS envoyé, le serveur d'invocation NE DOIT PAS terminer prématurément le flux de données adaptées jusqu'à ce que le serveur reçoive un message DSS du processeur OPES. Si le serveur viole cette règle, le processeur OPES DOIT agir comme si aucun message DWSS n'avait été reçu. Cela implique que la transaction OCP est terminée par le processeur, avec une erreur.

Un processeur OPES qui reçoit un message DWSS DEVRAIT répondre avec un message DSS, pourvu que le processeur ne viole pas les exigences du message DSS en le faisant. Le processeur DEVRAIT répondre immédiatement une fois que les exigences du message DSS peuvent être satisfaites.

11.14 Arrêter d'envoyer des données (DSS, *Stop Sending Data*)

```
DSS : étend message avec {
        xid;
  };
```

Le message " Arrêter d'envoyer des données" (DSS, *Stop Sending Data*) donne pour instruction au serveur d'invocation de terminer prématurément le flux de données adaptées en envoyant un message AME avec un code d'état 206 (partiel). Un serveur d'invocation est supposé solliciter le message DSS en envoyant un message DWSS (paragraphe 8.2).

Un serveur d'invocation qui reçoit un message DSS sollicité pour un flux de données adaptées non encore terminé DOIT immédiatement terminer le flux de données en envoyant un message AME avec un code d'état 206 (partiel). Si le serveur d'invocation a déjà terminé le flux de données adaptées, le serveur d'invocation DOIT ignorer le message DSS. Un serveur d'invocation qui reçoit un message DSS non sollicité pour un flux de données non encore terminé DOIT soit traiter de

message comme invalide, soit comme sollicité (c'est-à-dire, le serveur ne peut pas simplement ignorer les messages DSS non sollicités).

Le processeur OPES qui envoie un message DSS DOIT être capable de reconstruire le message d'application adapté correctement après que le serveur d'invocation termine le flux de données. Cette exigence implique que le processeur doit avoir accès à toutes les données originales envoyées au serveur d'invocation après le message DSS, si il en est. Par conséquent, le processeur OPES doit ne pas envoyer de données du tout ou en garder une copie.

Si un serveur d'invocation reçoit un message DSS et, en violation des règles ci-dessus, attend plus de données originales avant d'envoyer une réponse AME, un blocage peut s'ensuivre : le processeur OPES peut attendre le message AME pour envoyer plus de données originales.

11.15 Veux une pause de données (DWP, *Want Paused Data*)

```
DWP : étend message avec {
        xid your-offset;
  };
```

Le message "Veux une pause de données" (DWP, *Want Paused Data*) indique le manque d'intérêt temporaire de l'envoyeur pour la réception de données commençant avec le décalage spécifié. Ce désintérêt n'implique rien quant à l'intention de l'envoyeur d'envoyer des données.

Le paramètre "your-offset" se réfère au flux de données qui a pour origine l'agent OCP qui reçoit le paramètre.

Si, au moment de la réception du message DWP, le receveur a déjà envoyé les données au décalage spécifié, le receveur du message DOIT arrêter immédiatement d'envoyer des données. Autrement, le receveur DOIT arrêter immédiatement d'envoyer des données après l'envoi du décalage spécifié. Une fois que le receveur a arrêté d'envoyer des données, il DOIT immédiatement envoyer un message "Pause de mes données" (DPM, *Paused My Data*) et NE DOIT PAS envoyer plus de données jusqu'à ce qu'il reçoive un message "Veux plus de données" (DWM, *Want More Data*).

Comme le sont la plupart des mécanismes de cœur OCP, la pause des données est asynchrone. L'envoyeur du message DWP NE DOIT PAS s'imaginer que la pause des données va se faire exactement au décalage spécifié, ou même du tout.

11.16 Pause de mes données (DPM, *Paused My Data*)

```
DPM : étend message avec {
        xid;
  };
```

Le message "Pause de mes données" (DPM, *Paused My Data*) indique l'engagement de l'envoyeur de ne plus envoyer de données jusqu'à ce qu'il reçoive un message "Veux plus de données" (DWM, *Want More Data*).

Le receveur du message DPM PEUT s'attendre à ce qu'il y ait une pause dans la livraison des données. Si le receveur reçoit des données en dépit de cette attente, il PEUT interrompre la transaction correspondante avec un message "Fin de transaction" (TE) indiquant un échec.

11.17 Veux plus de données (DWM, *Want More Data*)

```
DWM : étend message avec {
        xid;
        [Size-request: your-size];
  };
```

Le message "Veux plus de données" (DWM, *Want More Data*) indique que l'envoyeur a besoin de plus de données.

Les paramètres de message se réfèrent toujours au flux de données généré chez l'autre agent OCP. Lorsque il est envoyé par un processeur OPES, your-size est adp-size ; quand il est envoyé par un serveur d'invocation, your-size est org-size.

Le paramètre "Size-request" *(demande de taille)* se réfère au flux de données généré chez l'agent OCP qui reçoit le paramètre. Si un paramètre "Size-request" est présent, sa valeur est la taille de données minimum suggérée. Il est destiné à

permettre au receveur de livrer les données en moins de tronçons. Le receveur PEUT ignorer le paramètre "Size-request". L'absence d'un paramètre "Size-request" implique "n'importe quelle taille".

Ce message annule aussi l'effet du message DPM. Si le receveur n'envoyait pas de données à cause de son message DPM, le receveur PEUT reprendre l'envoi des données. Noter cependant que le message DWM peut être envoyé sans considération de si le flux de données en question est en pause. Le paramètre "Size-request" fait de ce message une utile optimisation autonome.

11.18 Offre de négociation (NO, *Negotiation Offer*)

```
NO : étend message avec {
        features;
        [SG: my-sg-id];
        [Offer-Pending: boolean];
  };
```

Un message "Offre de négociation" (NO, *Negotiation Offer*) sollicite le choix d'une seule "meilleure" caractéristique parmi une liste fournie, en utilisant un message "Réponse de négociation" (NR, *Negotiation Response*). L'envoyeur est supposé faire la liste de ses caractéristiques préférées quand c'est possible. Le receveur PEUT ignorer les préférences de l'envoyeur. Si la liste de caractéristiques est vide, la négociation est vouée à l'échec mais reste valide.

Le processeur OPES et le serveur d'invocation sont tous deux autorisés à envoyer des messages NO. Les règles de cette section assurent qu'une seule offre est honorée si deux offres sont soumises concurremment. Un agent NE DOIT PAS envoyer un message NO si il attend encore une réponse à son offre précédente sur la même connexion.

Si un processeur OPES reçoit un message NO alors que sa propre offre est en instance, il DOIT ignorer l'offre du serveur. Autrement, il DOIT répondre immédiatement. Si un serveur d'invocation reçoit un message NO alors que sa propre offre est en instance, il DOIT ignorer sa propre offre. Dans l'un et l'autre cas, le serveur DOIT répondre immédiatement.

Si un agent reçoit une séquence de messages qui viole une des règles ci-dessus, il DOIT terminer la connexion avec un message "Fin de connexion" (CE) indiquant un échec.

Un paramètre "Offer-Pending" *(offre en cours)* facultatif est utilisé pour la maintenance de la phase de négociation (paragraphe 6.1). La valeur par défaut de l'option est "faux".

Un paramètre "SG" est utilisé pour rétrécir la portée des négociations au groupe de service spécifié. Si SG est présent, les caractéristiques négociées ne sont négociées et activées que pour les transactions qui utilisent l'identifiant de groupe de service spécifié. Les caractéristiques dont la portée est la connexion sont négociées et activées pour tous les groupes de service. La présence d'une portée n'implique pas une résolution automatique de conflit commune aux langages de programmation ; aucun conflit n'est permis. Dans la négociation de caractéristiques dont la portée est la connexion, un agent DOIT vérifier qu'il n'y a pas de conflit au sein de chaque groupe de service existant. Lors de la négociation de caractéristiques dont la portée est le groupe, un agent DOIT vérifier qu'il n'y a pas de conflit avec les caractéristiques à portée de connexion déjà négociées. Par exemple, il ne doit pas être possible de négocier un profil d'application HTTP à portée de connexion si un groupe de service a déjà un profil d'application SMTP, et vice versa.

Les agents OCP NE DEVRAIENT PAS envoyer d'offres avec des groupes de service utilisés par des transactions en instance. Sauf notation explicite contraire dans une documentation de caractéristique, les agents OCP NE DOIVENT PAS appliquer de négociation aux transactions en cours. En d'autres termes, les caractéristiques négociées prennent effet avec la nouvelle transaction OCP.

Comme avec les autres éléments de protocole, les extensions de cœur OCP peuvent documenter des restrictions de négociation supplémentaires. Par exemple, la spécification d'une caractéristique de sécurité du transport peut interdire l'utilisation du paramètre SG dans les offres de négociation, pour éviter des situations où le chiffrement n'est activé que pour une portion des transactions qui se chevauchent sur la même connexion de transport.

11.19 Réponse de négociation (NR, *Negotiation Response*)

```
NR : étend message avec {
        [feature];
        [SG: my-sg-id];
        [Rejects: features];
```

```
        [Unknowns: features];
        [Offer-Pending: boolean];
  };
```

Un message "Réponse de négociation" (NR, *Negotiation Response*) porte la réaction du receveur à la demande d'offre de négociation (NO). Une réponse acceptée (autrement dit, une réponse positive) est indiquée par la présence d'un paramètre anonyme "feature", contenant la caractéristique choisie. Si la caractéristique choisie ne correspond à aucune des caractéristiques offertes, l'agent qui offre DOIT considérer que la négociation a échoué et PEUT terminer la connexion avec un message "Fin de connexion" (CE) indiquant un échec.

Une offre rejetée (réponse négative) est indiquée en omettant le paramètre anonyme "feature".

La caractéristique dont la négociation a réussi devient effective immédiatement. L'envoyeur d'une réponse positive DOIT considérer la caractéristique correspondante comme activée immédiatement après l'envoi de la réponse ; le receveur d'une réponse positive DOIT considérer la caractéristique correspondante comme activée immédiatement après la réception de la réponse. Noter que la portée de l'application de la caractéristique négociée peut être limitée à un groupe de service spécifié. L'état de la phase de négociation n'affecte pas l'activation de la caractéristique.

Si l'offre de négociation contient un paramètre SG, celui qui répond DOIT inclure ce paramètre dans le message "Réponse de négociation" (NR). Le receveur d'un message NR sans le paramètre SG attendu DOIT traiter la réponse de négociation comme invalide.

Si l'offre de négociation n'a pas de paramètre SG, celui qui répond NE DOIT PAS inclure ce paramètre dans le message "Réponse de négociation" (NR). Le receveur d'un message NR avec un paramètre SG inattendu DOIT traiter la réponse de négociation comme invalide.

Un paramètre "Offer-Pending" facultatif est utilisé pour la maintenance de la phase de négociation (paragraphe 6.1). La valeur par défaut de l'option est "faux".

Lorsque il accepte ou rejette une offre, l'envoyeur du message NR PEUT fournir des détails supplémentaires via les paramètres Rejects et Unknowns. Le paramètre Rejects peut être utilisé pour faire la liste des caractéristiques qui étaient connues du receveur de l'offre de négociation mais n'ont pas pu être prises en charge étant donné l'état négocié qui existait quand le message NO a été reçu. Le paramètre Unknowns peut être utilisé pour faire la liste des caractéristiques qui n'étaient pas connues du receveur du NO.

11.20 Interrogation de capacités (AQ, *Ability Query*)

```
AQ : étend message avec {
        feature;
  };
```

Un message "Interrogation de capacités" (AQ, *Ability Query*) sollicite une "Réponse de capacités" (AA, *Ability Answer*) immédiate. Le receveur DOIT répondre immédiatement avec un message AA. C'est une interface en lecture seule, non modifiable. Le receveur NE DOIT PAS activer ou désactiver de caractéristique à cause d'une demande AQ.

Les extensions à OCP qui documentent une caractéristique PEUVENT étendre les messages AQ pour fournir des informations supplémentaires sur la caractéristique ou sur l'interrogation elle-même.

L'intention principale de l'interface d'interrogation de capacités est le débogage et la réparation et non le réglage fin automatisé d'un comportement et de la configuration d'agent. Cela peut être mieux réalisé par le mécanisme de négociation OCP de la Section 6.

11.21 Réponse de capacité (AA, *Ability Answer*)

```
AA : étend message avec {
       boolean;
  };
```

Un message "Réponse de" capacités" (AA, *Ability Answer*) exprime la prise en charge par l'envoyeur d'une caractéristique demandée via un message AQ. L'envoyeur DOIT régler la valeur au paramètre booléen anonyme pour la véracité de la déclaration suivante : "Au moment de la génération de la présente réponse, l'envoyeur prend en charge la caractéristique en

question". La signification de "prend en charge" et les détails supplémentaires sont spécifiques de la caractéristique. Les extensions à OCP qui documentent une caractéristique DOIVENT documenter la définition de "prise en charge" dans la portée de la déclaration ci-dessus et PEUVENT étendre les messages AA à fournir des informations supplémentaires sur la caractéristique ou sur la réponse elle-même.

11.22 Interrogation de progrès (PQ, *Progress Query*)

```
PQ : étend message avec {
        [xid];
  };
```

Un message "Interrogation de progrès" (PQ, *Progress Query*) sollicite une "Réponse de progrès (PA, *Progress Answer*) immédiate. Le receveur DOIT immédiatement répondre à une demande PQ, même si l'identifiant de transaction est invalide du point de vue du receveur.

11.23 Réponse de progrès (PA, *Progress Answer*)

```
PA : étend message avec {
        [xid];
        [Org-Data: org-size];
  };
```

Un message "Réponse de progrès" (PA, *Progress Answer*) porte l'état de l'envoyeur. La taille "Org-Data" est la taille totale des données originales reçues ou envoyées par l'agent jusqu'à présent pour le message d'application identifié (un agent peut être soit l'envoyeur, soit le receveur des données originales, de sorte qu'il n'y a pas d'ambiguïté). Lorsque on se réfère aux données reçues, les informations de progrès n'impliquent pas que les données aient par ailleurs été traitées de quelque façon que ce soit.

L'interface d'interrogation de progrès est utile à divers titres, incluant de garder en vie des connexions OCP inactives, de jauger la vitesse de traitement de l'agent, de vérifier les progrès de l'agent, et de débogage des communications OCP. La vérification des progrès, par exemple, peut être essentielle pour mettre en œuvre des temporisations pour des serveurs d'invocation qui n'envoient pas de données adaptées tant qu'ils n'ont pas reçu et traité en entier le message d'application original.

Un receveur de message PA NE DOIT PAS supposer que l'envoyeur ne travaille pas sur une transaction ou message d'application non identifié dans le message PA. Un message PA ne porte aucune information sur les multiples transactions ou messages d'application.

Si un agent travaille sur la transaction identifiée dans la demande PQ, l'agent DOIT envoyer l'identifiant de transaction correspondant (xid) quand il répond au message PQ avec un PA. Autrement, l'agent NE DOIT PAS envoyer l'identifiant de transaction. Si un agent travaille sur le message d'application original pour la transaction spécifiée, l'agent DOIT envoyer le paramètre Org-Data *(données originales)*. Si l'agent a déjà envoyé ou reçu le message AME pour le flux de données original, il NE DOIT PAS envoyer le paramètre Org-data.

De façon informelle, le message PA relaie les progrès de l'envoyeur avec la transaction et le flux de données original identifié par le message PQ, pourvu que l'identifiant de transaction soit toujours valide au moment de la réponse. Des informations absentes dans la réponse indiquent une transaction et/ou flux de données original invalides, inconnus ou fermés du point de vue du receveur de l'interrogation.

11.24. Rapport de progrès (PR, *Progress Report*)

```
PR : étend message avec {
        [xid];
        [Org-Data: org-size];
  };
```

Un message "Rapport de progrès" (PR, *Progress Report*) porte l'état de l'envoyeur. La sémantique du message et les exigences associées sont identiques à celles du message PA, sauf que le message PR est envoyé sans sollicitation. L'envoyeur PEUT rapporter ses progrès à tout moment. L'envoyeur PEUT rapporter des progrès sans relation avec une transaction ouun message d'application original ou relatif à toute transaction valide (en cours) ou flux de données original.

Les rapports de progrès non sollicités sont particulièrement utile pour les extensions à OCP qui traitent avec des services d'invocation "lents" qui introduisent des délais significatifs pour le receveur final du message d'application. Le rapport peut contenir des informations de progrès qui vont rendre ce receveur final plus tolérant quant aux délais.

12. Considérations de l'IAB

Le traitement OPES des considérations du Bureau de l'architecture de l'Internet de l'IETF (IAB) [RFC3238] sont documentées dans la [RFC3914].

13. Considérations sur la sécurité

La présente Section examine les considérations sur la sécurité pour OCP. Les menaces sur les OPES sont documentées dans la [RFC3837]

OCP relaie des messages d'application qui peuvent contenir des informations sensibles. Un chiffrement du transport approprié peut être négocié pour empêcher les fuites d'informations ou leur modification (voir la Section 6) mais les agents OCP peuvent prendre en charge par défaut un transport non chiffré. Ces configurations vont exposer les messages d'application à l'enregistrement et la modification par des tiers, même si les mandataires OPES sont eux-mêmes sûrs.

Des erreurs de mise en œuvre de OCP peuvent conduire à des vulnérabilités de la sécurité chez les agents OCP, même si le trafic OCP lui-même reste sûr. Par exemple, un débordement de mémoire tampon dans un serveur d'invocation causé par un processeur OPES malveillant peut accorder à ce processeur l'accès à des informations provenant d'autres connexions OCP (100 % sûres) incluant des connexions avec d'autres processeurs OPES.

Des mises en œuvre OCP négligentes peuvent s'appuyer sur divers identifiants OCP comme étant uniques à travers tous les agents OCP. Un agent malveillant peut injecter un message OCP qui correspond à des identifiants utilisés par d'autres agents, pour tenter d'obtenir l'accès à des données sensibles. Les mises en œuvre de OCP doivent toujours vérifier qu'un identifiant est "local" pour la connexion correspondante avant d'utiliser cet identifiant.

OCP est un protocole à états pleins. Plusieurs commandes OCP augmentent la quantité d'état que le receveur doit conserver. Par exemple, un message "Groupe de service créé" (SGC, *Service Group Created*) ordonne au receveur de conserver une association entre un identifiant de groupe de services et une liste de services.

Les mises en œuvre qui ne peuvent par traiter correctement l'épuisement de ressource augmentent les risques pour la sécurité. Voici la liste des ressource connues en rapport avec OCP qui peuvent être épuisées durant un échange de messages OCP conforme :

Structures de message OCP : la syntaxe du message OCP ne limite pas la profondeur d'incorporation des structures de message OCP et ne fixe aucune limite à la longueur (nombre d'octets) de la plupart des éléments de syntaxe.

Connexions concurrentes : OCP ne fixe pas de limite supérieure au nombre de connexions concurrentes que des messages "Début de connexion" (CS, *Connection Start*) peuvent ordonner à un serveur d'invocation de créer.

Groupes de service : OCP ne fixe pas de limite supérieure au nombre d'associations de groupes de service que des messages "Groupe de service créé" (SGC, *Service Group Created*) peuvent ordonner à un serveur d'invocation de créer.

Transactions concurrentes : OCP ne fixe pas de limite supérieure au nombre de transactions concurrentes que des messages "Début de transaction" (TS, *Transaction Start*) peuvent ordonner à un serveur d'invocation de conserver.

Flux concurrents : le cœur OCP ne fixe pas de limite supérieure au nombre de flux adaptés concurrents que des messages "Début de messages d'application" (AMS, *Application Message Start*) peuvent ordonner à un processeur OPES de maintenir.

7. Considérations relatives à l'IANA

L'IANA tient une liste des caractéristiques de OCP, incluant des profils d'application (paragraphe 10.11). Pour chaque caractéristique, sa valeur de paramètre uri est enregistrée avec les paramètres d'extension (si il en est). La syntaxe et la sémantique des caractéristiques enregistrées sont documentées avec la notation PETDM (section 9).

L'IESG est chargé d'allouer un expert désigné pour relire chaque enregistrement sur la voie de la normalisation avant l'allocation de l'IANA. La liste de diffusion du groupe de travail OPES peut être utilisée pour solliciter des commentaires pour les caractéristiques sur la voie de la normalisation et celles qui n'y sont pas.

Les extensions du cœur OCP sur la voie de la normalisation DEVRAIENT utiliser le préfixe http://www.iana.org/assignments/opes/ocp/ pour les paramètres uri de caractéristiques. Il est suggéré que l'IANA remplisse les ressources identifiées par de tels paramètres "uri" avec les enregistrements de caractéristique correspondants. Il est aussi suggéré que l'IANA tienne un index de toutes les caractéristiques OCP enregistrées à l'URL http://www.iana.org/assignments/opes/ocp/ ou sur une page liée à cet URL.

La présente spécification ne définit pas de caractéristique OCP à enregistrer par l'IANA.

15. Conformité

La présente spécification définit les conditions de conformité pour les sujets suivants : processeurs OPES (mises en œuvre de client OCP), serveurs d'invocation (mises en œuvre de serveur OCP), et extensions à OCP. Un agent OCP (processeur ou serveur d'invocation) peut aussi être appelé "envoyeur" ou "receveur" d'un message OCP.

Un sujet de conformité est conforme si il satisfait à toutes les exigences applicables marquées "DOIT" et "DEVRAIT". Par définition, satisfaire une exigence marquée "DOIT"signifie agir comme prescrit par l'exigence ; satisfaire une exigence marquée "DEVRAIT" signifie soit agir comme prescrit par l'exigence, soit avoir une raison d'agir différemment. Une exigence est applicable au sujet si elle donne une instruction (s'adresse) au sujet.

De façon informelle, la conformité à OCP signifie qu'il n'y a pas de violation connue d'une exigence marquée "DOIT", et que toutes les violations d'exigences marquées "DEVRAIT" sont délibérées. En d'autres termes, "DEVRAIT" signifie "DOIT satisfaire ou DOIT avoir une raison de violer". Il est prévu que les revendications de conformité soient accompagnées d'une liste des "DEVRAIT" non pris en charge (si il en est) dans un format approprié, expliquant pourquoi le comportement préféré n'a pas été choisi.

Seules les parties normatives de la présente spécification affectent la conformité. Les parties normatives sont celles qui sont explicitement marquée avec le mot "normatif", les définitions, et phrases contenant des mots-clés en majuscules non entre guillemets de la [RFC2119]. Par conséquent, les exemples et illustrations ne sont pas normatifs.

15.1 Extension du cœur OCP

Les extensions à OCP NE DOIVENT PAS changer le format du message de cœur OCP, comme défini par l'ABNF et les règles d'accompagnement normatives du paragraphe 3.1. Cette exigence est destinée à permettre à ceux qui voient le message OCP, aux valideurs, et logiciels "intermédiaires", d'au moins isoler et décomposer tout message OCP, même un message avec une sémantique inconnue d'eux (c'est-à-dire, étendue).

Les extensions à OCP peuvent changer les exigences normatives du cœur OCP pour les processeurs OPES et les serveurs d'invocation. Cependant, les extensions à OCP NE DEVRAIENT PAS faire ces changements et DOIVENT exiger sur un niveau marqué "DOIT" que ces changements soient négociés avant de prendre effet. De façon informelle, la présente spécification définit le comportement d'un agent OCP conforme jusqu'à ce que les changements à la présente spécification (si il en est) soient négociés avec succès.

Par exemple, si un profil RTSP pour OCP exige la prise en charge de décalages excédant 2 147 483 647 octets, la spécification du profil peut documenter les changements appropries à OCP tout en exigeant que les agents d'adaptation RTSP négocient la prise en charge de "grands décalages" avant d'utiliser de grands décalages. Cette négociation peut être liée avec la négociation d'une autre caractéristique (par exemple, négocier un profil RTSP peut impliquer la prise en charge de "grands décalages").

Comme l'impliquent les règles ci-dessus, les extensions à OCP peuvent altérer de façon dynamique le mécanisme de négociation lui-même, mais une telle alternative devrait être négociée d'abord, en utilisant le mécanisme de négociation

défini par cette spécification. Par exemple, la négociation réussie d'une caractéristique pourrait changer la valeur par défaut de "Offer-Pending" de faux en vrai.

Appendice A. Résumé des messages

Cet Appendice n'est pas normatif. Le tableau ci-dessous résume les propriétés des messages clés de OCP. Pour chaque message, le tableau donne les informations suivantes :
Nom : nom du message comme vu sur le réseau.
Titre : titre du message lisible par l'homme.
P : si cette spécification documente la sémantique du message comme envoyé par un processeur OPES.
S : si cette spécification documente la sémantique du message comme envoyé par un serveur d'invocation.
Lien : messages en rapport tels qu'une demande associée, un message de réponse, ou un message d'état associé.

Nom	Titre	P	S	Lien
AMS	Début de message d'application	X	X	AME
AME	Fin de message d'application	X	X	AMS, DSS
AQ	Interrogation de capacités	X	X	AA
AA	Réponse de capacités	X	X	AQ
CS	Début de connexion	X	X	CE
CE	Fin de connexion	X	X	CS
DUM	Utiliser mes données	X	X	DUY, DWP
DUY	Utiliser tes données	X		DUM, DPI
DPI	Intérêt à la préservation des données	X		DUY
DWSS	Veux arrêter l'envoi des données	X		DWSR, DSS
DWSR	Veux arrêter de recevoir des données	X		DWSS
DSS	Arrêt d'envoi des données	X		DWSS
DWP	Veux une pause des données	X	X	DPM
DPM	Pause de mes données	X	X	DWP, DWM
DWM	Veux plus de données	X	X	DPM
NO	Offre de négociation	X	X	NR, SGC
NR	Réponse de négociation	X	X	NO
PQ	Interrogation de progrès	X	X	PA
PA	Réponse de progrès	X	X	PQ, PR
PR	Rapport de progrès	X	X	PA
SGC	Groupe de services créé	X	X	SGD, TS
SGD	Groupe de service supprimé	X	X	SGC
TS	Début de transaction	X		TE, SGC
TE	Fin de transaction	X	X	TS

Appendice B. Résumé des états

Cet Appendice n'est pas normatif. Le tableau ci-dessous résume les états OCP. Certains états sont conservés à travers plusieurs transactions et messages d'application. Certains correspondent à un dialogue d'une seule demande/réponse ; la nature asynchrone de la plupart des échanges de message OCP exige que les agents OCP traitent d'autres messages tout en attendant une réponse à une demande et, donc, tout en maintenant l'état du dialogue.

Pour chaque état, le tableau donne les informations suivantes :
État : brève étiquette d'état.
Origine : messages qui créent cet état.
Fin : messages qui suppriment cet état.
Identifiant : identifiant associé, s'il en est.

État	Origine	Fin	Identifiant
connexion	CS	CE	-
groupe de services	SGC	SGD	sg-id
transaction	TS	TE	xid
message d'application et flux de données	AMS	AME	-
terminaison prématurée de flux de données original	DWSR	AME	-
terminaison prématurée de flux de données adaptées	DWSS	DSS AME	-

flux de données en pause	DPM	DWM	-
engagement de préservation	DUM	DPI AME	-
négociation	NO	NR	-
interrogation de progrès	PQ	PA	-
interrogation de capacités	PQ	PA	-

Appendice C. Remerciements

L'auteur remercie chaleureusement de leurs contributions Abbie Barbir (Nortel Networks), Oskar Batuner (Independent Consultant), Larry Masinter (Adobe), Karel Mittig (France Telecom R&D), Markus Hofmann (Bell Labs), Hilarie Orman (The Purple Streak), Reinaldo Penno (Nortel Networks), et Martin Stecher (Webwasher), ainsi qu'un participant anonyme au groupe de travail OPES.

Des remerciements particuliers à Marshall Rose pour son outil xml2rfc.

16. Références

16.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC2396] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiants de ressource uniformes (URI) : Syntaxe générique", août 1998. (*Obsolète, voir* RFC3986)

[RFC3835] A. Barbir et autres, "Architecture pour les services marginaux à connexion libre (OPES)", août 2004. *(Information)*

16.2 Références pour information

[OPES-RULES] Beck, A. et A. Rousskov, "P: Message Processing Language", projet non suivi, octobre 2003.

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[RFC3080] M. Rose, "Cœur du protocole extensible d'échange de blocs (BEEP)", mars 2001. *(P.S.)*

[RFC3238] S. Floyd, L. Daigle, "Considérations architecturales et de politique de l'IAB pour des services marginaux à connexion libre (OPES)", janvier 2002. *(Information)*

[RFC3752] A. Barbir et autres, "Services marginaux à connexion libre (OPES) : cas d'utilisation et scénarios de développement", avril 2004. *(Information)*

[RFC3836] A. Beck et autres, "Exigences pour les protocoles d'invocation de services marginaux à connexion libre (OPES)", août 2004. *(Information)*

[RFC3837] A. Barbir et autres, "Menaces et risques pour la sécurité des services marginaux à connexion libre (OPES)", août 2004. *(Information)*

[RFC3838] A. Barbir et autres, "Exigences de politique, d'autorisation, et de mise en application des services marginaux à connexion libre (OPES)", août 2004. *(Information)*

[RFC3897] A. Barbir, "Communication entre entités des services marginaux à connexion libre (OPES) et les points d'extrémité", septembre 2004. *(Information)*

[RFC3914] A. Barbir, A. Rousskov, "Traitement des considérations de l'IAB par les services marginaux à connexion libre (OPES)", octobre 2004. *(Information)*

[RFC4236] A. Rousskov, M. Stecher, "Adaptation HTTP avec les services marginaux à connexion libre (OPES)", novembre 2005. *(P.S.)*

Adresse de l'auteur

Alex Rousskov
The Measurement Factory

mél : rousskov@measurement-factory.com
URI : http://www.measurement-factory.com/

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.