Groupe de travail Réseau
Request for Comments : 2136
RFC mise à jour : 1035
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

P. Vixie, éditeur, ISC
S. Thomson, Bellcore
Y. Rekhter, Cisco
J. Bound, DEC
avril 1997

Mises à jour dynamiques dans le système des noms de domaine (DNS UPDATE)

Résumé
Le système des noms de domaines était à l'origine conçu pour prendre en charge les interrogations d'une base de données à configuration statique. Bien qu'il ait été prévu que les données changent, la fréquence de ces changements devait être très faible, et toutes les mises à jour devaient être faites par des interventions externes sur le fichier maître d'une zone.

En utilisant la présente spécification de l'opcode UPDATE, il est possible d'ajouter ou supprimer des RR ou des ensembles de RR d'une zone spécifiée. Les préalables sont spécifiés séparément des opérations de mise à jour, et peuvent spécifier une dépendance à l'existence ou à la non existence préalable d'un ensemble de RR (RRset) ou à l'existence d'un seul RR.

UPDATE est atomique, c'est-à-dire que tous les préalables doivent être satisfaits ou alors aucune opération de mise à jour n'aura lieu. Il n'y a pas de condition d'erreur dépendante des données définie après la satisfaction des préalables.

1 Définitions

Le présent document donne intentionnellement plus de définition aux rôles de serveur "maître", "esclave", et "maître principal", et à leur énumération dans les RR NS, et au champ SOA MNAME. Dans ce sens, les définitions de type de serveur suivantes peuvent être considérées comme un addendum à la [RFC1035], et elles sont destinées à être cohérentes avec la [RFC1996] :

esclave un serveur d'autorité qui utilise AXFR ou IXFR pour restituer la zone et est nommé dans le RRset NS de la zone.

maître un serveur d'autorité configuré pour être la source des données d'AXFR ou d'IXFR pour un ou plusieurs serveurs esclaves.

maître principal serveur maître à la racine du graphe de dépendance AXFR/IXFR. Le maître principal est nommé dans le champ SOA MNAME de la zone et facultativement par un NS RR. Il n'y a, par définition, qu'un seul serveur maître principal par zone.

Un nom de domaine identifie un nœud au sein de la structure d'arborescence de l'espace des noms de domaine. Chaque nœud a un ensemble (éventuellement vide) d'enregistrements de ressource (RR, *Resource Record*). Tous les RR qui ont le même NOM, CLASSE et TYPE sont appelés un ensemble d'enregistrements de ressource (RRset, *Resource Record Set*).

Le pseudo code utilisé dans le présent document n'est que pour des besoins de démonstration. Si il se trouve en désaccord avec le texte, c'est celui-ci qui fait foi. Si le texte est trouvé ambigu, le pseudo code peut être utilisé pour aider à résoudre l'ambiguïté.

1.1 Règles de comparaison

1.1.1 Deux RR sont considérés égaux si leurs champs NOM, CLASSE, TYPE, RDLENGTH et RDATA sont égaux. Noter que le champ Durée de vie (TTL, *time-to-live*) est explicitement exclu de la comparaison.

1.1.2 Les règles de comparaison des chaînes de caractères dans les noms sont spécifiées au paragraphe 2.3.3 de la [RFC1035].

1.1.3 L'utilisation de caractères génériques est désactivée. C'est-à-dire qu'un caractère générique ("*") dans une mise à jour ne correspond qu'à un caractère générique ("*") dans la zone, et vice versa.

1.1.4 Les alias sont désactivés : un CNAME dans la zone correspond à un CNAME dans la mise à jour, et ne sera pas suivi autrement. Toutes les opérations UPDATE sont faites sur la base des noms canoniques.

1.1.5 Les types de RR suivants ne peuvent pas être ajoutés à un RRset. Si les règles de comparaison suivantes sont satisfaites, une tentative d'ajout du nouveau RR aura alors pour résultat le remplacement du RR précédent :

SOA compare seulement NOM, CLASSE et TYPE – il n'est pas possible d'avoir plus d'un SOA *(sphère d'autorité)* par zone, même si un des champs de données diffère.

WKS compare seulement NOM, CLASSE, TYPE, ADRESSE, et PROTOCOLE – un seul RR WKS est possible pour ce tuplet, même si les gabarits de services diffèrent.

CNAME compare seulement NOM, CLASSE, et TYPE – il n'est pas possible d'avoir plus d'un RR CNAME, même si leurs champs de données diffèrent.

1.2 RR glu

Pour déterminer si un nom de domaine utilisé dans le protocole UPDATE est contenu au sein d'une zone spécifiée, un nom de domaine est "dans" une zone si il est possédé par le nom de domaine de cette zone. Voir les détails au paragraphe 7.18.

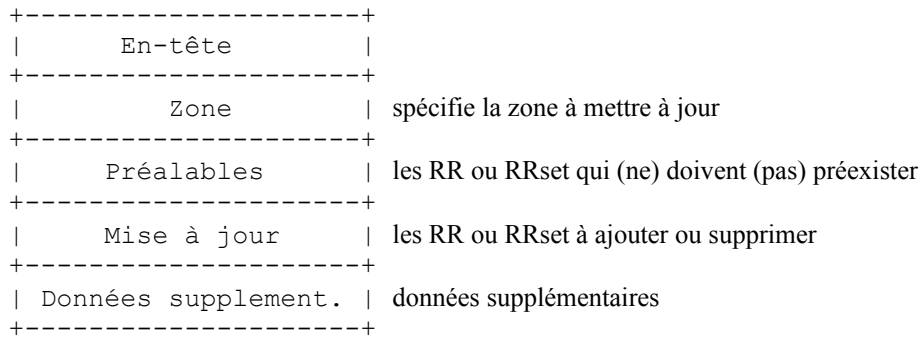
1.3 Nouveaux numéros alloués

CLASSE = AUCUN (254)
RCODE = YXDOMAIN (6)
RCODE = YXRRSET (7)
RCODE = NXRRSET (8)
RCODE = NOTAUTH (9)
RCODE = NOTZONE (10)
Opcode = UPDATE (5)

2. Format du message UPDATE

Le format du message DNS est défini par le paragraphe 4.1 de la [RFC1035]. Certaines extensions sont nécessaires (par exemple, plus de codes d'erreur sont possibles sous UPDATE que sous QUERY) et certains champs doivent être surchargés (voir la description des champs CLASSE ci-dessous).

Le format global d'un message UPDATE est le suivant [RFC1035] :



La section En-tête spécifie que ce message est une Mise à jour, et décrit la taille des autres sections. La section Zone désigne la zone qui va être mise à jour par ce message. La section Préalables spécifie les invariants de départ (en termes de contenu de zone) exigés pour cette mise à jour. La section Mise à jour contient les écritures à faire, et la section Données supplémentaires contient les données qui peuvent être nécessaires pour terminer, mais ne font pas partie de cette mise à jour.

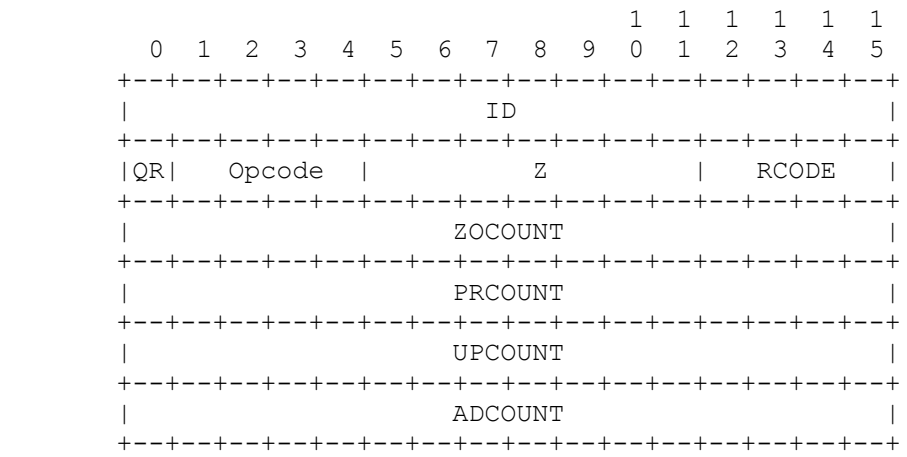
2.1 Questions de transport

Une transaction de mise à jour peut être portée dans un datagramme UDP, si la demande convient, ou dans une connexion TCP (à la discrétion du demandeur). Lorsque TCP est utilisé, le message est dans le format décrit au paragraphe 4.2.2 de la [RFC1035].

2.2 En-tête de message

L'en-tête du format de message DNS est défini au paragraphe 4.1 de la [RFC1035]. Tous les opcodes ne définissent pas les mêmes ensembles de bits de fanion, bien qu'en pratique, la plupart des bits définis pour QUERY (dans la [RFC1035]) soient identiquement définis par les autres opcodes. UPDATE utilise seulement un bit fanion (QR).

Le format de message DNS spécifie des comptes d'enregistrement pour ses quatre sections (Question, Réponse, Autorité, et Supplémentaires). UPDATE utilise les mêmes champs et les mêmes formats de sections, mais la dénomination et l'utilisation de ces sections diffèrent, comme le montre l'en-tête modifié suivant, d'après le paragraphe 4.1.1 de la [RFC1035] :



Ces champs sont utilisés comme suit :

ID Un identifiant de 16 bits alloué par l'entité qui génère toutes sortes de demandes. Cet identifiant est copié dans la réponse correspondante et peut être utilisé par le demandeur pour faire correspondre les réponses aux demandes en cours, ou par le serveur pour détecter les demandes dupliquées provenant d'un demandeur.

QR Un champ d'un bit qui spécifie si ce message est une demande (0), ou une réponse (1).

Opcode Un champ de quatre bits qui spécifie le type de demande du message. Cette valeur est réglée par l'origine d'une demande et est copiée dans la réponse. La valeur de Opcode qui identifie un message UPDATE est cinq (5).

Z Réservé pour utilisation future. Devrait être zéro (0) dans toutes les demandes et réponses. Un champ Z non à zéro devrait être ignoré par les mises en œuvre de cette spécification.

RCODE Code de réponse – ce champ de quatre bits est indéfini dans les demandes et établi dans les réponses. Les valeurs et significations de ce champ dans les réponses sont les suivantes :

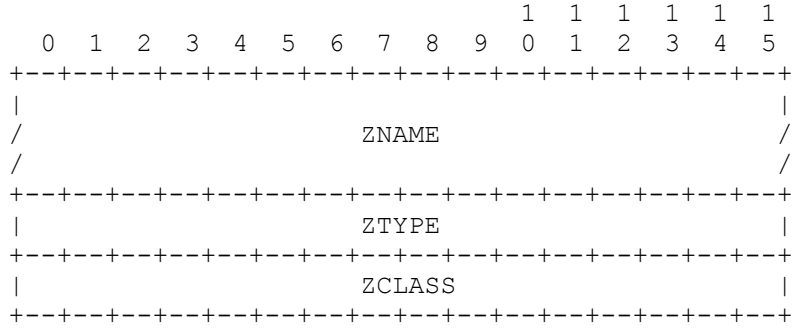
Mnémonique	**Valeur**	**Description**
NOERROR	0	Pas de condition d'erreur.
FORMERR	1	Le serveur de noms n'a pas été capable d'interpréter la demande, pour une erreur de format.
SERVFAIL	2	Le serveur de noms a rencontré une défaillance interne lors du traitement de cette demande, par exemple une erreur du système d'exploitation ou une fin de temporisation de transmission.
NXDOMAIN	3	Un nom qui devrait exister n'existe pas.
NOTIMP	4	Le serveur de noms ne prend pas en charge l'opcode spécifié.
REFUSED	5	Le serveur de noms refuse d'effectuer l'opération spécifiée pour des raisons de politique ou de sécurité.
YXDOMAIN	6	Un nom qui ne devrait pas exister existe.

YXRRSET	7	Un RRset qui ne devrait pas exister existe.
NXRRSET	8	Un RRset qui devrait exister n'existe pas.
NOTAUTH	9	Le serveur n'est pas d'autorité pour la zone désignée dans la section Zone.
NOTZONE	10	Un nom utilisé dans la section Préalables ou Mise à jour n'est pas dans la zone notée par la section Zone.

ZOCOUNT	Nombre de RR dans la section Zone.
PRCOUNT	Nombre de RR dans la section Préalables.
UPCOUNT	Nombre de RR dans la section Mise à jour.
ADCOUNT	Nombre de RR dans la section Données supplémentaires.

2.3 Section Zone

La section Zone a le même format que celui spécifié au paragraphe 4.1.2 de la [RFC1035], avec les champs redéfinis comme suit :



UPDATE utilise cette section pour désigner la zone de l'enregistrement qui est mise à jour. Tous les enregistrements à mettre à jour doivent être dans la même zone, et donc la section Zone a le droit de contenir exactement un enregistrement. Le ZNAME est le nom de la zone, le ZTYPE doit être SOA, et le ZCLASS est la classe de la zone.

2.4 Section Préalables

Cette section contient un ensemble de préalables RRset qui doivent être satisfaits au moment où le paquet UPDATE est reçu par le serveur maître principal. Le format de cette section est celui spécifié au paragraphe 4.1.3 de la [RFC1035]. Il y a cinq ensembles de sémantiques possibles qui peuvent être exprimés ici, résumés comme suit et expliqués ensuite.

(1) RRset existe (indépendamment de la valeur). Au moins un RR avec un NOM et TYPE spécifiés (dans la zone et la classe spécifiées par la section Zone) doit exister.

(2) RRset existe (selon la valeur). Un ensemble de RR avec un NOM et TYPE spécifiés existe et a les mêmes membres avec les mêmes RDATA que le RRset spécifié ici dans cette section.

(3) RRset n'existe pas. Aucun RR avec les NOM et TYPE spécifiés (dans la zone et la classe notées par la section Zone) ne peut exister.

(4) Le nom est utilisé. Au moins un RR avec un NOM spécifié (dans la zone et la classe spécifiées par la section Zone) doit exister. Noter que ce préalable N'EST PAS satisfait par des non terminaux vides.

(5) Le nom n'est pas utilisé. Aucun RR d'aucun type n'est possédé par un NOM spécifié. Noter que ce préalable EST satisfait par des non terminaux vides.

Leur syntaxe est la suivante:

2.4.1 RRset existe (indépendamment de la valeur)
Au moins un RR avec un NOM et un TYPE spécifiés (dans la zone et la classe spécifiées dans la section Zone) doit exister.

Pour ce préalable, un demandeur ajoute à la section un seul RR dont le NOM et le TYPE sont égaux à ceux du RRset de la zone dont l'existence est exigée. RDLENGTH est zéro et RDATA est donc vide. CLASSE doit être spécifiée

comme TOUT pour différencier cette condition de celle d'un RR réel dont RDLENGTH est naturellement zéro (0) (par exemple, NUL). TTL est spécifié comme zéro (0).

2.4.2 RRset existe (selon la valeur)
Un ensemble de RR avec un NOM et un TYPE spécifiés existe et a les mêmes membres avec les mêmes RDATA que le RRset spécifié ici dans cette section. Bien que l'ordre du RRset soit indéfini et donc non significatif pour cette comparaison, les ensembles sont identiques dans leur développement.

Pour ce préalable, un demandeur ajoute à la section un RRset entier dont la préexistence est exigée. Le NOM et le TYPE sont ceux du RRset noté. CLASSE est celle de la zone. TTL doit être spécifié comme zéro (0) et est ignoré dans la comparaison d'identité des RRset.

2.4.3 RRset n'existe pas
Aucun RR avec un NOM et un TYPE spécifiés (dans la zone et la classe notées par la section Zone) ne peut exister.

Pour ce préalable, un demandeur ajoute à la section un seul RR dont le NOM et le TYPE sont égaux à ceux du RRset dont la non existence est exigée. Le RDLENGTH de cet enregistrement est zéro (0), et le champ RDATA est donc vide. CLASSE doit être spécifiée comme AUCUNE afin de distinguer cette condition d'un RR valide dont RDLENGTH est naturellement zéro (0) (par exemple, le RR NUL). TTL doit être spécifié comme zéro (0).

2.4.4 Le nom est utilisé
Le nom est utilisé. Au moins un RR avec un NOM spécifié (dans la zone et la classe spécifiées par la section Zone) doit exister. Noter que ce préalable N'EST PAS satisfait par des non terminaux vides. Pour ce préalable, un demandeur ajoute à la section un seul RR dont le NOM est égal à celui du nom dont la possession d'un RR est exigée. RDLENGTH est zéro et RDATA est donc vide. CLASSE doit être spécifiée comme TOUT pour différencier cette condition de celle d'un RR réel dont RDLENGTH est naturellement zéro (0) (par exemple, NUL). TYPE doit être spécifié comme TOUT pour différencier ce cas de celui d'un essai d'existence d'un RRset. TTL est spécifié comme zéro (0).

2.4.5 Le nom n'est pas utilisé
Le nom n'est pas utilisé. Aucun RR d'aucun type n'est possédé par un NOM spécifié. Noter que ce préalable EST satisfait par des non terminaux vides.

Pour ce préalable, un demandeur ajoute à la section un seul RR dont le NOM est égal à celui du nom dont la non possession d'un RR est exigée. RDLENGTH est zéro et RDATA est donc vide. CLASSE doit être spécifiée comme AUCUNE. TYPE doit être spécifié comme TOUT. TTL doit être spécifié comme zéro (0).

2.5 Section UPDATE

Cette section contient des RR à ajouter ou supprimer de la zone. Le format de cette section est celui spécifié par le paragraphe 4.1.3 de la [RFC1035]. Il y a quatre ensembles de sémantiques possibles, résumés ci-dessous et détaillés ensuite.

(1) Ajouter des RR à un RRset.
(2) Supprimer un RRset.
(3) Supprimer tous les RRset d'un nom.
(4) Supprimer un RR d'un RRset.

Leur syntaxe est la suivante:

2.5.1 Ajouter à un RRset
Les RR sont ajoutés à la section UPDATE dont le NOM, TYPE, TTL, RDLENGTH et RDATA sont ceux qui sont ajoutés, et dont CLASSE est la même que la classe de la zone. Tout RR dupliqué sera ignoré en silence par le maître principal.

2.5.2 Supprimer un RRset
Un RR est ajouté à la section UPDATE dont le NOM et le TYPE sont ceux du RRset à supprimer. TTL doit être spécifié comme zéro (0) et n'est pas autrement utilisé par le maître principal. CLASSE doit être spécifiée comme TOUT. RDLENGTH doit être zéro (0) et RDATA doit donc être vide. Si il n'existe pas de tel RRset, ce RR UPDATE sera alors ignoré en silence par le maître principal.

2.5.3 Supprimer tous les RRset d'un nom
Un RR est ajouté à la section UPDATE dont le NOM est celui du nom à épurer des RRset. Le TYPE doit être spécifié comme TOUT. Le TTL doit être spécifié comme zéro (0) et n'est autrement pas utilisé par le maître principal. CLASSE doit être spécifiée comme TOUT. RDLENGTH doit être zéro (0) et RDATA doit donc être vide. Si il n'existe pas de tel RRset, ce RR UPDATE sera ignoré en silence par le maître principal.

2.5.4 Supprimer un RR d'un RRset
Les RR à supprimer sont ajoutés à la section UPDATE. Les NOM, TYPE, RDLENGTH et RDATA doivent correspondre au RR à supprimer. Le TTL doit être spécifié comme zéro (0) et sera autrement ignoré par le maître principal. CLASSE doit être spécifiée comme AUCUNE pour distinguer cela d'un ajout de RR. Si il n'existe pas de tels RR, ce RR UPDATE sera ignoré en silence par le maître principal.

2.6 Section Données supplémentaires

Cette section contient des RR qui se rapportent à la mise à jour elle-même, ou aux nouveaux RR qui sont ajoutés par la mise à jour. Par exemple, ceux à partir de la zone glu (des RR auxquels se réfèrent les nouveaux RR NS) devraient être présentés ici. Le serveur peut utiliser ou ignorer ce qui sort de la zone glu, à la discrétion de la mise en œuvre de serveur. Le format de cette section est celui spécifié par le paragraphe 4.1.3 de la [RFC1035].

3 Comportement du serveur

Un serveur, à réception d'une demande UPDATE, va signaler NOTIMP au demandeur si l'opcode UPDATE n'est pas reconnu ou si, étant reconnu, il n'a pas été mis en œuvre. Autrement, le traitement se continue comme suit.

3.1 Traitement de la section Zone

3.1.1 La section Zone est vérifiée pour voir si il y a exactement un RR dedans et si le ZTYPE de ce RR est SOA, autrement, on signale FORMERR au demandeur. Ensuite, le ZNAME et la ZCLASS sont vérifiés pour voir si la zone ainsi nommée est une des zones d'autorité de ce serveur, autrement, on signale NOTAUTH au demandeur. Si le serveur est un esclave de la zone, la demande sera retransmise au maître principal.

3.1.2 Pseudo code pour le traitement de la section Zone

```
si (zcount != 1 || ztype != SOA)
    retourner (FORMERR)
si (zone_type(zname, zclass) == SLAVE)
    retourner forward()
si (zone_type(zname, zclass) == MASTER)
    retourner update()
retourner (NOTAUTH)
```

Les paragraphes 3.2 à 3.8 décrivent le comportement du maître principal, tandis que la Section 6 décrit le comportement d'un transmetteur.

3.2 Traitement de la section Préalables

Ensuite, on vérifie la section Préalables pour voir si tous les préalables sont satisfaits par l'état actuel de la zone. En utilisant les définitions du paragraphe 1.2, si un NOM d'un RR n'est pas dans la zone spécifiée dans la section Zone, on signale NOTZONE au demandeur.

3.2.1 Pour les RR de cette section dont CLASSE est TOUT, vérifier que TTL et RDLENGTH sont tous deux à zéro (0), autrement signaler FORMERR au demandeur. Si TYPE est TOUT, vérifier qu'il y a au moins un RR dans la zone dont NOM est le même que celui du RR Préalables, autrement signaler NXDOMAIN au demandeur. Si TYPE n'est pas TOUT, vérifier qu'il y a au moins un RR dans la zone dont les NOM et TYPE sont les mêmes que ceux du RR Préalables, autrement, signaler NXRRSET au demandeur.

3.2.2 Pour les RR de cette section dont CLASSE est AUCUNE, vérifier que TTL et RDLENGTH sont tous deux à zéro

(0), autrement signaler FORMERR au demandeur. Si le TYPE est TOUT, vérifier qu'il n'y a pas dans la zone de RR dont le NOM est le même que celui du RR Préalables, autrement signaler YXDOMAIN au demandeur. Si le TYPE n'est pas TOUT, vérifier qu'il n'y a pas dans la zone de RR dont NOM et TYPE sont les mêmes que ceux du RR Préalables, autrement signaler YXRRSET au demandeur.

3.2.3 Pour les RR de cette section dont CLASSE est la même que la ZCLASS, vérifier que le TTL est zéro (0), autrement, signaler FORMERR au demandeur. Puis, construire un RRset pour chaque <NOM, TYPE> unique et comparer chaque RRset résultant (mêmes membres, pas plus, pas moins) aux RRset de la zone. Si un RRset Préalables n'est pas entièrement et exactement en correspondance avec un RRset de la zone, signaler NXRRSET au demandeur. Si un RR de cette section a une CLASSE autre que ZCLASS ou AUCUNE ou TOUT, signaler FORMERR au demandeur.

3.2.4 Tableau des méta valeurs utilisées dans la section Préalables

CLASSE	**TYPE**	**RDATA**	**Signification**
TOUT	TOUT	vide	Le nom est utilisé
TOUT	rrset	vide	Le RRset existe (indépendamment de la valeur)
AUCUNE	TOUT	vide	Le nom n'est pas utilisé
AUCUNE	rrset	vide	Le RRset n'existe pas
Zone	rrset	rr	Le RRset existe (selon la valeur)

3.2.5 Pseudo code pour le traitement de la section Préalables

```
   pour un rr dans les préalables
      si (rr.ttl != 0)
retourner (FORMERR)
      si (zone_of(rr.name) != ZNAME)
retourner (NOTZONE);
      si (rr.class == TOUT)
         si (rr.rdlength != 0)
               retourner (FORMERR)
         si (rr.type == TOUT)
               si (!zone_name<rr.name>)
                     retourner (NXDOMAIN)
         autrement
               si (!zone_rrset<rr.name, rr.type>)
                     retourner (NXRRSET)
      si (rr.class == AUCUNE)
         si (rr.rdlength != 0)
               retourner (FORMERR)
         si (rr.type == TOUT)
               si (zone_name<rr.name>)
                     retourner (YXDOMAIN)
         autrement
               si (zone_rrset<rr.name, rr.type>)
                     retourner (YXRRSET)
      si (rr.class == zclass)
         temp<rr.name, rr.type> += rr
      autrement
         retourner (FORMERR)

   pour un rrset dans temp
      si (zone_rrset<rrset.name, rrset.type> != rrset)
         retourner (NXRRSET)
```

3.3 Vérification des permissions du demandeur

3.3.1 On peut vérifier ensuite que le demandeur a la permission de mettre à jour les RR désignés dans la section UPDATE, d'une façon propre à la mise en œuvre ou en utilisant un mécanisme qui sera spécifié dans un protocole sécurisé de mise à jour du DNS à venir. Si le demandeur n'a pas la permission d'effectuer ces mises à jour, le serveur peut écrire un message d'avertissement dans ses journaux de fonctionnement, et peut soit signaler REFUSÉ au demandeur, soit ignorer le problème de la permission et poursuivre la mise à jour.

3.3.2 Bien que le processus exact soit défini par la mise en œuvre, si ces activités de vérification doivent être effectuées, c'est le moment dans le traitement du serveur où de telles performances devraient avoir lieu, car, si une condition REFUSÉ est rencontrée après qu'une mise à jour a été partiellement appliquée, il sera nécessaire de défaire la mise à jour partielle et de restaurer la zone dans son état d'origine avant de répondre au demandeur.

3.3.3 Pseudo code pour la vérification de permission

```
si (une politique de sécurité existe)
  si (cette mise à jour n'est pas permise)
      si (option locale)
          enregistrer un message sur le problème de permission
      si (option locale)
          retourner (REFUSÉ)
```

3.4 Traitement de la section MISE À JOUR

Puis, la section MISE À JOUR est traitée comme suit.

3.4.1 Pré examen
La section MISE À JOUR est analysée par RR et chaque CLASSE de RR est vérifiée pour voir si elle est TOUT, AUCUNE, ou la même que la CLASSE de la zone, autrement, on signale une FORMERR au demandeur. En utilisant les définitions du paragraphe 1.2, chaque NOM de RR doit être dans la zone spécifiée dans la section Zone, autrement, signaler NOTZONE au demandeur.

3.4.1.2 Pour les RR dont CLASSE n'est pas TOUT, vérifier le TYPE et si il est TOUT, AXFR, MAILA, MAILB, ou tout autre méta type que QUERY, ou de tout type non reconnu, signaler FORMERR au demandeur. Pour les RR dont CLASSE est TOUT ou AUCUNE, vérifier que le TTL est zéro (0), autrement, signaler FORMERR au demandeur. Pour tout RR dont la CLASSE est TOUT, vérifier le RDLENGTH pour s'assurer qu'elle est zéro (0) (c'est-à-dire que le champ RDATA est vide) et que le TYPE n'est pas AXFR, MAILA, MAILB, ou tout autre méta type de QUERY en-dehors de TOUT, ou d'un type non reconnu, autrement signaler FORMERR au demandeur.

3.4.1.3 Pseudo code pour le pré examen de la section MISE À JOUR

```
[rr] pour rr dans les mises à jour
    si (zone_of(rr.name) != ZNAME)
        retourner (NOTZONE);
    si (rr.class == zclass)
        si (rr.type & TOUT|AXFR|MAILA|MAILB)
            retourner (FORMERR)
    autrement si (rr.class == TOUT)
        si (rr.ttl != 0 || rr.rdlength != 0
            || rr.type & AXFR|MAILA|MAILB)
            retourner (FORMERR)
    autrement si (rr.class == AUCUNE)
        si (rr.ttl != 0 || rr.type & TOUT|AXFR|MAILA|MAILB)
            retourner (FORMERR)
    autrement
        retourner (FORMERR)
```

3.4.2 Mise à jour
La section Mise à jour est analysée par RR et ces RR sont traités dans l'ordre.

3.4.2.1 Si il survient une défaillance système (comme une condition de manque de mémoire, ou une erreur de matériel dans une mémoire persistante) durant le traitement de cette section, signaler SERVFAIL au demandeur et défaire toutes les mises à jour appliquées à la zone durant cette transaction.

3.4.2.2 Tout RR Mise à jour dont la CLASSE est la même que ZCLASS est ajouté à la zone. En cas de RDATA dupliqués (ce qui pour les RR SOA est toujours le cas, et pour les RR WKS est le cas si les champs ADRESSE et PROTOCOLE correspondent tous deux) le RR Zone est remplacé par le RR Update. Si le TYPE est SOA et si il n'y a pas de RR SOA de zone, ou si le nouveau SOA.SERIAL est inférieur (conformément à la [RFC1982]) ou égal au SOA.SERIAL du RR SOA de zone actuel, le RR Mise à jour est ignoré. Dans le cas d'un RR Mise à jour

CNAME et d'un RR de zone non CNAME, ou vice versa, ignorer le RR Mise à jour CNAME, autrement, remplacer le RR CNAME de zone par la mise à jour du RR CNAME.

3.4.2.3 Pour tout RR Mise à jour dont la CLASSE est TOUT et dont le TYPE est TOUT, tous les RR Zone avec le même NOM sont supprimés, sauf si le NOM est le même que ZNAME auquel cas seuls les RR dont le TYPE est autre que SOA ou NS sont supprimés. Pour tout RR Mise à jour dont la CLASSE est TOUT et dont le TYPE n'est pas TOUT, tous les RR Zone avec le même NOM et TYPE sont supprimés, sauf si le NOM est le même que le ZNAME auquel cas ni les RR SOA ni les RR NS ne seront supprimés.

3.4.2.4 Pour tout RR Mise à jour dont la classe est AUCUNE, tout RR Zone dont NOM, TYPE, RDATA et RDLENGTH sont égaux au RR Mise à jour est supprimé, sauf si le NOM est le même que le ZNAME et si soit le TYPE est SOA, soit le TYPE est NS, et si le RR Zone correspondant est le seul NS restant dans le RRset, auquel cas ce RR Mise à jour est ignoré.

3.4.2.5 Signaler NOERROR au demandeur.

3.4.2.6 Tableau des Méta valeurs utilisées dans la section Mise à jour

CLASSE	TYPE	RDATA	Signification
TOUT	TOUT	vide	Supprime tous les RRset d'un nom
TOUT	rrset	vide	Supprime un RRset
AUCUNE	rrset	rr	Supprime un RR d'un RRset
zone	rrset	rr	Ajoute à un RRset

3.4.2.7 Pseudocode pour le traitement de la section Mise à jour

```
[rr] pour rr dans updates
        si (rr.class == zclass)
            si (rr.type == CNAME)
                si (zone_rrset<rr.name, ~CNAME>)
                    prochain [rr]
            autrement si (zone_rrset<rr.name, CNAME>)
                prochain [rr]
            si (rr.type == SOA)
                si (!zone_rrset<rr.name, SOA> ||
                    zone_rr<rr.name, SOA>.serial > rr.soa.serial)
                    prochain [rr]
            pour zrr dans zone_rrset<rr.name, rr.type>
                si (rr.type == CNAME || rr.type == SOA ||
                    (rr.type == WKS && rr.proto == zrr.proto &&
                    rr.address == zrr.address) ||
                    rr.rdata == zrr.rdata)
                    zrr = rr
                    prochain [rr]
            zone_rrset<rr.name, rr.type> += rr
        autrement si (rr.class == TOUT)
            si (rr.type == TOUT)
                si (rr.name == zname)
                    zone_rrset<rr.name, ~(SOA|NS)> = Nil
                autrement
                    zone_rrset<rr.name, *> = Nil
            autrement si (rr.name == zname &&
                    (rr.type == SOA || rr.type == NS))
                prochain [rr]
            autrement
                    zone_rrset<rr.name, rr.type> = Nil
        autrement si (rr.class == AUCUNE)
            si (rr.type == SOA)
                prochain [rr]
            si (rr.type == NS && zone_rrset<rr.name, NS> == rr)
                prochain [rr]
                    zone_rr<rr.name, rr.type, rr.data> = Nil
    retourner (NOERROR)
```

3.5 Stabilité

Lorsque une zone est modifiée par une opération UPDATE, le serveur doit confier le changement à une mémoire non volatile avant d'envoyer une réponse au demandeur ou de répondre à toute interrogation ou transfert pour la zone modifiée. Il est raisonnable qu'un serveur ne mémorise que les enregistrements mis à jour dans la mesure où un redémarrage du système ou une panne d'alimentation causera l'incorporation de ces enregistrements dans la zone au prochain démarrage du serveur. Il est aussi raisonnable que le serveur copie la totalité de la zone modifiée sur une mémoire non volatile après chaque opération de mise à jour, bien que cela puisse obérer les performances lorsqu'il s'agit de grandes zones.

3.6 Zone Identité

Si le SOA SERIAL de la zone est changé par une opération de mise à jour, ce changement doit être dans une direction positive (en utilisant l'arithmétique modulo 2**32 spécifiée dans la [RFC1982]). Les tentatives de remplacer un SOA par un dont le SERIAL serait inférieur à celui en cours seront ignorées en silence par le serveur maître principal.

Si le SERIAL du SOA de la zone n'est pas changé par suite d'une opération de mise à jour, le serveur devra alors l'incrémenter automatiquement avant que le SOA ou tout nom modifié ou RR ou RRset soit inclus dans une réponse ou transfert. La mise en œuvre de serveur maître principal peut choisir d'incrémenter automatiquement le SOA SERIAL si un des événements suivants survient :

(1) à chaque opération de mise à jour;

(2) si un nom, un RR ou RRset a changé dans la zone et a ensuite été visible à un client du DNS depuis que le SOA non incrémenté a été visible à un client du DNS, et si le SOA est sur le point de devenir visible à un client du DNS.

(3) si une période de temps configurable s'est écoulée depuis la dernière opération de mise à jour. Cette période devra être inférieure ou égale à un tiers du temps de rafraîchissement de la zone, et sa durée par défaut devra être le plus petit de ce maximum et de 300 secondes.

(4) un nombre configurable de mises à jour a été appliqué depuis le dernier changement de SOA. La valeur par défaut pour ce paramètre de configuration devra être de cent (100).

Il est impératif que le contenu de la zone et le SERIAL du SOA soient étroitement synchronisés. Si il advient que la zone change, le SOA doit changer aussi.

3.7 Atomicité

Durant le traitement d'une transaction UPDATE, le serveur doit s'assurer de l'atomicité par rapport aux autres transactions (concurrentes) UPDATE ou QUERY. Deux de ces transactions ne peuvent être traitées concurremment si l'une ou l'autre dépend du résultat final de l'autre ; en particulier, une transaction QUERY ne devrait pas pouvoir restituer des RRset qui ont été partiellement modifiés par une transaction UPDATE concurrente, et une UPDATE ne devrait pas pouvoir commencer à partir de préalables qui pourraient ne plus tenir à la fin d'une autre opération UPDATE concurrente. Finalement, si deux transactions UPDATE doivent modifier les mêmes noms, RR ou RRset, de telles transactions UPDATE devront être traitées l'une après l'autre.

3.8 Réponse

À la fin d'un traitement UPDATE, un code de réponse sera connu. Un message de réponse est généré en copiant les champs Identifiant et Opcode de la demande, et en copiant les champs ZOCOUNT, PRCOUNT, UPCOUNT, et ADCOUNT et leurs sections associées, ou en plaçant des zéros (0) dans ces champs de "compte" et en n'incluant aucune partie de la mise à jour originale. Le bit QR est mis à un (1), et la réponse est renvoyée au demandeur. Si le demandeur a utilisé UDP, la réponse sera alors envoyée à l'accès UDP de source du demandeur. Si le demandeur a utilisé TCP, la réponse sera alors renvoyée sur la connexion TCP ouverte par le demandeur.

4. Comportement du demandeur

4.1 Du point de vue d'un demandeur, tout serveur d'autorité pour la zone peut paraître capable de traiter les demandes de mise à jour, bien que seul le serveur maître principal soit réellement capable de modifier le fichier maître de la zone. Les demandeurs sont supposés connaître le nom de la zone qu'ils ont l'intention de mettre à jour et de connaître ou d'être capable de déterminer les serveurs de noms de cette zone.

4.2 Si la mise en ordre des mises à jour est désirée, le demandeur aura besoin de savoir la valeur des RR SOA existants. Les demandeurs qui mettent à jour le RR SOA doivent mettre à jour le champ SERIAL SOA dans une direction positive (comme défini par la [RFC1982]) et aussi préserver les autres champs SOA sauf si le demandeur a l'intention explicite de les changer. Le champ SERIAL SOA ne doit jamais être mis à zéro (0).

4.3 Si le demandeur a de bonnes raisons de penser que tous les serveurs d'une zone seront également accessibles, il devrait alors s'arranger pour essayer d'abord le serveur maître principal (comme indiqué par le champ MNAME du SOA si il correspond à un NSDNAME de NS) pour éviter des transmissions inutiles avec les serveurs esclaves. (Noter que le serveur maître principal ne sera dans certains cas pas accessible par tous les demandeurs, à cause de pare-feu ou de partitions du réseau.)

4.4 Une fois que les serveurs de noms de la zone ont été trouvés et éventuellement triés de sorte que ceux qui sont le plus vraisemblablement accessibles et/ou qui prennent en charge l'opcode UPDATE soient les premiers de la liste, le demandeur compose un message UPDATE de la forme suivante, et l'envoie au premier serveur de noms de sa liste :

ID :	(nouveau)
Opcode :	UPDATE
zcount de zone :	1
zname de zone :	(nom de zone)
zclass de zone :	(classe de zone)
ztype de zone :	T_SOA
Section Préalables:	(voir le texte précédent)
Section Mise à jour:	(voir le texte précédent)
Section Données supplémentaires :	(vide)

4.5 Si le demandeur reçoit une réponse, et si la réponse a un RCODE autre que SERVFAIL ou NOTIMP, le demandeur retourne alors une réponse appropriée à celui qui l'a appelé.

4.6 Si une réponse est reçue dont le RCODE est SERVFAIL ou NOTIMP, ou si aucune réponse n'est reçue pendant une période de temporisation qui dépend de la mise en œuvre, ou si une erreur ICMP est reçue, indiquant que l'accès du serveur n'est pas joignable, le demandeur va alors supprimer le serveur inutilisable de sa liste interne de serveurs de noms et essayer le suivant, répétant jusqu'à ce que la liste des serveurs de noms soit vide. Si le demandeur est à court de serveurs à essayer, une erreur appropriée sera retournée au correspondant du demandeur.

5 Détection des duplicata, ordre et exclusion mutuelle

5.1 Pour un fonctionnement correct, des mécanismes peuvent être nécessaires pour assurer l'idempotence, l'ordre des demandes UPDATE et l'exclusion mutuelle. Un message ou une réponse UPDATE peut être délivré zéro fois, une fois, ou plusieurs fois. La duplication des datagrammes est d'un intérêt particulier car elle couvre le cas de ce qu'on appelle "l'attaque en répétition" où une demande correcte est dupliquée de façon malveillante par un intrus.

5.2 Plusieurs demandes ou réponses UPDATE en transit peuvent être délivrées dans n'importe quel ordre, du fait de changement de la topologie du réseau ou d'un équilibrage de charge, ou de graphes de transmission à plusieurs chemins dans lesquels plusieurs serveurs esclaves transmettent tous au maître principal. Dans certains cas, il peut être exigé que la mise à jour la plus ancienne ne soit pas appliquée après la dernière mise à joue, où "plus ancienne" et "dernière" sont définis par une base de temps externe visible d'un certain ensemble de demandeurs, plutôt que par l'ordre de réception des demandes par le maître principal.

5.3 Un demandeur peut assurer l'idempotence des transactions en supprimant explicitement certains "RR marqueurs" (plutôt que de supprimer le RRset dont il fait partie) et en ajoutant alors un nouveau "RR marqueur "avec un champ RDATA différent. La section Préalables devrait spécifier que le "RR marqueur " d'origine doit être présent afin que ce message UPDATE soit accepté par le serveur.

5.4 Si la demande est dupliquée par une erreur du réseau, toutes les demandes dupliquées vont échouer car seule la première va trouver présent le "RR marqueur" original qui a sa propre valeur précédente connue. Les décisions sur le fait d'utiliser un tel "RR marqueur " et quel RR utiliser appartiennent au programmeur d'application, bien qu'un choix évident soit le RR SOA de la zone comme décrit ci-dessous.

5.5 Les demandeurs peuvent s'assurer de l'ordre de mise à jour en synchronisant en externe leur utilisation des valeurs successives du "RR marqueur." L'exclusion mutuelle peut être traitée comme un cas dérivé, en ce qu'une seule succession du "RR marqueur " est tout ce qui est nécessaire.

5.6 Un cas particulier où l'ordre de mise à jour et la duplication de datagramme se recoupent est quand la validité d'un RR se change en une nouvelle valeur puis revient à la valeur précédente. Sans un "RR marqueur " tel que décrit ci-dessus, cette séquence de mises à jour peut laisser la zone dans un état indéfini si les datagrammes sont dupliqués.

5.7 Pour achever un cycle atomique multi transaction "lecture-modification-écriture", un demandeur pourrait d'abord restituer le RR SOA, et construire un message UPDATE dont un des préalables serait l'ancien RR SOA. Il spécifierait alors des mises à jour qui supprimeraient le RR SOA et en ajouterait un nouveau avec un SOA SERIAL incrémenté, ainsi qu'avec tous les préalables et toutes les mises à jour réelles qui seraient l'objet de la transaction. Si la transaction réussit, le demandeur sait que les RR qui ont été changés n'ont pas été autrement altérés par un autre demandeur.

6. Transmission

Quand un esclave de zone transmet un message UPDATE en amont vers le serveur maître principal de la zone, il doit allouer un nouvel identifiant et se préparer à jouer le rôle de "serveur transmetteur", qui est celui d'un demandeur par rapport au serveur de transmission.

6.1 L'ensemble des serveurs de transmission sera le même que l'ensemble de serveurs que cet esclave de zone utiliserait comme source des données AXFR ou IXFR. Aussi, alors que le demandeur d'origine peut avoir utilisé le RRset NS de la zone pour localiser son serveur de mise à jour, un transmetteur transmet toujours vers ses serveurs maîtres de zone désignés.

6.2 Si le demandeur d'origine a utilisé TCP, la connexion TCP provenant du demandeur est encore ouverte et le transmetteur doit utiliser TCP pour la transmission du message. Si le demandeur d'origine a utilisé UDP, le transmetteur peut utiliser UDP ou TCP pour la transmission du message, au gré de la mise en œuvre.

6.3 Il est raisonnable pour les serveurs de transmission d'être eux-mêmes des transmetteurs, si le graphe de dépendance AXFR qui est suivi est profond et implique des pare-feu et plusieurs domaines de connexité. Dans la plupart des cas, le graphe de dépendance AXFR sera peu profond et le serveur de transmission sera le serveur maître principal.

6.4 Le transmetteur ne répondra pas à son demandeur avant d'avoir reçu une réponse de son serveur de transmission. Les transactions UPDATE qui impliquent des transmetteurs sont donc synchronisées par rapport au demandeur original et au serveur maître principal.

6.5 Lorsque il y a plusieurs sources possibles de données AXFR et donc plusieurs serveurs de transmission possibles, un transmetteur va utiliser la même stratégie de repli par rapport à la connexité ou aux erreurs de temporisation que celle qu'il utiliserait en effectuant un AXFR. Cela dépend de la mise en œuvre.

6.6 Lorsque un transmetteur reçoit une réponse d'un serveur transmetteur, il copie cette réponse dans un nouveau message de réponse, alloue son identifiant de demandeur à ce message, et renvoie la réponse au demandeur.

7. Notes sur le concept, la mise en œuvre, le fonctionnement, et le protocole

Certains des principes qui ont guidé la conception de cette spécification UPDATE sont les suivants. Noter qu'ils ne font pas partie de la spécification formelle et que tout désaccord entre cette section et toute autre du présent document devrait se résoudre en faveur de cette autre section.

7.1 L'utilisation de méta valeurs pour CLASS n'est possible que parce que tous les RR dans le paquet sont supposés être dans la même zone, et CLASSE est un attribut d'une zone plutôt que d'un RRset. (C'est pour cette raison que la section Zone n'est pas facultative.)

7.2 Comme il n'y a pas d'erreur "données présentes" ou "données absentes" possibles à partir du traitement de la section Mise à jour, toute dépendance nécessaire à la présence et l'absence des données devrait être spécifiée dans la section Préalables.

7.3 La section Données supplémentaires peut être utilisée pour fournir à un serveur des données glu hors zone qui seront nécessaires dans les références. Par exemple, si l'ajout d'un nouveau RR NS à HOME.VIX.COM spécifiant un serveur de noms appelé NS.AU.OZ, le RR A pour NS.AU.OZ peut être inclus dans la section Données supplémentaires. Les serveurs peuvent utiliser ces informations ou les ignorer, à la discrétion de la mise en œuvre.

Nous déconseillons la mise en antémémoire de ces informations pour une utilisation dans des réponses DNS ultérieures.

7.4 La section Données supplémentaires pourrait être utilisée si certains des RR nécessaires ultérieurement pour la mise à jour sécurisée du DNS ne sont pas réellement des mises à jour de zone, mais plutôt des clés auxiliaires ou des signatures non destinées à être mémorisées dans la zone (comme le serait une mise à jour) bien que nécessaires pour valider l'opération de mise à jour.

7.5 Il est prévu qu'en l'absence de la mise à jour sécurisée du DNS, un serveur n'accepte de mises à jour que si elles viennent d'une adresse de source qui a été configurée de façon statique dans la description du serveur d'une zone maîtresse principale. Les serveurs DHCP seraient vraisemblablement candidats à l'inclusion dans cette liste configurée de façon statique.

7.6 Il n'est pas possible de créer une zone en utilisant ce protocole, car il n'y a aucune disposition qui puisse dire à un serveur esclave qui sont ses serveurs maîtres. Il est prévu que ce protocole soit étendu à l'avenir pour couvrir ce cas. Donc, pour l'instant, l'ajout de RR SOA n'est pas pris en charge. Pour des raisons similaires, la suppression des RR SOA n'est elle non plus, pas prise en charge.

7.7 Le préalable pour spécifier qu'un nom possède au moins un RR diffère sémantiquement de QUERY, en ce que QUERY retournerait <NOERROR,ANCOUNT=0> plutôt que NXDOMAIN si il était interrogé sur un RRset à ce nom, alors que la condition préalable de UPDATE [paragraphe 2.4.4] NE serait PAS satisfaite.

7.8 Il est possible qu'une réponse UDP soit perdue dans le transit et qu'une demande soit réessayée du fait d'une condition de fin de temporisation. Dans ce cas, une UPDATE qui réussirait la première fois qu'elle a été reçue par le maître principal pourrait se révéler avoir échoué en fin de compte lorsque la réponse à une demande dupliquée est finalement reçue par le demandeur. (Cela parce que le préalable original peut ne plus être satisfait après l'application de la mise à jour.) Pour cette raison, les demandeurs qui exigent un code de réponse précis doivent utiliser TCP.

7.9 Parce qu'un demandeur qui exige un code de réponse précis va initier sa transaction UPDATE avec TCP, un transmetteur qui reçoit une demande via TCP doit la transmettre avec TCP.

7.10 Il est possible de différer l'auto incrémentation des SOA SERIAL afin que les numéros de série puissent être conservés et le retour à zéro à 2**32 peut être rendu très peu fréquent. Les SOA SERIAL visibles (aux clients DNS) doivent différer si la zone diffère. Noter que la SOA de section Autorité dans une réponse QUERY est une forme de visibilité, pour les besoins de ce préalable.

7.11 Un SOA SERIAL de zone ne devrait jamais être réglé à zéro (0) du fait de problèmes d'interopérabilité avec certaines mises en œuvre anciennes mais largement répandues du DNS. Lorsque on incrémente un SOA SERIAL, si le résultat de l'incrément est zéro (0) (comme ce sera le cas lors d'un retour à zéro après 2**32) il est nécessaire de l'incrémenter à nouveau ou de le régler à un (1). Voir la [RFC1982] pour des précisions sur ce sujet.

7.12 Du fait de la minimisation du TTL nécessaire lors de la mise en antémémoire d'un RRset, il est recommandé que tous les TTL d'un RRset soient réglés à la même valeur. Bien que le format de message du DNS permette que des TTL variés existent dans le même RRset, et que cette variation puisse exister à l'intérieur d'une zone, une telle variation va avoir des résultats contraires à l'intuition et son utilisation est déconseillée.

7.13 La gestion des coupures de zone présente des aspects obscurs aux opérations d'ajout et suppression dans la section Mise à jour. Il est possible de supprimer un RR NS pour autant qu'il n'est pas le dernier RR NS à la racine d'une zone. Si on supprime tous les RR d'un nom, le SOA et les RR NS à la racine d'une zone ne sont pas affectés. Si on supprime les RRset, il n'est pas possible de supprimer le SOA ou les RRset NS au sommet d'une zone. Une tentative d'ajout d'une SOA sera traitée comme une opération de remplacement si une SOA existe déjà, ou a un no-op si la SOA est nouvelle.

7.14 Aucune vérification sémantique n'est exigée chez le serveur maître principal lorsque il ajoute de nouveaux RR. Donc un demandeur peut causer l'ajout d'un CNAME ou d'un NS ou de toute autre sorte de RR même si leur nom cible n'existe pas ou n'a pas les RRset appropriés pour rendre utile le RR original. Les serveurs maîtres principaux qui mettent effectivement en œuvre cette sorte de vérification devraient faire très attention à éviter des dépendances hors zone (dont la véracité ne peut pas être vérifiée avec autorité) et devraient mettre en œuvre toutes ces vérifications durant la phase de pré-examen.

7.15 Les CNAME non terminaux ou à caractère générique *(wildcard)* ne sont pas bien spécifiés dans la [RFC1035] et leur utilisation va probablement conduire à des résultats imprévisibles. Leur utilisation est déconseillée.

7.16 Les non terminaux vides (nœuds avec des enfants mais pas de RR en propre) causent l'envoi de réponses <NOERROR,ANCOUNT=0> à une interrogation de tout type pour ce nom. Il n'y a pas de dispositions sur les nœuds terminaux vides – aussi si tous les RR d'un nœud terminal sont supprimés, le nom n'est plus utilisé, et les interrogations de tout type pour ce nom vont résulter en une réponse NXDOMAIN.

7.17 Dans un graphe de dépendance AXFR profond, ce n'était pas une erreur historique que les esclaves dépendent mutuellement les uns des autres. Cette configuration a été utilisée pour permettre qu'une zone s'écoule du maître principal aux esclaves même si tous les esclaves n'ont pas de connexité directe avec le maître principal. L'utilisation par UPDATE du graphe de dépendance AXFR pour la transmission interdit cette sorte de boucle de dépendance, car la transmission UPDATE n'a pas de mécanisme de détection de boucle analogue au pré essai SOA SERIAL utilisé par AXFR.

7.18 Les noms pré existants qui sont englobés par une nouvelle coupure de zone sont toujours considérés comme faisant partie de la zone parente, pour les besoins des transferts de zone, bien que les interrogations sur de tels noms se réfèrent aux serveurs de la nouvelle sous-zone. Si une coupure de zone est retirée, tous les noms de la zone parente qui étaient englobés par elle deviendront à nouveau visibles aux interrogations. (Ceci est une précision à la [RFC1034].)

7.19 Si un serveur est d'autorité à la fois pour une zone et sa fille, les interrogations sur des noms à la coupure de zone entre elles auront alors des réponses d'autorité en utilisant seulement les données de la zone fille. (Ceci est une précision à la [RFC1034].)

7.20 La mise en ordre des mises à jour en utilisant le RR SOA est problématique car il n'y a aucun moyen de savoir quels RR NS d'une zone représentent le maître principal, et les esclaves d'une zone peuvent être périmés si leurs temporisateurs SOA.REFRESH ne se sont pas écoulés depuis la dernière fois que la zone a été changée sur le maître principal. On recommande qu'une zone qui a besoin de mises à jour ordonnées utilise seulement les serveurs qui mettent en œuvre NOTIFY (voir la [RFC1996]) et IXFR (voir la [RFC1995]) et qu'un client qui reçoit une erreur de préalable lorsque il essaye une mise à jour ordonnée réessaye simplement après un délai aléatoire pour permettre à la zone de s'installer.

8. Considérations pour la sécurité

8.1 En l'absence de la technologie de la [RFC2137] ou de son équivalant, le protocole décrit par le présent document rend possible à tous ceux qui peuvent atteindre un serveur de nom d'autorité d'altérer le contenu de toutes les zones sur ce serveur. C'est une augmentation sérieuse de la vulnérabilité à partir de la technologie actuelle. Il est donc très fortement recommandé que les protocoles décrits dans le présent document ne soient pas utilisés sans les mesures de sécurité fortes de la [RFC2137] ou d'autres équivalentes, par exemple, IPsec.

8.2 Une attaque de déni de service peut être lancée en inondant un transmetteur de mise à jour avec des sessions TCP contenant des mises à jour que le serveur maître principal va finalement refuser pour des problèmes de permission. Cela survient du fait de l'exigence qu'un transmetteur de mise à jour qui reçoit une demande via TCP utilise une session TCP synchrone pour son opération de transmission. Les mécanismes de gestion de connexion du paragraphe 4.2.2 de la [RFC1035] sont suffisant pour empêcher des dommages à grande échelle à la suite d'une telle attaque, mais pas pour empêcher certaines interrogations de rester sans réponse durant l'attaque.

Remerciements

Nous tenons à remercier le groupe de travail DNSIND de l'IETF pour ses apports et son assistance, en particulier, Rob Austein, Randy Bush, Donald Eastlake, Masataka Ohta, Mark Andrews, et Robert Elz. Des remerciements tout particuliers vont à Bill Simpson, Ken Wallich et Bob Halley pour leur relecture de ce document.

Références

[RFC1035] P. Mockapetris, "Noms de domaines – Mise en œuvre et spécification", STD 13, novembre 1987. *(MàJ par RFC2137)*

[RFC1982] R. Elz, R. Bush, "Arithmétique des numéros de série", août 1996. (*MàJ* RFC1034, RFC1035) *(P.S.)*

[RFC1995] M. Ohta, "Transferts de zone par incréments dans le DNS", août 1996.

[RFC1996] P. Vixie, "Mécanisme de notification rapide des changements de zone (DNS NOTIFY)", août 1996. *(P.S.)*

[RFC2065] D. Eastlake 3rd, C. Kaufman, "Extensions de sécurité du système de noms de domaines", janvier 1997. (*Obsolète, voir* RFC2535) *(P.S.)*

[RFC2137] D. Eastlake 3rd, "Mise à jour dynamique sécurisée du système de noms de domaines", avril 1997. (*Obsolète, voir* RFC3007) *(P.S.)*

Adresse des auteurs

Yakov Rekhter
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706
téléphone : +1 914 528 0090
mél : yakov@cisco.com

Susan Thomson
Bellcore
445 South Street
Morristown, NJ 07960
téléphone : +1 201 829 4514
mél : set@thumper.bellcore.com

Jim Bound
Digital Equipment Corp.
110 Spitbrook Rd ZK3-3/U14
Nashua, NH 03062-2698
téléphone : +1 603 881 0400
mél : bound@zk3.dec.com

Paul Vixie
Internet Software Consortium
Star Route Box 159A
Woodside, CA 94062
téléphone : +1 415 747 0204
mél : paul@vix.com