Groupe de travail Réseau
Request for Comments : 1876
RFC mises à jour : 1034, 1035
Catégorie : Expérimentale
Traduction Claude Brière de L'Isle

C. Davis, Kapor Enterprises
P. Vixie,Vixie Enterprises
T. Goodwin, FORE Systems
I. Dickinson, University of Warwick
janvier 1996

Un moyen pour exprimer les informations de localisation dans le système des noms de domaine

Statut de ce mémoire
Le présent mémoire définit un protocole expérimental pour la communauté de l'Internet. Il ne spécifie aucune forme de norme de l'Internet. On appelle à des discussions et suggestions pour son amélioration. La distribution de ce mémoire n'est soumise à aucune restriction.

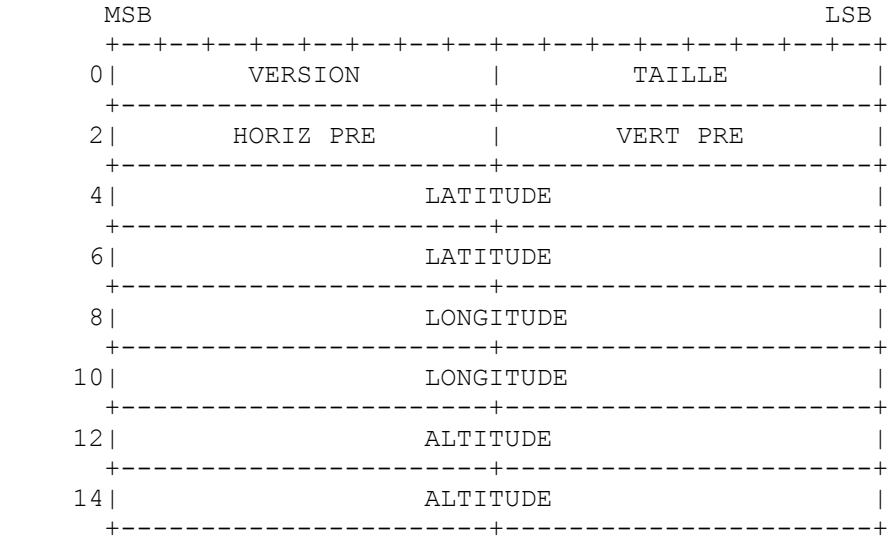
1. Résumé

Le présent mémoire définit un nouveau type d'enregistrement de ressource (RR) du DNS à des fins expérimentales. La présente RFC décrit un mécanisme pour permettre au DNS de porter des information de localisation sur les hôtes, réseaux et sous réseaux. De telles informations sont actuellement contenus pour un petit sous ensemble d'hôtes dans le fichier plat des transpositions UUCP. Cependant, tout comme le DNS a remplacé l'utilisation de HOSTS.TXT pour porter les informations d'adresses d'hôte et de réseau, il est possible de remplacer les transpositions UUCP comme transporteurs des informations de localisation.

La présente RFC définit le format d'un nouvel enregistrement de ressource (RR, *Resource Record*) pour le système des noms de domaines (DNS, *Domain Name System*) et réserve un mnémonique de type DNS correspondant (LOC) et un code numérique (29).

La présente RFC suppose que le lecteur est familier avec le DNS [RFC1034], [RFC1035]. Les données montrées dans les exemples sont pour un usage pédagogique et ne reflètent pas nécessairement l'Internet réel.

2. Format de RDATA



(octet)

où :

VERSION : numéro de version de la représentation. Il doit être zéro. Les mises en œuvre doivent vérifier ce champ et ne

pas faire d'hypothèses sur le format des versions non reconnues.

SIZE : le diamètre d'une sphère enblobant l'entité décrite, en centimètres, exprimé comme une paire d'entiers non signés de quatre bits, chacun dans la gamme de zéro à neuf, avec les quatre bits de poids fort qui représentent la base et le second nombre représentant la puissance de dix par laquelle multiplier la base. Cela permet d'exprimer des tailles de 0e0 (<1cm) à 9e9 (90 000 km). Cette représentation a été choisie de telle façon que la représentation hexadécimale puisse être lue visuellement ; 0x15 = 1e5. Les valeurs de quatre bits supérieures à 9 sont indéfinies, comme le sont les valeurs avec une base de zéro et un exposant non zéro.
Comme 20 000 000 m (représenté par la valeur 0x29) est supérieur au diamètre équatorial de l'éllipsoïde WGS 84 (12 756 274 m) il convient donc pour être utilisé comme taille "mondiale".

HORIZ PRE : précision horizontale des données, en centimètres, exprimée en utilisant la même représentation que SIZE. C'est le diamètre du "cercle d'erreur" horizontal, plutôt qu'une valeur "plus ou moins". (Cela a été choisi pour correspondre à l'interprétation de SIZE ; pour obtenir une valeur "plus ou moins", diviser par 2.)

VERT PRE : précision verticale des données, en centimètres, exprimée en utilisant la même représentation que pour SIZE. C'est l'erreur verticale potentielle totale, plutôt qu'une valeur "plus ou moins". (Cela a été choisi pour correspondre à l'interprétation de SIZE ; pour obtenir une valeur "plus ou moins", diviser par 2.) Noter que si l'altitude au dessus ou en dessous du niveau de la mer est utilisée comme approximation de l'altitude relative à l'ellipsoïde [WGS 84], la valeur de précision devrait être ajustée.

LATITUDE : latitude du centre de la sphère décrite par le champ SIZE, exprimée comme un entier de 32 bits, octet de poids fort en premier (ordre standard des octets du réseau) en millièmes de seconde d'arc. 2^31 représente l'équateur ; les nombres au dessus sont de latitude nord.

LONGITUDE : longitude du centre de la sphère décrite par le champ SIZE, exprimée comme un entier de 32 bits, octet de poids fort en premier (ordre standard des octets du réseau) en millièmes de seconde d'arc, arrondis à partir du méridien d'origine. 2^31 représente le méridien d'origine ; les nombres au dessus sont de longitude est.

ALTITUDE : altitude du centre de la sphère décrite par le champ SIZE, exprimée comme un entier de 32 bits, octet de poids fort en premier (ordre standard des octets du réseau) en centimètres, à partir d'une base de 100 000 m en dessous du sphéroïde de référence [WGS 84] utilisé par le GPS (axe semi majeur a=6 378 137,0, applatissement réciproque rf=298,257223563). Une altitude au dessus (ou en dessous) du niveau de la mer peut être utilisée comme approximation de l'altitude relative au sphéroïde [WGS 84], bien que du fait que la surface de la Terre n'est pas un sphéroïde parfait, il y aura des différences. (Par exemple, le géoïde (qui approche le niveau de la mer) pour les USA continental va de 10 mètres à 50 mètres en dessous du sphéroïde [WGS 84]. Des ajustements à ALTITUDE et/ou VERT PRE seront nécessaires dans la plupart des cas. L'Agence de cartographie de la Défense publie des valeurs de hauteur du géoïde relatives à l'ellipsoïde [WGS 84].

3. Format de fichier maître

L'enregistrement LOC est exprimé dans un fichier maître dans le format suivant :

<owner> <TTL> <class> LOC (d1 [m1 [s1]] {"N"|"S"} d2 [m2 [s2]] {"E"|"W"} alt["m"] [siz["m"] [hp["m"] [vp["m"]]]])

(Les parenthèses sont utilisées pour les données multi lignes comme spécifié dans la [RFC1035] paragraphe 5.1.)

où :

d1 :	[0 .. 90]	(latitude en degrés)
d2 :	[0 .. 180]	(longitude en degrés)
m1, m2 :	[0 .. 59]	(latitude/longitude en minutes)
s1, s2 :	[0 .. 59,999]	(latitude/longitude en secondes)
alt :	[-100000,00 .. 42849672,95] par 0,01	(altitude en mètres)
siz, hp, vp :	[0 .. 90000000,00]	(taille/précision en mètres)

Si elles sont omises, les minutes et secondes sont zéro par défaut, la taille prend 1 m par défaut, la précision horizontale prend 10 000 m par défaut, et la précision verticale 10 m par défaut. Ces valeurs par défaut sont choisies pour représenter les tailles de zone de code ZIP/postal normales, car il est souvent facile de trouver une approximation de localisation

géographique par code ZIP/postal.

4. Exemple de données

;;; Noter que ces données n'apparaîtraient pas toutes dans une fichier de zone.;;;

;; RR LOC de réseau déduit de données ZIP. Noter l'utilisation de la précision par défaut.

cambridge-net.kei.com. LOC 42 21 54 N 71 06 18 W -24m 30m

;; RR LOC d'hôte de plus grande précision. Noter l'utilisation de la précision verticale par défaut.

loiosh.kei.com. LOC 42 21 43.952 N 71 5 6.344 W -24m 1m 200m
pipex.net. LOC 52 14 05 N 00 08 50 E 10m
curtin.edu.au. LOC 32 7 19 S 116 2 25 E 10m
rwy04L.logan-airport.boston. LOC 42 21 28.764 N 71 00 51.617 W -44m 2000m

5. Application du RR LOC

5.1 Suggestions d'utilisations

Certaines utilisations du RR LOC ont déjà été suggérées, incluant les cartes des flux du cœur de réseau USENET, une application "traceroute visuel" montrant le chemin géographique d'un paquet IP, et des applications de gestion de réseau qui pourraient utiliser les RR LOC pour générer une carte des hôtes et routeurs gérés.

5.2 Algorithmes de recherche

Ce paragraphe spécifie comment utiliser le DNS pour traduire les noms de domaines et/ou adresses IP en informations de localisation.

Si une application souhaite avoir un comportement de "repli", afficher une zone moins précise ou plus grande quand un hôte n'a pas de RR LOC associé, elle PEUT prendre en charge l'utilisation de l'algorithme du paragraphe 5.2.3, comme noté aux paragraphes 5.2.1 et 5.2.2. Si un repli est désiré, ce comportement est celui RECOMMANDÉ par défaut, mais dans certains cas, il peut devoir être modifié sur la base des exigences spécifiques de l'application concernée.

Cet algorithme de recherche est conçu pour permettre aux administrateurs de réseau de spécifier la localisation d'un réseau ou sous réseau sans exiger de données de RR LOC pour chaque hôte individuel. Par exemple, un atelier informatique avec 24 stations de travail, toutes sur le même sous réseau et dans la même localisation de base, va seulement avoir besoin d'un RR LOC pour le sous réseau. (Cependant, si la localisation du serveur de fichiers a été mesurée avec plus de précision, un RR LOC RR séparé pour lui peut être placé dans le DNS.)

5.2.1 Recherche par nom

Si l'application commence par un nom, plutôt que par une adressz IP (comme le font les cartes de flux du cœur de réseau USENET) elle DOIT vérifier qu'un RR LOC est associé à ce nom. (Les enregistrements CNAME devraient être suivis comme pour tout autre type de RR.)

Si il n'y a pas de RR LOC pour ce nom, tous les enregistrements A (si il y en a) associés à ce nom PEUVENT être vérifiés pour des RR LOC de réseau (ou sous réseau) en utilisant l'algorithme "Recherche par réseau ou sous réseau" (5.2.3). Si plusieurs enregistrements A existent et ont des RR LOC de réseau ou sous réseau associés, l'application peut choisir d'utiliser un, certains, ou tous les RR LOC trouvés, éventuellement en combinaison. Il est suggéré que les hôtes multi rattachements aient des RR LOC pour leur nom dans le DNS pour éviter toute ambiguïté dans ces cas.

Noter que les noms de domaines qui n'ont pas d'enregistrement A associé doivent avoir un RR LOC associé à leur nom afin que les informations de localisation soient accessibles.

5.2.2 Recherche par adresse

Si l'application commence par une adresse IP (comme ce peut être pour une application "traceroute visuel") elle DOIT d'abord transposer l'adresse en un nom en utilisant l'espace de noms IN-ADDR.ARPA (voir la [RFC1034], paragraphe 5.2.1) puis chercher un RR LOC associé à ce nom.

Si il n'y a pas de RR LOC pour le nom, l'adresse PEUT être cherchée pour des RR LOC de réseau (ou sous réseau) en utilisant l'algorithme "Recherche par réseau ou sous réseau" (5.2.3).

5.2.3 Recherche par réseau ou sous réseau

Même si le nom d'un hôte n'a pas de RR LOC associé, le ou les réseaus/sous réseaux sur lesquels il est le peuvent. Si l'application souhaite chercher de telles données moins spécifiques, l'algorithme qui suit DEVRAIT être suivi pour trouver un RR LOC de réseau ou sous réseau associé à l'adresse IP. Cet algorithme est avec une légère adaptation celui spécifié aux paragraphes 4.3 et 4.4 de la [RFC1101].

Comme les RR LOC de sous réseau sont (s'il en est) plus spécifiques que les RR LOC de réseau, il vaut mieux les utiliser si il en est de disponibles. Afin de le faire, on construit une pile des noms de réseaux et de sous réseaux trouvés tout en effectuant la recherche de la [RFC1101], puis on parcourt la pile jusqu'à trouver un RR LOC.

1. Créer une adresse d'hôte zéro en utilisant la portion réseau de l'adresse IP (un, deux, ou trois octets pour les réseaux respectivement de classe A, B, ou C). Par exemple, pour l'hôte 128.9.2.17, sur le réseau de classe B 128.9, cela donnerait l'adresse "128.9.0.0".

2. Inverser les octets, suffixe IN-ADDR.ARPA, et interroger sur des enregistrements PTR et A. On récupère :

```
0.0.9.128.IN-ADDR.ARPA.  PTR    isi-net.isi.edu.
                         A      255.255.255.0
```

Pousser le nom "isi-net.isi.edu" sur la pile de noms à chercher ultérieurement pour les RR LOC.

3. Comme un RR A a été trouvé, répéter en utilisant un gabarit provenant du RR (255.255.255.0), pour construire une interrogation pour 0.2.9.128.IN-ADDR.ARPA. On récupère :

```
0.2.9.128.IN-ADDR.ARPA.  PTR    div2-subnet.isi.edu.
                         A      255.255.255.240
```

Pousser le nom "div2-subnet.isi.edu" sur la pile de noms à chercher ultérieurement pour des RR LOC.

4. Comme un autre RR A a été trouvé, répéter en utilisant le gabarit 255.255.255.240 (x'FFFFFFF0'), en construisant une interrogation pour 16.2.9.128.IN-ADDR.ARPA. On récupère :

```
16.2.9.128.IN-ADDR.ARPA. PTR    inc-subsubnet.isi.edu.
```

Pousser le nom "inc-subsubnet.isi.edu" sur la pile des noms à chercher ultérieurement pour des RR LOC.

5. Comme aucun RR A n'est présent à 16.2.9.128.IN-ADDR.ARPA., il n'y a plus de niveaux de sous réseaux à explorer. On prend alors le nom du sommet de la pile et on cherche un RR LOC associé. On répète jusqu'à trouver un RR LOC.

Dans ce cas, on suppose que inc-subsubnet.isi.edu n'a pas de RR LOC associé, mais que div2-subnet.isi.edu en a un. On va alors utiliser le RR LOC de div2-subnet.isi.edu comme approximation de cette localisation d'hôte. (Noter que même si isi-net.isi.edu a un RR LOC, il ne sera pas utilisé si un sous réseau a aussi un RR LOC.)

5.3 Applicabilité aux classes non IN et aux adresses non IP

L'enregistrement LOC est défini pour toutes les classes de RR, et peut être utilisé avec des classes non IN comme HS et CH. La sémantique d'une telle utilisation n'est pas définie dans le présent mémoire.

L'algorithme de recherche du paragraphe 5.2.3 peut être adapté aux autres schémas d'adressage en étendant le codage de noms de réseaux de la [RFC1101] pour couvrir ces schémas. Ces extensions ne sont pas définies dans ce mémoire.

6. Références

[RFC1034] P. Mockapetris, "Noms de domaines - Concepts et facilités", STD 13, novembre 1987. (*MàJ par* RFC1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020, 8482, 8767)

[RFC1035] P. Mockapetris, "Noms de domaines – Mise en œuvre et spécification", STD 13, novembre 1987. (*MàJ par* RFC1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766, 8482, 8767)

[RFC1101] P. Mockapetris, "Codage par le DNS des noms de réseau et autres types", avril 1989.

[WGS 84] United States Department of Defense; "DoD WGS-1984 - Its Definition and Relationships with Local Geodetic Systems"; Washington, D.C.; 1985; Report AD-A188 815 DMA; 6127; 7-R-138-R; CV, KV;

7. Considérations de sécurité

Des informations de RR LOC de haute précision pourraient être utilisées pour planifier une pénétration de sécurité physique, conduisant à de potentielles attaques de déni de machine. Pour éviter de paraître suggérer cette méthode à de potentiels attaquants, on a refusé la proposition de nommer ce RR "ICBM".

8. Adresse des auteurs

Le groupe des auteurs peut être joint à <loc@pipex.net>.

Christopher Davis
Kapor Enterprises, Inc.
238 Main Street, Suite 400
Cambridge, MA 02142
USA
téléphone : +1 617 576 4532
mél : ckd@kei.com

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

Tim Goodwin
Public IP Exchange Ltd (PIPEX)
216 The Science Park
Cambridge CB4 4WA
UK
téléphone : +44 1223 250250
mél : tim@pipex.net

Ian Dickinson
FORE Systems
2475 The Crescent
Solihull Parkway
Birmingham Business Park
B37 7YE UK
téléphone: +44 121 717 4444
mél : idickins@fore.co.uk

Appendice A. Échantillon de sous programmes de conversion

/* Sous programmes pour convertir entre le format de RR sur le réseau et le format de fichier de zone. Ne contient pas la conversion de/vers les degrés décimaux ; diviser ou multiplier par 60*60*1000 pour cela. */

```
static unsigned int poweroften[10] = {1, 10, 100, 1000, 10000, 100000, 1000000,10000000,100000000,1000000000};

/*Prend une valeur de précision/taille XeY, retourne une représentation de chaîne.*/
static const char *
precsize_ntoa(prec)
        u_int8_t prec;
```

```
{
        static char retbuf[sizeof("90000000.00")];
        unsigned long val;
        int mantissa, exponent;

        mantissa = (int)((prec >> 4) & 0x0f) % 10;
        exponent = (int)((prec >> 0) & 0x0f) % 10;

        val = mantissa * poweroften[exponent];

        (void) sprintf(retbuf,"%d.%.2d", val/100, val%100);
        return (retbuf);
}

/* Convertit l'ascii taille/précision X * 10**Y(cm) en 0xXY. Déplace le pointeur.*/
static u_int8_t
precsize_aton(strptr)
        char **strptr;
{
        unsigned int mval = 0, cmval = 0;
        u_int8_t retval = 0;
        register char *cp;
        register int exponent;
        register int mantissa;

        cp = *strptr;

        while (isdigit(*cp))
                mval = mval * 10 + (*cp++ - '0');

        if (*cp == '.') {                               /* centimètres */
                cp++;
                if (isdigit(*cp)) {
                        cmval = (*cp++ - '0') * 10;
                        if (isdigit(*cp)) {
                                cmval += (*cp++ - '0');
                        }
                }
        }
        cmval = (mval * 100) + cmval;

        for (exponent = 0; exponent < 9; exponent++)
                if (cmval < poweroften[exponent+1])
                        break;

        mantissa = cmval / poweroften[exponent];
        if (mantissa > 9)
                mantissa = 9;

        retval = (mantissa << 4) | exponent;

        *strptr = cp;

        return (retval);
}

/* Convertit l'ascii lat/lon en un nombre non signé codé sur 32 bits. Déplace le pointeur. */
static u_int32_t
latlon2ul(latlonstrptr,which)
        char **latlonstrptr;
        int *which;
```

```
{
        register char *cp;
        u_int32_t retval;
        int deg = 0, min = 0, secs = 0, secsfrac = 0;

        cp = *latlonstrptr;

        while (isdigit(*cp))
                deg = deg * 10 + (*cp++ - '0');

        while (isspace(*cp))
                cp++;

        if (!(isdigit(*cp)))
                goto fndhemi;

        while (isdigit(*cp))
                min = min * 10 + (*cp++ - '0');

        while (isspace(*cp))
                cp++;

        if (!(isdigit(*cp)))
                goto fndhemi;

        while (isdigit(*cp))
                secs = secs * 10 + (*cp++ - '0');

        if (*cp == '.') {                               /* secondes décimales */
                cp++;
                if (isdigit(*cp)) {
                        secsfrac = (*cp++ - '0') * 100;
                        if (isdigit(*cp)) {
                                secsfrac += (*cp++ - '0') * 10;
                                if (isdigit(*cp)) {
                                        secsfrac += (*cp++ - '0');
                                }
                        }
                }
        }

        while (!isspace(*cp))                           /* si il y a des déchets en queue */
                cp++;

        while (isspace(*cp))
                cp++;

fndhemi:
        switch (*cp) {
        case 'N': case 'n':
        case 'E': case 'e':
                retval = ((unsigned)1<<31)
                        + (((((deg * 60) + min) * 60) + secs) * 1000)
                        + secsfrac;
                break;
        case 'S': case 's':
        case 'W': case 'w':
                retval = ((unsigned)1<<31)
                        - (((((deg * 60) + min) * 60) + secs) * 1000)
                        - secsfrac;
                break;
```

```
        default:
                retval = 0;                     /* valeur invalide – indique l'erreur */
                break;
        }

        switch (*cp) {
        case 'N': case 'n':
        case 'S': case 's':
                *which = 1;                     /* latitude */
                break;
        case 'E': case 'e':
        case 'W': case 'w':
                *which = 2;                     /* longitude */
                break;
        default:
                *which = 0;                     /* erreur */
                break;
        }

        cp++;                                   /* saute l'hémisphère */

        while (!isspace(*cp))                   /* si il y a des déchets en queue */
                cp++;

        while (isspace(*cp))                    /* passe au prochain champ */
                cp++;

        *latlonstrptr = cp;

        return (retval);
}

/* Convertit une représentation de fichier de zone dans une chaîne en une représentation de RDATA dans le réseau. */
u_int32_t
loc_aton(ascii, binary)
        const char *ascii;
        u_char *binary;
{
        const char *cp, *maxcp;
        u_char *bcp;

        u_int32_t latit = 0, longit = 0, alt = 0;
        u_int32_t lltemp1 = 0, lltemp2 = 0;
        int altmeters = 0, altfrac = 0, altsign = 1;
        u_int8_t hp = 0x16;                     /* par défaut = 1e6 cm = 10 000,00 m = 10 km */
        u_int8_t vp = 0x13;                     /* par défaut = 1e3 cm = 10,00 m */
        u_int8_t siz = 0x12;                    /* par défaut = 1e2 cm = 1,00 m */
        int which1 = 0, which2 = 0;

        cp = ascii;
        maxcp = cp + strlen(ascii);

        lltemp1 = latlon2ul(&cp, &which1);

        lltemp2 = latlon2ul(&cp, &which2);

        switch (which1 + which2) {
        case 3:                                 /* 1 + 2, la seule combinaison valide */
                if ((which1 == 1) && (which2 == 2)) {   /* cas normal  */
                        latit = lltemp1;
                        longit = lltemp2;
```

```
        } else if ((which1 == 2) && (which2 == 1)) {          /*inversé */
                longit = lltemp1;
                latit = lltemp2;
        } else {                                              /* une forme de coupure */
                return 0;
        }
        break;
default:                                                      /* on n'a pas eu une de chaque */
        return 0;
}

/* altitude */
if (*cp == '-') {
        altsign = -1;
        cp++;
}

if (*cp == '+')
        cp++;

while (isdigit(*cp))
        altmeters = altmeters * 10 + (*cp++ - '0');

if (*cp == '.') {                                             /* mètres en décimal */
        cp++;
        if (isdigit(*cp)) {
                altfrac = (*cp++ - '0') * 10;
                if (isdigit(*cp)) {
                        altfrac += (*cp++ - '0');
                }
        }
}

alt = (10000000 + (altsign * (altmeters * 100 + altfrac)));

while (!isspace(*cp) && (cp < maxcp))                         /* si il y a des déchets en queue ou m */
        cp++;

while (isspace(*cp) && (cp < maxcp))
        cp++;

if (cp >= maxcp)
        goto defaults;

siz = precsize_aton(&cp);

while (!isspace(*cp) && (cp < maxcp))                         /* si il y a des déchets en queue ou m */
        cp++;

while (isspace(*cp) && (cp < maxcp))
        cp++;

if (cp >= maxcp)
        goto defaults;

hp = precsize_aton(&cp);

while (!isspace(*cp) && (cp < maxcp))                         /* si il y a des déchets en queue ou m*/
        cp++;

while (isspace(*cp) && (cp < maxcp))
```

```
                cp++;

        if (cp >= maxcp)
                goto defaults;

        vp = precsize_aton(&cp);

 defaults:

        bcp = binary;
        *bcp++ = (u_int8_t) 0;                          /* octet de version */
        *bcp++ = siz;
        *bcp++ = hp;
        *bcp++ = vp;
        PUTLONG(latit,bcp);
        PUTLONG(longit,bcp);
        PUTLONG(alt,bcp);

        return (16);                                    /* taille de RR en octets */
}

/* prend un RR LOC sur le réseau et l'imprime dans le format de fichier zone (lisible par l'homme). */
char *
loc_ntoa(binary,ascii)
        const u_char *binary;
        char *ascii;
{

        static char tmpbuf[255*3];

        register char *cp;
        register const u_char *rcp;

        int latdeg, latmin, latsec, latsecfrac;
        int longdeg, longmin, longsec, longsecfrac;
        char northsouth, eastwest;
        int altmeters, altfrac, altsign;

        const int referencealt = 100000 * 100;

        int32_t latval, longval, altval;
        u_int32_t templ;
        u_int8_t sizeval, hpval, vpval, versionval;

        char *sizestr, *hpstr, *vpstr;

        rcp = binary;
        if (ascii)
                cp = ascii;
        else {
                cp = tmpbuf;
        }

        versionval = *rcp++;

        if (versionval) {
                sprintf(cp,"; erreur : version de RR LOC inconnue");
                return (cp);
        }

        sizeval = *rcp++;
```

```
hpval = *rcp++;
vpval = *rcp++;

GETLONG(templ,rcp);
latval = (templ - ((unsigned)1<<31));

GETLONG(templ,rcp);
longval = (templ - ((unsigned)1<<31));

GETLONG(templ,rcp);
if (templ < referencealt) {                    /* en dessous du sphéroïde WGS 84 */
        altval = referencealt - templ;
        altsign = -1;
} else {

        altval = templ - referencealt;
        altsign = 1;
}

if (latval < 0) {
        northsouth = 'S';
        latval = -latval;
}
else
        northsouth = 'N';

latsecfrac = latval % 1000;
latval = latval / 1000;
latsec = latval % 60;
latval = latval / 60;
latmin = latval % 60;
latval = latval / 60;
latdeg = latval;

if (longval < 0) {
        eastwest = 'W';
        longval = -longval;
}
else
        eastwest = 'E';

longsecfrac = longval % 1000;
longval = longval / 1000;
longsec = longval % 60;
longval = longval / 60;
longmin = longval % 60;
longval = longval / 60;
longdeg = longval;

altfrac = altval % 100;
altmeters = (altval / 100) * altsign;

sizestr = savestr(precsize_ntoa(sizeval));
hpstr = savestr(precsize_ntoa(hpval));
vpstr = savestr(precsize_ntoa(vpval));

sprintf(cp,
        "%d %.2d %.2d.%.3d %c %d %.2d %.2d.%.3d %c %d.%.2dm %sm %sm %sm",
        latdeg, latmin, latsec, latsecfrac, northsouth,
        longdeg, longmin, longsec, longsecfrac, eastwest,
```

```
                altmeters, altfrac, sizestr, hpstr, vpstr);
        free(sizestr);
        free(hpstr);
        free(vpstr);

        return (cp);
}
```