Groupe de travail Réseau
Request for Comments : 3402
RFC rendues obsolètes : 2915, 2168
Catégorie : En cours de normalisation

M. Mealling
Verisign
octobre 2002
Traduction Claude Brière de L'Isle

Système de découverte dynamique de délégation (DDDS) Partie II : l'algorithme

Résumé
Le présent document décrit l'algorithme du système de découverte dynamique de délégation (DDDS) pour appliquer les règles de transformation de chaîne restituée de façon dynamique à une chaîne d'application unique. Les règles de transformation bien formées vont refléter la délégation de gestion des informations associées à la chaîne. Le présent document fait aussi partie d'une série qui est complètement spécifiée dans "Système de découverte dynamique de délégation (DDDS) Partie I : DDDS complet" (RFC3401). Il est important de noter qu'il est impossible de lire et comprendre un des documents de la série sans lire les autres.

Table des Matières

1. Introduction

Le système de découverte dynamique de délégation (DDDS) est utilisé pour mettre en œuvre un lien lâche de chaînes de données, afin de prendre en charge les systèmes de délégation à configuration dynamique. Le DDDS fonctionne en transposant une chaîne unique en données mémorisées au sein d'une base de données DDDS en appliquant de façon itérative les règles de transformation de chaîne jusqu'à atteindre une condition de fin.

Le présent document décrit l'algorithme DDDS général, et non une application ou scénario d'usage particulier. La série entière des documents est spécifiée dans "Système de découverte dynamique de délégation (DDDS) Partie I : DDDS complet" [RFC3401]. Il est important de noter qu'il est impossible de lire et comprendre un des documents de la série sans lire les autres.

L'histoire de DDDS est une évolution à partir du travail réalisé par le groupe de travail Noms de ressource uniformes. Lorsque les noms de ressource uniformes (URN) [RFC2141] ont été formulés à l'origine, il y avait le désir de localiser un serveur d'autorité pour un URN qui ne contienne (par conception) aucune information sur les localisations réseau. Le système qui a été formulé pouvait utiliser une base de données de règles qui pourraient s'appliquer à un URN pour découvrir des informations sur des morceaux de syntaxe spécifiques. Ce système était appelé à l'origine le service de découverte de résolveur (RDS, *Resolver Discovery Service*) [RFC2276] et ne s'appliquait qu'aux URN.

Avec le temps, d'autres systèmes ont commencé à appliquer ce même algorithme et son infrastructure à d'autres systèmes, sans rapport avec les URN (voir à la Section 6 des exemples des autres façons d'utiliser le DDDS). Cela a causé le changement de certaines des hypothèses sous-jacentes et un besoin de clarification. Cette série de documents est une mise à jour des spécifications originales des URN afin de permettre de développer de nouvelles applications et bases de données de règles d'une façon normalisée.

Le présent document rend obsolètes les [RFC2168] et [RFC2915] et met à jour la [RFC2276].

2. Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" dans ce document sont à interpréter comme décrit dans la [RFC2119].

Chaîne unique d'application
C'est une chaîne qui est l'entrée initiale d'une application DDDS. La structure lexicale de cette chaîne doit impliquer un chemin unique de délégation, qui est analysé et retracé par la sélection répétée et l'application des règles de réécriture.

Règle de réécriture
C'est une règle qui est appliquée à une chaîne unique d'application pour produire une nouvelle clé de sélection d'une nouvelle règle de réécriture à partir de la base de données des règles, ou une chaîne de résultat finale qui est retournée à l'application invoquante. Aussi appelée simplement une règle.

Première règle bien connue
C'est une règle de réécriture qui est définie par l'application et n'est pas réellement dans la base de données des règles. Elle est utilisée pour produire la première clé valide.

Règle terminale
C'est une règle de réécriture qui, lorsque elle est utilisée, donne une chaîne qui est le résultat final du processus DDDS, plutôt qu'une autre clé de la base de données.

Application
C'est un ensemble de protocoles et spécifications qui spécifient les valeurs réelles pour les diverses parties généralisées de l'algorithme DDDS. Une application doit définir la syntaxe et la sémantique de la chaîne unique d'application, la première règle bien connue, et une ou plusieurs bases de données qui sont valides pour l'application.

Base de données de règles
C'est toute mémorisation de règles telle qu'une clé unique puisse identifier un ensemble de règles qui spécifient l'étape de délégation utilisée lorsque cette clé particulière est utilisée.

Services
Une base de données de règles communes peut être utilisée pour associer différents services à une chaîne unique d'application donnée ; par exemple, différentes fonctions de protocole, différentes caractéristiques de fonctionnement, une ségrégation géographique, une rétro compatibilité, etc. Différents services possibles seraient un message qui reçoit des services de messagerie électronique/télécopie/messagerie vocale, l'équilibrage de charge sur des serveurs de la Toile, le choix d'un serveur miroir dans le voisinage, des compromis entre coûts et performances, etc. Ces services sont inclus au titre d'une règle pour permettre à l'application de prendre des décisions d'embranchement sur la base de l'applicabilité d'une ou de l'autre branche du point de vue d'un service.

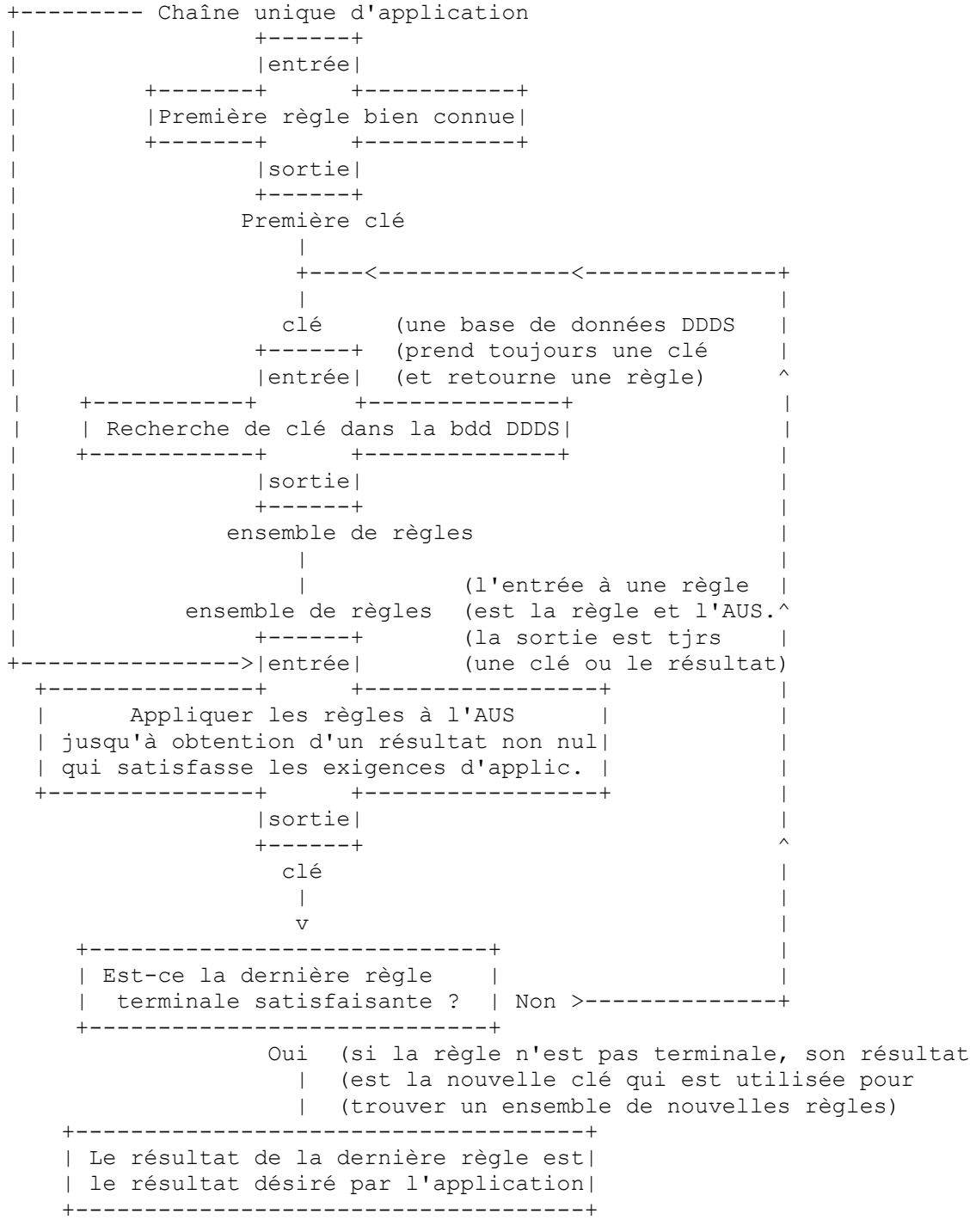
Fanions
La plupart des applications vont exiger un moyen pour qu'une règle signale à l'application que certaines règles donnent des résultats particuliers alors que d'autres ne le font pas ; par exemple, différents formats de résultats, des mécanismes d'extension, la signalisation de règles terminales, etc. La plupart des bases de données vont définir un champ Fanions qu'une application peut utiliser pour coder diverses valeurs qui expriment ces signaux.

3. Algorithme

L'algorithme DDDS se fonde sur le concept de règles de réécriture. Ces règles sont collectées dans une base de données de règles, et on y accède pas des clés uniques particulières. Une règle donnée, lorsque appliquée à une chaîne unique d'application (AUS, *Application Unique String*) transforme cette chaîne en une nouvelle clé qui peut être utilisée pour restituer une nouvelle règle à partir de la base de données de règles. Cette nouvelle règle est alors appliquée à nouveau à la chaîne unique d'application originale et le cycle se répète jusqu'à atteindre une condition de fin. Une application NE DOIT PAS appliquer une règle au résultat de la règle précédente. Toutes les règles de réécriture pour toutes les applications DOIVENT TOUJOURS s'appliquer à la même chaîne unique d'application exacte avec laquelle l'algorithme a commencé.

C'est une hypothèse fondamentale que la chaîne unique d'application a une sorte de structure lexicale régulière à laquelle peuvent être appliquées les règles. On a pour hypothèse du DDDS que l'élément lexical utilisé pour prendre une décision de délégation est assez simple pour être contenue au sein de la chaîne unique d'application elle-même. Le DDDS ne résout pas le cas où une décision de délégation est prise en utilisant les connaissances contenues en-dehors d'une AUS et de la règle (heure, transactions financière, gestion des droits, etc.).

Dans un diagramme, l'algorithme ressemble à :



3.1 Composants d'une règle

Une règle est faite de quatre éléments d'information :

Priorité
C'est un simple nombre utilisé pour montrer laquelle de deux règles par ailleurs égales peut avoir la préséance. Cela permet à la base de données d'exprimer les règles qui peuvent offrir à peu près le même résultat mais un chemin de délégation peut être plus rapide, meilleur, moins cher que l'autre.

Ensemble de fanions
Les fanions sont utilisés pour spécifier des attributs de la règle qui déterminent si cette règle est la dernière à appliquer. La dernière règle est appelée la règle terminale et son résultat DEVRAIT être le résultat attendu pour l'application. Les fanions sont uniques pour une application. Une application peut spécifier qu'elle utilisé un fanion défini par une autre application mais elle DOIT utiliser la définition de cette autre application. Une application ne peut pas redéfinir un fanion utilisé par une autre application. Cela peut signifier qu'un registre des fanions sera nécessaire à l'avenir mais ce n'est pas une exigence pour le moment.

Description de services
Les services sont utilisés pour spécifier les attributs sémantiques d'une branche de délégation particulière. Il y a de nombreux cas où deux branches de délégation sont identiques sauf que une des délégations donne un résultat qui fournit un ensemble de caractéristiques alors que l'autre donne un autre ensemble. Les caractéristiques peuvent comporter des questions de comportement comme l'équilibrage de charge, une ségrégation géographique du trafic, des fonctions dégradées mais rétro compatibles pour des clients de longue date, etc. Par exemple, deux règles peuvent s'appliquer également à une décision de délégation spécifique d'une chaîne. Une règle peut conduire à une règle terminale qui produit des informations à utiliser dans des environnements de forte disponibilité, alors que l'autre peut conduire à un service d'archives qui peut être plus lent mais plus stable sur le long terme.

Expression de substitution
C'est la partie réelle de la modification de la chaîne. C'est une combinaison d'une expression régulière étendue POSIX [IEEE] et d'une chaîne de remplacement similaire à une expression de substitution de style sed Unix.

3.2 Syntaxe d'expression de substitution

Le ou les jeux de caractères dans laquelle est l'expression de substitution et peut agir dépendent à la fois de l'application et de la base de données utilisées. Une application DOIT définir quels sont les jeux de caractères admis pour la chaîne unique d'application. Une spécification de base de données DDDS DOIT définir les jeux de caractères qui sont requis pour produire ses clés et pour la façon dont l'expression de substitution elle-même est codée. Les caractères exigés par la grammaire ci-dessous ont une signification une fois qu'un jeu de caractères spécifique est défini pour la base de données et/ou l'application.

La syntaxe de la partie expression de substitution de la règle est une expression de substitution de style sed. Les vraies expressions de substitution de style sed ne sont pas appropriées pour l'utilisation dans cette application pour diverses raisons, donc le contenu du champ regexp DOIT suivre cette grammaire :

```
subst-expr      = delim-char ere delim-char repl delim-char *flags
delim-char      = "/" / "!" / <Tout octet qui n'est pas en 'POS-DIGIT' ou 'fanions'>
                  ; Toutes les occurrences d'un delim_char dans une subst_expr DOIVENT être le même caractère.>
ere             = <Expression régulière étendue POSIX>
repl            = *(chaîne / backref)
chaîne          = *(anychar / escapeddelim)
anychar         = <tout caractère autre que delim-char>
escapeddelim    = "\" delim-char
backref         = "\" POS-DIGIT
fanions         = "i"
POS-DIGIT       = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
```

Le résultat de l'application de l'expression de substitution à la chaîne DOIT résulter en une clé qui obéit aux règles de la base de données (sauf bien sûr si elle est une règle terminale auquel cas le résultat suit les règle de l'application). Comme il est possible que l'expression régulière soit spécifiée de façon impropre, de telle sorte qu'une clé non conforme soit construite, le logiciel client DEVRAIT vérifier que le résultat est une clé de base de données légale avant de l'utiliser.

Les expressions "backref" dans la portion "repl" de l'expression de substitution sont remplacées par la chaîne de caractères (éventuellement vide) comprise entre '(' et ')' dans la portion ERE de l'expression de substitution. N est un seul chiffre de 1 à 9, inclus. Il spécifie la N[ème] expression backref, celle qui commence par la N[ème] '(' et continue jusqu'à la ')' correspondante. Par exemple, le ERE "(A(B(C)DE)(F)G)" a les expressions backref :

\1 = ABCDEFG
\2 = BCDE
\3 = C
\4 = F
\5..\9 = erreur – pas de sous expression correspondante

Le fanion "i" indique que le ERE correspondant DEVRA être effectué de façon insensible à la casse. De plus, tout remplacement backref PEUT être normalisé en minuscules lorsque le fanion"i" est mis. Ce fanion n'a de signification que lorsque l'application et la base de données définissent un jeu de caractères où l'insensibilité à la casse est valide.

Le premier caractère dans l'expression de substitution devra être utilisé comme le caractère qui délimite les composants de l'expression de substitution. Il DOIT y avoir exactement trois occurrences sans échappement du caractère délimiteur dans une expression de substitution. Comme les occurrences avec échappement du caractère délimiteur seront interprétées comme de ce caractère, les chiffres NE DOIVENT PAS être utilisés comme délimiteurs. Les "backref" seraient confondus avec des chiffres littéraux si ceux-ci étaient admis. De même, si des fanions sont spécifiés dans l'expression de substitution, le caractère délimiteur NE DOIT PAS être aussi un caractère de fanion.

3.3 Algorithme complet

Ce qui suit est l'algorithme DDDS complet exact :

1. La première règle bien connue est appliquée à la chaîne unique d'application qui produit une clé.

2. L'application interroge la base de données sur l'ensemble ordonné de règles qui sont liées à cette clé (voir la note ci-dessous pour les détails sur l'ordre).

3. On applique l'expression de substitution pour chaque règle dans la liste, dans l'ordre, à la chaîne unique d'application jusqu'à produire une chaîne non vide. La position dans la liste est notée et la règle qui a produit la chaîne non vide est utilisée pour l'étape suivante. Si l'étape suivante rejette cette règle et revient à cette étape, le traitement de l'application de l'expression de substitution se continue au point où il avait quitté. Si la liste est épuisée sans correspondance valide, il est alors notifié à l'application qu'aucun résultat valide n'est disponible.

4. Si la description de service de la règle ne satisfait pas aux exigences du client, retourner à l'étape 3 et continuer à travers la liste des règles déjà restituées. Si cela ne satisfait pas aux exigences du client, cette règle est alors utilisée pour l'étape suivante. Si et seulement si le client est capable de la traiter et si la spécification de l'application l'estime sûr, le client peut faire une note de la règle actuelle mais retourner quand même à l'étape 3 bien qu'il l'ait rejetée. Dans l'un et l'autre cas, le résultat de cette étape est une seule règle.

5. Si la partie Fanions de la règle stipule que cette règle N'EST PAS terminale, retourner à l'étape 2 avec le résultat de substitution comme nouvelle clé.

6. Notifier à l'application que le processus est termine et a fourni à l'application les parties Fanions et Services de la règle ainsi que le résultat de la dernière expression de substitution.

Note 1 : Dans certaines applications et/ou bases de données, l'ensemble de résultats peut exprimer le cas où deux règles ou plus sont considérées comme égales. Ces règles sont traitées comme la même règle, chacune ayant éventuellement une priorité qui est utilisée pour communiquer une préférence pour des règles par ailleurs équivalentes. Cela permet aux règles d'agir comme position de repli pour les autres. On DEVRAIT noter que c'est une préférence réelle, et non un mécanisme d'équilibrage de charge. Les applications DEVRAIENT définir la différence avec soin.

Note 2 : Les bases de données peuvent avoir ou non des règles qui déterminent quand et comment y expirent les enregistrements (date d'expiration, durée de vie, etc.). Ces mécanismes d'expiration DOIVENT être suivis dans tous les cas. Précisément, comme l'expiration d'un enregistrement de base de données pourrait causer la restitution d'une nouvelle règle qui serait non compatible avec les règles précédentes, alors que dans l'algorithme toute tentative d'optimiser le processus en revenant aux clés précédentes et les règles DOIT assurer qu'aucune règle précédemment restituée n'est expirée. Si une règle est expirée, l'application DOIT alors recommencer à l'étape 1.

4. Spécifier une application

Pour que cet algorithme soit utile, on DOIT écrire une spécification qui décrit une application et une ou plusieurs bases de données. Afin de spécifier une application, les éléments d'information suivants sont nécessaires :

Chaîne unique d'application :
C'est la seule chaîne à laquelle vont s'appliquer les règles de réécriture. La chaîne DOIT avoir une structure régulière et être unique au sein de l'application de telle sorte que quiconque applique les règles tirées de la même base de données doive terminer avec les mêmes clés. Par exemple, l'application de résolution d'URI définit la chaîne unique d'application comme étant un URI.

Aucune application ne doit définir une chaîne unique d'application telle que la clé obtenue par une règle de réécriture soit traitée comme chaîne unique d'application pour l'entrée dans une nouvelle règle. Cela conduit à des règles de réécriture de style sendmail qui sont fragiles et enclines à l'erreur. La seule exception est lorsque une application définit un fanion ou état où les règles pour cette application sont suspendues et qu'une nouvelle application DDDS ou quelque autre ensemble arbitraire de règles les supplante. Si c'est alors le cas, par définition, aucune de ces règles ne s'applique. Un tel cas peut se trouver dans l'application de résolution d'URI qui définit le fanion 'p' qui indique que la prochaine étape est "spécifique du protocole" et ceci sort du domaine d'application de DDDS.

Première règle bien connue
C'est la première règle qui, lorsque elle est appliquée à une chaîne unique d'application, produit la première clé valide. Elle peut être exprimée sous la même forme qu'une règle ou elle peut être quelque chose de plus complexe. Par exemple, l'application de résolution d'URI peut spécifier que la règle est que la séquence de caractères dans l'URI jusqu'aux premiers deux points non inclus (le schéma d'URI) est la première clé.

Base de données valide :
L'application peut définir quelles bases de données sont valides. Pour chaque base de données, l'application DOIT définir comment le résultat de la première règle bien connue (la première clé) est transformé en quelque chose de valide pour la base de données. Par exemple, l'application de résolution d'URI pourrait utiliser le système des noms de domaines (DNS) comme base de données. L'opération pour transformer cette première clé en quelque chose de valide pour la base de données serait de la transformer en un nom de domaine valide pour le DNS. De plus, pour chaque base de données que définit une application, elle DOIT aussi spécifier quels sont les jeux de caractères valides qui vont produire les clés correctes. Dans l'exemple de la résolution d'URI montré ici, le jeu de caractères d'un URI est l'ASCII à 7 bits qui correspond très bien avec la limitation à 8 bits du DNS sur les caractères dans ses fichiers de zone.

Résultat attendu :
L'application DOIT définir ce que DEVRAIT être le résultat attendu de la règle terminale. Par exemple, l'application de résolution d'URI est concernée par la découverte de serveurs qui contiennent des données d'autorité sur un URI donné. Donc, le résultat de la règle terminale serait des informations (hôtes, accès, protocoles, etc.) qui seraient utilisées pour contacter ce serveur d'autorité.

Dans le passé, il y a eu une certaine confusion en ce qui concerne l'équilibrage de charge et l'utilisation de la "priorité du DDDS". Les applications DEVRAIENT savoir que la priorité d'une certaine règle est juste ceci : un moyen de spécifier que cette règle est "meilleure, plus rapide, moins chère" qu'une autre. Si une application a besoin d'une méthode pour permettre à un client d'équilibrer la charge entre des serveurs (c'est-à-dire, une sélection aléatoire pondérée, etc.) il DEVRAIT alors le faire en dehors de l'algorithme DDDS. Par exemple, les applications qui utilisent la base de données du DNS peuvent utiliser l'enregistrement de ressource SRV *(serveur)* comme moyen de signifier qu'un service particulier est en fait traité par plusieurs hôtes coopérant les uns avec les autres. La différence étant que l'équilibrage de charge est fait entre des hôtes qui sont identiques les uns aux autres alors qu'un DDDS est concerné par les chemins de délégation qui ont un ensemble de caractéristiques ou un domaine administratif particulier.

5. Spécifier une base de données

De plus, toute application DOIT avoir au moins une base de données correspondante d'où restituer les règles. Il est important de noter qu'une base de données particulière peut être utilisée par plus d'une application. Si c'est le cas, chaque règle DOIT utiliser une certaine combinaison de ses services et/ou expressions de substitution pour correspondre seulement aux chaînes uniques d'application pour lesquelles elle est valide.

Une spécification de base de données DOIT inclure les éléments d'information suivants :

Spécification générale :
La base de données DOIT avoir une spécification générale. Elle peut faire référence à d'autres normes (SQL, DNS, etc.) ou elle peut être entièrement spécifiée dans un nouveau système de base de données. Cette spécification DOIT être claire sur les jeux de caractères admis qui existent afin de savoir dans quel jeu de caractères sont codées les clés et les règles.

Procédure de recherche :
Elle spécifie comment est formulée une interrogation et comment elle est soumise à la base de données. Dans le cas de bases de données qui sont utilisés pour d'autres objets (comme le DNS) la spécification DOIT être claire sur la façon dont est formulée une interrogation, en particulier pour que la base de données soit une base de données DDDS. Par exemple, une base de données fondée sur le DNS DOIT spécifier quels enregistrements de ressource ou quels types d'interrogations sont utilisés.

Format de clé :
Si des opérations sont nécessaires afin de transformer une clé en quelque chose de valide pour la base de données, elles DOIVENT être clairement définies. Par exemple, dans le cas d'une base de données du DNS, les clés DOIVENT être construites comme des noms de domaine valides.

Format de règle :
C'est la spécification du format du résultat d'une règle.

Procédure d'insertion de règle :
C'est la spécification de la façon dont une règle est insérée dans la base de données. Cela peut inclure des déclarations de politique sur le fait qu'il est ou non permis d'ajouter une règle.

Évitement de collision de règle :
Comme une base de données peut être utilisée par plusieurs applications (ENUM et résolution d'URI par exemple) la spécification DOIT être claire sur la façon dont les collisions de règles seront évitées. Il y a normalement deux méthodes pour traiter cela : 1) interdire qu'une clé soit valide dans deux différentes applications ; 2) si 1 n'est pas possible, écrire alors l'expression de substitution de façon telle que la partie d'expression régulière contienne assez de la chaîne unique d'application au titre de sa correspondance pour différencier les deux applications.

6. Exemples

Les exemples qu'on donne ici ne sont qu'à des fins pédagogiques. Ils sont tirés spécifiquement d'applications fictives qui n'ont été spécifiées dans aucun document publié.

6.1 Un système d'identification de pièces d'automobiles

Imaginons dans cet exemple un système dans lequel tous les fabricants d'automobiles se groupent pour un système normalisé de numérotation des pièces pour les diverses parties qui constituent le processus de fabrication et de réparation des automobiles (écrous, boulons, châssis, instruments, etc.). Le problème d'un tel système est que l'industrie automobile est un système réparti dans lequel des éléments sont construits par divers sous-traitants répartis tout autour du monde. Pour trouver les informations sur une certaine partie, un système DOIT être capable de trouver qui fabrique cette partie pour le contacter à son sujet.

Pour faciliter ce système réparti, le numéro d'identification alloué à une partie l'est hiérarchiquement de telle sorte que les cinq premiers chiffres constituent le numéro d'identification du constructeur de cette partie. Les trois chiffres suivants sont l'identifiant d'une ligne d'automobiles (Ford, Toyota, etc.). Le reste des chiffres est alloué par le fabricant de la pièce selon les règles fixées par ce fabricant.

L'industrie automobile décide d'utiliser le DDDS pour créer un système de restitution des informations réparties qui achemine les interrogations au propriétaire réel des données. L'industrie spécifie une base de données et une syntaxe d'interrogation pour les règles de réécriture des restitutions (le réseau APIDA) puis spécifie l'application DDDS d'identification des pièces automobiles (APIDA, *Auto Parts Identification DDDS Application*).

La spécification APIDA définirait ce qui suit :
- Chaîne unique d'application : le numéro de pièce.
- Première règle bien connue : prendre les cinq premiers chiffres (le numéro d'identification du fabricant) et l'utiliser comme clé.
- Bases de données valides : le réseau APIDA.
- Résultat attendu : informations EDIFAC sur la pièce.

La spécification de la base de données du réseau APIDA définirait ce qui suit :
- o Spécification générale : un réseau de base de données et services à capacité d'EDI qui, quand on lui donne un sous composant d'un numéro de pièce va retourner une règle de réécriture codée en XML.
- o Procédure de recherche : suivant les protocoles normaux du réseau APIDA, demander au réseau une règle de réécriture pour la clé.
- o Format de clé : aucune conversion n'est requise.
- o Format de règle : voir dans la documentation du réseau APIDA le DTD XML.
- o Procédure d'insertion de règle : déterminée par l'autorité qui a le contrôle de chaque section du numéro de pièce. C'est-à-dire que pour obtenir un identifiant de fabricant on DOIT être un membre de l'association des fabricants de pièces d'automobiles.

Pour illustrer comment pourrait fonctionner le système, imaginons le numéro de pièce "4747301AB7D". Le système prendrait les cinq premiers chiffres, '47473' et demanderait au réseau cette règle de réécriture. Cette règle serait fournie par la base de données des fabricants de pièces et permettait au fabricant soit de sous-déléguer encore l'espace soit d'envoyer l'interrogateur directement aux informations EDIFAC dans le système.

Dans cet exemple supposons que le fabricant retourne une règle qui déclare que les trois chiffres suivants DEVRAIENT être utilisés au titre d'une interrogation à leur service afin de trouver une nouvelle règle. Cette nouvelle règle permettrait au fabricant de pièces de sous déléguer l'interrogation sur les usines de pièces pour chaque ligne d'automobiles. Dans notre exemple, le numéro de pièce '01A' note les automobiles de marque Toyota. La règle que le fabricant retourne sous délègue l'interrogation sur un centre de pièces détachées au Japon. Cette règle note aussi que cette règle est terminale et donc le résultat de cette dernière interrogation sera l'information réelle sur la pièce.

6.2 Un service d'identification de documents

Cet exemple est très semblable au précédent car les documents dans ce système peuvent simplement être vus comme les pièces automobiles du dernier exemple. La différence est ici que les informations sur le document sont conservées très près de l'auteur (normalement sur son ordinateur). Il est donc probable que le nombre de délégations peut être très important. Aussi, afin de se garder d'avoir un large espace plat d'auteurs, ceux-ci sont organisés par entreprises et par départements.

Supposons que la chaîne unique d'application dans cet exemple ressemble à ce qui suit :

<entreprise>-<département>-<auteur>:<projet>-<bibliothèque>-<ouvrage>

La spécification d'application ressemblerait à :
- o Chaîne unique d'application : la chaîne d'identification de document donnée ci-dessus.
- o Première règle bien connue : les caractères jusqu'au premier '-' non inclus sont traités comme étant la première clé.
- o Bases de données valides : le répertoire DIS LDAP.
- o Résultat attendu : un enregistrement d'un serveur LDAP contenant des informations bibliographiques sur le document en XML.

La spécification de base de données pour le répertoire DIS LDAP ressemblerait à ceci :
- o Spécification générale : la base de données utilise le service de répertoire LDAP. Chaque serveur LDAP a un enregistrement qui contient la règle de réécriture. Les règles se réfèrent aux autres serveurs LDAP qui utilisent le schéma d'URL LDAP.
- o Procédure de recherche : en utilisant les interrogations LDAP standard, le client demande au serveur LDAP les informations sur la clé.
- o Format de clé : aucune conversion n'est nécessaire.
- o Format de règle : voir la spécification de la règle de réécriture LDAP.
- o Procédure d'insertion de règle : voir les procédures publiées par l'entité qui a autorité sur cette section de l'arborescence du DIS. La première section, l'entreprise, est possédée par l'Agence DIS.

Dans cet exemple, la première recherche est pour la règle de l'entreprise. À ce point, l'entreprise peut diriger le client directement sur quelque grosse base de données au niveau de l'entreprise qui contient le résultat attendu. D'autres entreprises peuvent décentraliser ce processus de façon que les règles continuent de déléguer l'interrogation jusqu'au choix des auteurs de l'environnement de gestion de documents.

7. Considérations pour la sécurité

Le présent document définit simplement l'algorithme DDDS et donc n'implique par lui-même aucun problème de sécurité.

C'est quand cet algorithme est couplé avec une base de données et une application que des considérations de sécurité peuvent apparaître suffisamment pour les énumérer au delà de dire simplement que les points de délégation dynamiques sont un point d'attaque possible.

8. Considérations relatives à l'IANA

Le présent document ne crée aucune exigence pour l'IANA. Les spécifications de base de données et d'application peuvent avoir des exigences considérables mais elles ne peuvent être énumérées ici.

Références

[IEEE] The Institute of Electrical and Electronics Engineers, "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Std 1003.2-1992, ISBN 1-55937-255-9, janvier 1993.

[RFC2141] R. Moats, "Syntaxe des URN", mai 1997.

[RFC2168] R. Daniel, M. Mealling, "Résolution des identifiants de ressource uniformes avec le système des noms de domaines", juin 1997. (*Obsolète, voir* RFC3401, RFC3402, RFC3403, RFC3404) (*MàJ par* RFC2915) *(Exp.)*

[RFC2276] K. Sollins, "Principes d'architecture de la résolution de nom de ressource uniforme", janvier 1998. (*MàJ par* RFC3401) *(Information)*

[RFC2915] M. Mealling, R. Daniel, "Enregistrement de ressource DNS Pointeur d'autorité de dénomination (NAPTR)", septembre 2000. (*Obsolète, voir* RFC3401, RFC3402, RFC3403, RFC3404) *(P.S.)*

[RFC2916] P. Faltstrom, "Numéros E.164 et DNS", septembre 2000. (*Obsolète, voir* RFC3761) *(P.S.)*

[RFC3401] M. Mealling, "Système de découverte dynamique de délégation (DDDS) Partie I : DDDS complet", octobre 2002. *(Info.)*

[RFC3402] M. Mealling, "Système de découverte dynamique de délégation (DDDS) Partie II : l'algorithme", octobre 2002.

[RFC3403] M. Mealling, "Système de découverte dynamique de délégation (DDDS) Partie III : base de données du système de noms de domaines (DNS)", octobre 2002. *(P.S.)*

[RFC3404] M. Mealling, "Système de découverte dynamique de délégation (DDDS) Partie IV : Identifiants de ressource uniformes (URI)", octobre 2002. *(P.S.)*

[RFC3405] M. Mealling, "Système de découverte dynamique de délégation (DDDS) Partie V : Procédures d'allocation URI.ARPA", octobre 2002. (BCP0065)

Adresse de l'auteur

Michael Mealling
VeriSign
21345 Ridgetop Circle
Sterling, VA 20166
USA
mél : michael@neonym.net
URI : http://www.verisignlabs.com

Déclaration complète de droits de reproduction

successeurs ou ayants droit.

Remerciement

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