Groupe de travail Réseau
Request for Comments : 4992
RFC mise à jour : 3981
Catégorie : sur la voie de la normalisation

A. Newton, VeriSign, Inc.
août 2007

Traduction Claude Brière de L'Isle

Traitement XML en parallèle avec tronçons" pour le service d'informations de registre Internet

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata 1009, 2827, 2828, 2829, 2831, 2832, 2833, 2834, 2835, 2836)

Résumé
Le présent document décrit un protocole simple de transfert TCP pour le service d'informations de registre Internet (IRIS, *Internet Registry Information Service*). Les données sont transférées entre clients et serveurs en utilisant des tronçons pour réaliser un traitement en parallèle.

Table des Matières

1. Introduction

En utilisant S-NAPTR [RFC3958], IRIS a la capacité de définir l'utilisation de plusieurs transports d'application (ou protocoles de transfert) pour différents types de services de registre, tous à la discrétion de l'opérateur du serveur. Le protocole de transfert TCP défini dans le présent document est complètement modulaire et peut être utilisé par tous les types de registres.

Ce protocole de transfert définit un tramage simple pour l'envoi de XML dans des tronçons afin que les fragments XML puissent être traités (ou traités en parallèle) avant la réception de l'instancee XML entière. Le présent document appelle cela du traitement XML en parallèle avec des tronçons (XPC) et son utilisation avec IRIS, IRIS-XPC.

XPC est à utiliser avec des interactions de simple demande et réponse entre clients et serveurs. Les clients envoient une série de demandes à un serveur dans des blocs de données. Le serveur va répondre à chaque bloc de données individuel avec un bloc de données correspondant, mais sur la même connexion. Les blocs de données de demande et réponse sont envoyés en utilisant la fonction TCP SEND et reçus en utilisant la fonction TCP RECEIVE.

Le cycle de vie d'une session XPC a les phases suivantes :

1. Un client établit une connexion TCP avec un serveur.

2. Le serveur envoie un bloc de réponse de connexion (CRB, *connection response block*).

3. Le client envoie un bloc de demande (RQB, *request block*). Dans cette demande, le client peut établir un fanion "garder ouvert" qui demande que le serveur garde la session XPC ouverte à la suite de la réponse à cette demande.

4. Le serveur répond avec un bloc de réponse (RSB, *response block*). Dans cette réponse, le serveur peut indiquer au client si la session XPC va ou non être close.

5. Si la session XPC ne va pas se terminer, le cycle de vie se répète alors à partir de l'étape 3.

6. La connexion TCP est close.

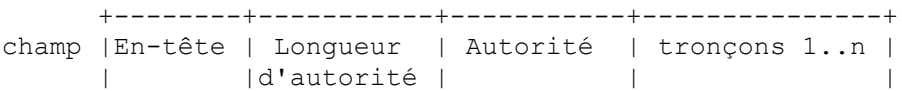
2. Terminologie du document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Les champs d'octets avec des valeurs numériques sont donnés conformément aux conventions de la [RFC1166] : le bit le plus à gauche du champ est le bit de poids fort ; quand une quantité multi octets est transmise, l'octet de poids fort est transmis en premier. Les bits qui ont une signification de fanion dans un octet sont numérotés en accord avec les conventions de la [RFC1166] : le bit 0 est le bit de poids fort et le bit 7 est le bit de moindre poids. Quand un diagramme décrit un groupe d'octets, l'ordre de transmission des octets commence à partir de la gauche.

3. Bloc de demande (RQB)

Le format du bloc de demande (RQB, *request block*) est comme suit :



Bloc de demande

Ces champs ont la signification suivante :

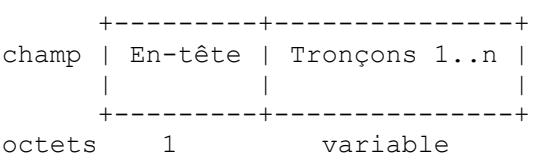
- o En-tête - comme décrit à la Section 5.
- o Longueur d'autorité - longueur du champ Autorité dans ce bloc de demande.
- o Autorité - chaîne d'octets qui décrit l'autorité contre laquelle cette demande doit être exécutée. Voir dans la [RFC3981] la définition et la description d'une autorité. Le nombre d'octets dans cette chaîne DOIT être ni plus ni moins que le nombre spécifié par Longueur d'autorité.
- o tronçons 1..n - données de la demande réparties en des tronçons (voir la Section 6).

4. Blocs de réponse

Il y a deux types de blocs utilisés par un serveur pour répondre à un client. Le premier type est un bloc de réponse (RSB) défini au paragraphe 4.1. Il est utilisé par un serveur pour répondre aux bloc de demande (RQB, *request block*). Le second type est une version spécialisée d'un bloc de réponse appelé bloc de réponse de connexion (CRB, *connection response block*) défini au paragraphe 4.2. Il est envoyé par un serveur à un client quand une connexion est établie pour initier la négociation du protocole. Conceptuellement, un CRB est un type de RSB ; ils partagent le même format, mais un CRB est contraint à ne porter que des informations spécifiques et n'est envoyé qu'au début du cycle de vie de la session.

4.1 Bloc de réponse (RSB)

Le format du bloc de réponse (RSB) est comme suit :



Bloc de réponse

Ces champs ont la signification suivante :

- o En-tête - comme décrit à la Section 5.
- o Tronçons 1..n - données de réponse coupées en tronçons (Section 6).

Les serveurs NE DEVRAIENT PAS envoyer un RSB à un client avant d'avoir reçu le RQB entier. Les serveurs qui commencent par envoyer un RSB avant la réception du RQB entier doivent considérer que les clients ne sont pas supposés commencer à traiter le RSB avant qu'ils aient fini d'envoyer le RQB, et que le RSB peut remplir les mémoires tampon TCP du client.

4.2 Bloc de réponse de connexion (CRB)

Un bloc de réponse de connexion (CRB, *connection response block*) est un bloc de réponse envoyé par un serveur à un client en réponse à l'initialisation d'une session par le client. Un bloc de réponse de connexion a le même format qu'un bloc de réponse (RSB) (paragraphe 4.1). La seule différence est qu'il est contraint d'une des deux façons suivantes :

1. Il contient un seul tronçon (voir la Section 6) contenant les informations de version (voir le paragraphe 6.2) et le fanion garder ouvert (KO, *keep-open*) dans l'en-tête de bloc (voir la Section 5) a une valeur de 1 (signifiant que la connexion n'est pas en train de fermer). Les serveurs DOIVENT utiliser ce type de CRB pour indiquer la disponibilité du service.

2. Il contient un seul tronçon (voir la Section 6) contenant une erreur système (voir "erreur système" au paragraphe 6.4) et le fanion garder ouvert (KO) dans l'en-tête de bloc (voir à la Section 5) a une valeur de 0 (signifiant que le serveur va clore la connexion immédiatement après l'envoi du CRB). Les serveurs DOIVENT utiliser ce type de CRB quand ils peuvent accepter des connexions mais ne peuvent pas traiter les demandes.

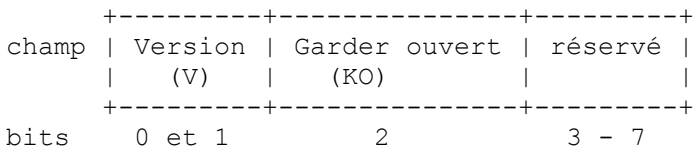
5. En-tête de bloc

Chaque bloc de données commence par un en-tête d'un octet appelé l'en-tête de bloc. Cet en-tête a le même format pour les deux blocs de données de demande et de réponse, bien que certains des bits de l'en-tête n'aient de signification que dans un type de bloc de données. Les bits sont ordonnés en accord avec la convention donnée dans la [RFC1166], où le bit 0 est le bit de poids fort et le bit 7 est le bit de moindre poids. Chaque bit dans l'en-tête de bloc a la signification suivante :

o bits 0 et 1 - version (champ V) - si il sont à 0 (les deux bits à zéro) la version du protocole est celle définie dans le présent document. Autrement, le reste des bits dans l'en-tête et le bloc peut être interprété comme une autre version. Si un serveur reçoit une demande pour une version qu'il ne prend pas encharge, il DEVRAIT suivre le comportement décrit à la Section 8.

o bit 2 - fanion Garder ouvert (KO) - ce fanion est utilisé par un client pour demander qu'une connexion reste ouverte et par un serveur pour indiquer qu'une connexion va rester ouverte, selon le type de bloc. Dans un bloc de demande (RQB) une valeur de 1 indique qu'un client demande que le serveur ne ferme pas la session TCP, et une valeur de 0 indique que le client va s'attendre à ce que son serveur close la session TCP immédiatement après l'envoi de la réponse correspondante. Dans un bloc de réponse (RSB) ou un bloc de réponse de connexion (CRB) une valeur de 1 indique que le serveur s'attend à ce que le client garde la session TCP ouverte pour que le serveur reçoive une autre demande, et une valeur de 0 indique que le serveur s'attend à ce que le client ferme la session TCP immédiatement après ce bloc.

o bits 3, 4, 5, 6, et 7 - réservés - ces bits DOIVENT être à 0. Si un serveur reçoit une demande dans laquelle un de ces bits est réglé à 1 et si le serveur ne comprend pas l'objet de cette valeur, le serveur DEVRAIT suivre le comportement décrit à la Section 8.



En-tête de bloc

6. Tronçons

Les blocs de demande et réponse coupent les données de demande et réponse XML en tronçons. Les blocs de demande et réponse DOIVENT toujours avoir un minimum de un tronçon. Chaque tronçon a un descripteur de un octet. Le premier bit du descripteur détermine si le tronçon est le dernier du bloc.

Les bits de l'octet descripteur de tronçon sont ordonnés en accord avec la convention de la [RFC1166], où le bit 0 est le bit de poids fort et le bit 7 est le bit de moindre poids. Les bits de l'octet descripteur de tronçon ont la signification suivante :

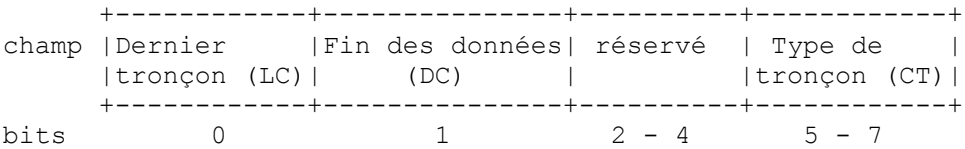
o bit 0 -fanion Dernier tronçon (LC, *last chunk*) - si il est réglé à 1, ce tronçon est le dernier du bloc.

o bit 1 - fanion Fin des données (DC, *data complete*) - si il est réglé à 1, les données de ce tronçon représentent la fin des données pour ce type de tronçon. Si ce bit n'est jamais réglé à 1 dans un descripteur de tronçon pour des tronçons du même type dans un bloc, les clients et serveurs NE DOIVENT PAS supposer que les données vont continuer dans un autre bloc. Si le bloc passe d'un type de tronçon à un autre sans signaler l'achèvement des données, les clients et serveurs

DOIVENT supposer que les données restantes ne vont pas être envoyées dans un tronçon restant.

- bits 2, 3, et 4 - réservés - Ils DOIVENT être 0.

- bits 5, 6, et 7 - champ Type de tronçon (CT, *chunk type*) - détermine le type des données portées dans le tronçon. Ce sont des valeurs binaires pour les types de tronçons :
  * 000 - pas de données ou type "nd" (voir au paragraphe 6.1)
  * 001 - informations de version ou type "vi" (voir au paragraphe 6.2)
  * 010 - informations de taill ou type "si" (voir au paragraphe 6.3)
  * 011 - autres informations ou type "oi" (voir au paragraphe 6.4)
  * 100 - données SASL (authentification simple et couche de sécurité) ou type "sd" (voir au paragraphe 6.5)
  * 101 - informations de succès d'authentification ou type "as" (voir au paragraphe 6.6)
  * 110 - informations d'échec d'authentification ou type "af" (voir au paragraphe 6.7)
  * 111 - données d'application ou type "ad" (voir au paragraphe 6.8)



Descripteur de tronçon

Un bloc PEUT avoir de multiples types de tronçons, mais tous les tronçons de même type DOIVENT être contigus dans un bloc et DOIVENT être ordonnés dans le bloc selon l'ordre dans lequel leurs données sont à interpréter. Les tronçons contigus doivent être ordonnés par type au sein d'un bloc de la façon suivante :

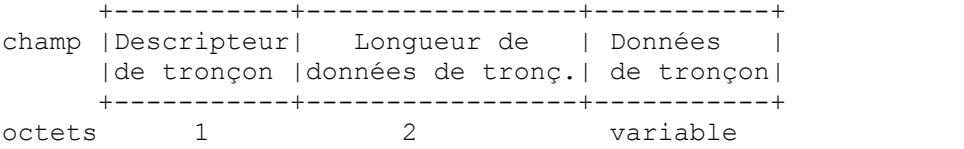
1. tronçons relatifs à l'authentification - des tronçons de données SASL (type 100), des tronçons d'informations de succès d'authentification (type 101), ou des tronçons d'informations d'échec d'authentification (type 110) mais pas de plus d'un type. Durant l'établissement des mécanismes de sécurité qui utilisent ces tronçons, les clients NE DOIVENT PAS envoyer d'autre demande tant qu'ils n'ont pas reçu de tronçon de succès ou d'échec d'authentification.

2. tronçons de données - soit des tronçons pas de données (type 000) soit des tronçons données d'application (type 111), mais pas les deux.

3. tronçons d'informations - soit informations de version (type 001) soit autres informations (type 011), mais pas les deux.

Un bloc DOIT avoir au moins un type de tronçon ci-dessus.

Le format pour un tronçon est comme suit :



Tronçon

Ces champs ont la signification suivante :
- Descripteur de tronçon - comme décrit plus haut.
- Longueur des données de tronçon - longueur des données du tronçon.
- Données de tronçon - les données dans le tronçon.

6.1 Types Pas de données

Les serveurs et clients DOIVENT ignorer les données dans les types de tronçons étiquetés "no data". Il n'est pas exigé que ces types de tronçons soient de longueur zéro. Un client PEUT envoyer "no data" à un serveur, et le serveur DOIT répondre avec un tronçon de même type ou de type autres informations (paragraphe 6.4).

6.2 Types Information de version

Les tronçons de ce type contiennent de l'XML conforme au schéma spécifié dans la [RFC4991] et DOIVENT avoir l'élément <versions> comme élément racine, sauf si ils sont envoyés par le client pour solliciter des informations de version ; les tronçons de client à serveur peuvent être de longueur zéro,comme précisé ci-dessous.

Dans le contexte de IRIS-XPC, les identifiants de protocole pour ces éléments sont comme suit :
o <transferProtocol> - la valeuR "iris.xpc1" pour indiquer le protocole spécifié dans le présent document.
o <application> - l'identifiant d'espace de noms XML pour IRIS [RFC3981].
o <dataModel> - l'identifiant d'espace de noms XML pour les registres IRIS.

Dans le contexte de IRIS-XPC, les identifiants de mécanisme d'authentification sont les noms de mécanisme SASL qui se trouvent dans le registre des noms de mécanismes SASL de l'IANA définis par la [RFC4422].

Le présent document ne définit pas d'identifiants d'extension.

Les clients PEUVENT envoyer un bloc avec ce type de tronçon à un serveur. Ces tronçons DEVRAIENT être de longueur zéro, et les serveurs DOIVENT ignorer toutes les données qui sont dedans. Quand un serveur reçoit un tronçon de ce type, il DOIT répondre avec un tronçon de ce type. Cet échange permet à un client d'interroger les informations de version d'un serveur.

Les tailles d'octets pour les attributs "requestSizeOctets" et "responseSizeOctets" de l'élément <tranferProtocol> sont définies au paragraphe 6.3.

6.3 Types Informations de taille

Les tronçons de ce type contiennent de l'XML conforme au schéma spécifié dans la [RFC4991] et DOIVENT avoir l'élément <size> comme élément racine.

Les comptes d'octets fournis par ces informations sont définis comme la somme du compte de toutes les données de tronçons d'un type de tronçon particulier. Par exemple, si une instance XML est coupée en des tronçons de 20, 30, et 40 octets, le compte d'octet va être 90 (20 + 30 + 40).

Les clients NE DOIVENT PAS envoyer des tronçons de ce type, et les serveurs PEUVENT clore une session en utilisant la procédure de la Section 8 si un tronçon de ce type est reçu.

6.4 Types Autres informations

Les tronçons de ce type contiennent de l'XML conforme au schéma spécifié dans la [RFC4991] et DOIVENT avoir l'élément <other> comme élément racine.

Les valeurs pour l'attribut de "type" de <other> sont comme suit :
'block-error' - indique qu'il y a eu une erreur de de codage d'un bloc. Les serveurs DEVRAIENT envoyer une erreur de bloc dans les cas suivants :

1. Quand un bloc de demande est reçu qui contient un tronçon de ce type.
2. Quand un bloc de demande est reçu qui contient des informations de succès d'authentification (voir le paragraphe 6.6) ou d'échec d'authentification (voir le paragraphe 6.7).
3. Quand un bloc de demande est reçu contenant des informations de taille (voir le paragraphe 6.3).
4. Quand des bits réservés dans le bloc de demande sont à 1.
5. Quand un bloc n'a pas été reçu entièrement et que la session TCP a été inactive pendant une période spécifiée (c'est-à-dire qu'un bloc de données a été reçu mais qu'aucun tronçon de terminaison n'a été reçu pour le bloc de données). Deux minutes est RECOMMANDÉ pour cette valeur de temporisation. Noter qu'il y a une différence entre une condition d'inactivité due à la réception incomplète d'un bloc de données et une condition d'inactivité entre des transactions de demande/réponse associées au maintien de l'ouverture de la session. Pour cette dernière, voir la Section 7.

'data-error' - indique qu'il y a eu une erreur d'analyse des données dans des tronçons contenant une application ou des données SASL (par exemple, le XML n'est pas valide dans les données d'application).

'system-error' - indique que le receveur ne peut pas traiter la demande à cause d'une condition sans relation avec ce

protocole. Les serveurs DEVRAIENT envoyer une erreur système quand ils sont capables de répondre aux demandes mais pas capables de les traiter.

'authority-error' - indique que l'autorité prévue spécifiée dans la demande correspondante n'est pas servie par le receveur. Les serveurs DEVRAIENT envoyer une erreur d'autorité quand ils reçoivent une demande dirigée sur une autorité autre que celle qu'ils servent.

'idle-timeout' - indique qu'une session XPC a été inactive pendant trop longtemps. L'usage de cette valeur est défini à la Section 7. Noter qu'il y a une différence entre une condition inactive due à la réception incomplète d'un bloc de données et une condition d'inactivité entre des transactions de demande/réponse associées au maintien de l'ouverture de la session. Pour la première, voir à 'block-error' ci-dessus.

Les clients NE DOIVENT PAS envoyer des tronçons de ce type, et les serveurs PEUVENT clore une session en utilisant la procédure de la Section 8 si un tronçon de ce type est reçu.

6.5 Types SASL

Le type de tronçon SASL permet aux clients et serveurs d'échanger des données SASL.

Le format pour les données de ce type de tronçon est comme suit :

```
        +-----------+-----------+------------+-----------+
champ   | Longueur  |  Nom du   |  Longueur  | Données du|
        | du nom de | mécanisme |des données | mécanisme |
        | mécanisme |           |du mécanisme|           |
        +-----------+-----------+------------+-----------+
octets      1         variable       2         variable
```

Authentification SASL

Ces champs ont la signification suivante :
- o Longueur du nom de mécanisme - longueur du nom du mécanisme SASL.
- o Nom du mécanisme - nom du mécanisme SASL tel qu'enregistré dans le registre IANA des mécanismes SASL défini par la [RFC4422].
- o Longueur des données de mécanisme - longueur des données SASL, ou la valeur spéciale 65535 (pour indiquer l'absence d'une réponse initiale SASL -- voir ci-dessous).
- o Données du mécanisme - données utilisées pour SASL.

Ces champs NE DOIVENT PAS s'étendre sur plusieurs tronçons. Donc, on devrait noter qu'une longueur de données de mécanisme qui excède la longueur du tronçon moins la longueur du nom du mécanisme moins trois est une erreur.

Selon la nature du mécanisme SASL utilisé, les données SASL sont envoyées des clients aux serveurs et des serveurs aux clients et peuvent exiger plusieurs transactions de demande/réponse pour être menées à bien, une fois qu'un échange SASL est achevé et qu'un serveur peut déterminer l'état d'authentification, le serveur DOIT envoyer des informations de réussite d'authentification (voir le paragraphe 6.6) ou d'échec d'authentification (voir le paragraphe 6.7).

Quand elle est utilisée comme une réponse de défi initial pour les mécanismes SASL qui prennent en charge une telle caractéristique, la longueur des données du mécanisme peut être réglée à une valeur décimale de 65 535 pour indiquer l'absence d'une réponse initiale. Une valeur de 0 indique une réponse initiale vide.

6.6 Types d'informations de succès d'authentification

Les tronçons de ce type contiennent de l'XML conforme au schéma spécifié dans la [RFC4991] et DOIVENT avoir l'élément <authenticationSuccess> comme élément racine.

Ce type de tronçon n'est envoyé que d'un serveur à un client. Si un client l'envoie à un serveur, il en résultera une erreur de bloc (voir "block-error" au paragraphe 6.4). L'usage de ce type de tronçon est défini au paragraphe 6.5. Un serveur PEUT clore une session à cause de la réception de ce type de tronçon en utilisant la procédure de la Section 8.

Les mécanismes SASL peuvent utiliser l'élément <data> pour renvoyer des données binaires arbitraires comme binaire base 64. L'absence de cet élément indique l'absence de telles données, tandis que la présence de l'élément sans contenu indique un enssmble de données vide.

6.7 Types d'informations d'échec d'authentification

Les tronçons de ce type contiennent de l'XML conforme au schéma spécifié dans la [RFC4991] et DOIVENT avoir l'élément <authenticationFailure> comme élément racine.

Ce type de tronçon n'est envoyé que d'un serveur à un client. Si un client l'envoie à un serveur, il va en résulter une erreur de bloc (voir "block-error" au paragraphe 6.4). L'usage de ce type de tronçon est défini au paragraphe 6.5. Un serveur PEUT clore une session à cause de la réception de ce type de tronçon en utilisant la procédure de la Section 8.

6.8. Types Données d'application

Ces tronçons contiennent des données d'application. Pour IRIS, ce sont des instances XML IRIS [RFC3981].

7. Sessions inactives

Si un serveur a besoin de clore une connexion parce qu'elle est inactive, il DEVRAIT faire ce qui suit :

1. Envoyer un bloc de réponse non sollicitée contenant une erreur de fin de temporisation pour inactivité (voir "idle-timeout" au paragraphe 6.4) avec le fanion Garder ouvert (KO, *keep-open*) dans l'en-tête de bloc (Section 5) réglé à la valeur de 0.

2. Clore la connexion TCP.

8. Fermeture de session due à une erreur

Si un serveur va clore une session à cause d'une erreur, il DEVRAIT faire ce qui suit :

1. Envoyer un bloc de réponse contenant une erreur de bloc ou de données (voir au paragraphe 6.4) ou des informations de version (voir au paragraphe 6.2) avec le fanion KO dans l'en-tête de bloc (Section 5) réglé à 0.

2. Clore la connexion TCP.

9. Utilisation sur TLS

XPC peut être tunnelé sur TLS [RFC4346] en établissant une session TLS immédiatement après l'ouverture d'une session TCP et avant l'envoi de blocs. Ce type de session est appelé XPCS.

Quand TLS est utilisé, une convention doit être établie pour permettre à un client d'authentifier la validité d'un serveur. XPCS utilise la même convention que décrite par IRIS-BEEP [RFC3983].

TLS permet l'authentification et la confidentialité.

Ceux qui mettent en œuvre la présente spécification devraient noter que alors que XPC et XPCS ont des noms dc schéma d'URI et dcs étiquettes de protocole d'application S-NAPTR distincts, tous deux sont identifiés avec la même valeur de <transferProtocol> dans les informations de version des tronçons (voir au paragraphe 6.2).

10. Mise à jour de la RFC 3981

Le praragraphe 7.2 de la [RFC3981] (IRIS-CORE) déclare que IRIS-BEEP [RFC3983] est le transport par défaut pour

IRIS. Le présent document révise la RFC 3981 et spécifie IRIS-XPC comme transport par défaut pour IRIS. L'enregistrement de l'accès TCP bien connu est spécifié au paragraphe 13.5.

11. Définitions de transposition de transport IRIS

Cette Section fait la liste des définitions requises par IRIS [RFC3981] pour les transpositions de transport.

11.1 Schéma d'URI

Voir les paragraphes 13.1 et 13.2.

11.2 Étiquette de protocole d'application

Voir les paragraphes 13.3 et 13.4.

12. Considérations d'internationalisation

Les processeurs XML ont l'obligation de reconnaître les deux codages UTF-8 et UTF-16 [Unicode]. L'utilisation du XML défini par la [RFC4991] NE DOIT PAS utiliser d'autres codages de caractères que UTF-8 ou UTF-16.

13. Considérations relatives à l'IANA

13.1 Enregistrement de schéma d'URI XPC

Nom de schéma d'URL : iris.xpc
Statut : permanent
Syntaxe de schéma d'URL : définie dans la [RFC3981].
Considérations de codage de caractères : comme défini dans la [RFC3986].
Utilisation prévue : identifie l'XML IRIS en utilisant des tronçons sur TCP
Applications qui utilisent ce schéma : définies dans IRIS [RFC3981].
Considérations d'interopérabilité : non applicable
Considérations de sécurité : définies à la Section 14.
Publications pertinentes : IRIS [RFC3981].
Informations de contact : Andrew Newton <andy@hxr.us>
Auteur/contrôleur des changements : IESG

13.2 Enregistrement de schéma d'URI XPCS

Nom de schéma d'URL : iris.xpcs
Statut : permanent
Syntaxe de schéma d'URL : définie dans la [RFC3981].
Considérations de codage de caractères : comme défini dans la [RFC3986].
Utilisation prévue : identifie l'XML IRIS en utilisant des tronçons sur TCP
Applications qui utilisent ce schéma : définies dans IRIS [RFC3981].
Considérations d'interopérabilité : non applicable
Considérations de sécurité : définies à la Section 14.
Publications pertinentes : IRIS [RFC3981].
Informations de contact : Andrew Newton <andy@hxr.us>
Auteur/contrôleur des changements : IESG

13.3 Enregistrement XPC S-NAPTR

Étiquette de protocole d'application (voir la [RFC3958]) : iris.xpc
Utilisation prévue : identifie un serveur IRIS qui utilise XPC

Applications qui utilisent ce schéma : définies dans IRIS [RFC3981].
Considérations d'interopérabilité : non applicable
Considérations de sécurité : définies à la Section 14.
Publications pertinentes : IRIS [RFC3981].
Informations de contact : Andrew Newton <andy@hxr.us>
Auteur/contrôleur des changements : IESG

13.4 Enregistrement XPCS S-NAPTR

Étiquette de protocole d'application (voir la [RFC3958]) : iris.xpcs
Utilisation prévue : identifie un serveur IRIS qui utilise XPCS sécurisé
Applications qui utilisent ce schéma : définies dans IRIS [RFC3981].
Considérations d'interopérabilité : non applicable
Considérations de sécurité : définies à la Section 14.
Publications pertinentes : IRIS [RFC3981].
Informations de contact : Andrew Newton <andy@hxr.us>
Auteur/contrôleur des changements : IESG

13.5 Enregistrement d'accès TCP bien connu pour XPC

Nom de protocole : TCP
Numéro d'accès TCP : 713
Formats, types, opcodes, et séquences de message : définis à la Section 3, et aux paragraphes 4.1 et 4.2,
Fonctions : définies dans IRIS [RFC3981].
Utilisation de diffusion/diffusion groupée : aucune
Nom proposé : IRIS sur XPC
Nom abrégé : iris.xpc
Informations de contact : Andrew Newton <andy@hxr.us>

13.6 Enregistrement d'accès TCP bien connu pour XPCS

Nom de protocole : TCP
Numéro d'accès TCP : 714
Formats, types, opcodes, et séquences de message : définis aux paragraphes 4.1 et 4.2, et aux Sections 3 et 9.
Fonctions : définies dans IRIS [RFC3981].
Utilisation de diffusion/diffusion groupée : aucune
Nom proposé : IRIS sur XPCS
Nom abrégé : iris.xpcs
Informations de contact : Andrew Newton <andy@hxr.us>

14. Considérations sur la sécurité

Les mises en œuvre devraient avoir pleinement consciense des considérations sur la sécurité de IRIS [RFC3981] et de TLS [RFC4346]. Pour ce qui concerne l'authentification du serveur pour l'utilisation de TLS, voir la Section 6 de IRIS-BEEP [RFC3983].

14.1 Mécanismes de sécurité

Les clients DEVRAIENT être prêts à utiliser les mécanismes de sécurité suivants de la manière suivante :
- SASL/DIGEST-MD5 - pour l'authentification de l'utilisateur sans chiffrement de session.
- SASL/OTP - pour l'authentification de l'utilisateur sans chiffrement de session.
- TLS avec le chiffrement TLS_RSA_WITH_3DES_EDE_CBC_SHA - pour le chiffrement.
- TLS avec le chiffrement TLS_RSA_WITH_3DES_EDE_CBC_SHA avec des certificats côté client - pour le chiffrement et l'authentification de l'utilisateur.
- TLS avec le chiffrement TLS_RSA_WITH_AES_128_CBC_SHA - pour le chiffrement. Voir la [RFC3268].
- TLS avec le chiffrement TLS_RSA_WITH_AES_128_CBC_SHA avec des certificats côté client - pour le chiffrement et l'authentification de l'utilisateur. Voir la [RFC3268].

- o TLS avec le chiffrement TLS_RSA_WITH_AES_256_CBC_SHA - pour le chiffrement. Voir la [RFC3268].
- o TLS avec le chiffrement TLS_RSA_WITH_AES_256_CBC_SHA avec des certificats côté client - pour le chiffrement et l'authentification de l'utilisateur. Voir la [RFC3268].

L'accès de client anonyme DEVRAIT être considéré dans une des deux méthodes suivantes :
1. Quand aucune authentification n'a été utilisée.
2. En utilisant le profil SASL anonyme : SASL/ANONYMOUS

Comme spécifié par SASL/PLAIN, les clients NE DOIVENT PAS utiliser le mécanisme SASL/PLAIN sans chiffrer d'abord la session TCP (par exemple, avec TLS). Les clients DOIVENT mettre en œuvre SASL/PLAIN et TLS en utilisant le chiffrement TLS_RSA_WITH_3DES_EDE_CBC_SHA.

14.2 Conformité à SASL

La liste suivante détaille la conformité de IRIS-XPC utilisé avec SASL, comme spécifié à la Section 4 de la [RFC4422].

1. Le nom de service SASL à utiliser par IRIS-XPC est "iris-xpc".

2. Le paragraphe 6.2 décrit la facilité de négociation utilisée pour déterminer les mécanismes de sécurité disponibles. Cette facilité peut être utilisée à la fois avant l'initiation des échanges SASL et après l'installation des mécanismes de sécurité.

3.
   a) Le paragraphe 6.5 décrit le mécanisme pour initier les échanges d'authentification.
   b) Le paragraphe 6.5 décrit le mécanisme pour transférer des défis du serveur et les réponses du client.
   c) Les paragraphes 6.6 et 6.7 décrivent les mécanismes pour indiquer le résultat d'un échange d'authentification. Le paragraphe 6.6 décrit comment des données supplémentaires peuvent être portées avec ce message.

4. Les chaînes d'identité d'autorisation non vides utilisées dans IRIS-XPC DOIVENT être normalisées en accord avec la [RFC4013]. La sémantique des chaînes d'identité d'autorisation non vides dépend du serveur, et les clients DOIVENT utiliser les valeurs pour ces chaînes telles que données par la configuration ou l'utilisateur.

5. Les clients ou serveurs qui souhaitent interrompre un échange d'authentification en cours DOIVENT clore la connexion.

6. Après la négociation de nouvelles couches de sécurité, elles entrent en vigueur sur le premier octet qui suit le tronçon de succès d'authentification (as, *authentication success*) (paragraphe 6.6) envoyé par le serveur et sur le premier octet envoyé après réception du tronçon as envoyé par le client.

7. IRIS-XPC peut être utilisé avec TLS et SASL. Quand il est utilisé en combinaison, TLS DOIT toujours être appliqué avant tout mécanisme SASL.

8. IRIS-XPC ne prend pas en charge plusieurs authentifications SASL. Cependant, si TLS est utilisé en combinaison avec SASL, l'authentification TLS DOIT se produire avant toute authentification SASL.

15. Références

15.1 Références normatives

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

[RFC3268] P. Chown, "Suites de chiffrement de la norme de chiffrement évolué (AES) pour la sécurité de la couche Transport (TLS)", juin 2002. (*Obsolète, voir* RFC5246) *(P.S.)*

[RFC3958] L. Daigle, A. Newton, "Localisation de service d'application fondée sur le domaine avec les enregistrements de ressource de SRV et le service de recherche dynamique de délégation (DDDS)", janvier 2005. *(P.S.)*

[RFC3981] A. Newton, M. Sanz, "IRIS : Protocole central du service d'information des registres Internet (IRIS)", janvier 2005. (*MàJ par* RFC4992) *(P.S.)*

[RFC3983] A. Newton, M. Sanz, "Utilisation du service d'information des registres Internet (IRIS) sur le protocole extensible d'échange de blocs (BEEP)", janvier 2005. *(P.S.)*

[RFC3986] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiant de ressource uniforme (URI) : Syntaxe générique", STD 66, janvier 2005. *(P.S. ; MàJ par* RFC8820)

[RFC4013] K. Zeilenga, "SASLprep : Profil Stringprep pour les noms d'utilisateur et mots de passe", février 2005.

[RFC4346] T. Dierks et E. Rescorla, "Protocole de sécurité de la couche Transport (TLS) version 1.1", avril 2006. *(Remplace* RFC2246 *; Remplacée par* RFC5246 *; MàJ par* RFC4366, 4680, 4681, 5746, 6176, 7465, 7507, 7919)

[RFC4422] A. Melnikov et K. Zeilenga, éd, "Authentification simple et couche de sécurité (SASL)", juin 2006. *(P.S.)*

[RFC4991] A. Newton, "Schéma commun pour les protocoles de transfert de service d'information de registre Internet", août 2007. *(P.S.)*

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 3", ISBN 0-201-61633-5, 2000.

15.2 Références pour information

[RFC1166] S. Kirkpatrick, M. Stahl et M. Rekhter, "Numéros de l'Internet", octobre 1990. *(Obsolète, voir 1700)*

Appendice A. Exemples

Cette Section donne des exemples de sessions IRIS-XPC. Les lignes qui commencent pas un "C:" notent des données envoyées par le client au serveur, et les lignes qui commencent pas un "S:" notent des données envoyées par le serveur au client. À la suite du "C:" ou "S:", la ligne contient soit des valeurs d'octet en notation hexadécimale avec des commentaires, soit des fragments XML. Aucune ligne ne contient à la fois des valeurs d'octets avec des commentaires et des fragments XML. Les commentaires sont contenus entre des parenthèses.

On devrait aussi noter que les valeurs de fanion de"oui" et "non" reflètent les valeurs binaires respectivement de 1 et 0.

L'exemple suivant montre un client IRIS qui produit deux demandes dans une session XPC. Dans la première demande, le client demande des informations d'état pour "exemple.com". Cette demande et sa réponse sont transférées avec un tronçon. Dans la seconde demande, le client demande des informations d'état pour "milo.exemple.com", "felix.exemple.com", et "hobbes.exemple.com". Cette demande et sa réponse sont transférées avec trois tronçons.

```
S: (bloc de réponse de connexion)
S: 0x20      (en-tête de bloc : V=0,KO=oui)
S: (tronçon 1)
S: 0xC1      (LC=oui,DC=oui,CT=vi)
S: 0x01 0xBF (longueur de tronçon=447)
S: (Informations de version )
S: <?xml version="1.0"?>
S: <versions xmlns="urn:ietf:params:xml:ns:iris-transport">
S:   <transferProtocol protocolId="iris.xpc1"
S:     authenticationIds="PLAIN EXTERNAL">
S:     <application protocolId="urn:ietf:params:xml:ns:iris1"
S:       extensionIds="http://exemple.com/SIMPLEBAG">
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dchk1"/>
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dreg1"/>
S:     </application>
S:   </transferProtocol>
S: </versions>

C: (Bloc de demande)
```

```
C: 0x20      (en-tête de bloc : V=0,KO=oui)
C: 0x0B      (longueur d'autorité=11)
C:           (authority="exemple.com")
C: 0x65 0x78 0x61 0x6D 0x70 0x6C 0x65 0x23 0x63 0x6F 0x6D
C: (tronçon 1)
C: 0xC7      (LC=oui,DC=oui,CT=ad)
C: 0x01 0x53 (longueur de tronçon=339)
C: (Demande XML IRIS)
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:   xsi:schemaLocation="urn:ietf:params:xml:ns:iris1 iris.xsd" >
C:   <searchSet>
C:     <lookupEntity
C:       registryType="urn:ietf:params:xml:ns:dchk1"
C:       entityClass="domain-name"
C:       entityName="exemple.com" />
C:   </searchSet>
C: </request>

S:  (Bloc de réponse)
S: 0x20      (en-tête de bloc: V=0,KO=oui)
S: (tronçon 1)
S: 0xC7      (LC=oui,DC=oui,CT=ad)
S: 0x01 0xE0 (longueur de tronçon=480)
S: (Réponse XML IRIS)
S: <iris:response xmlns:iris="urn:ietf:params:xml:ns:iris1">
S:   <iris:resultSet>
S:     <iris:answer>
S:       <domain authority="exemple.com" registryType="dchk1"
S:         entityClass="domain-name" entityName="exemple.com-1"
S:         temporaryReference="vrai"
S:         xmlns="urn:ietf:params:xml:ns:dchk1">
S:         <domainName>exemple.com</domainName>
S:         <status>
S:           <assignedAndActive/>
S:         </status>
S:       </domain>
S:     </iris:answer>
S:   </iris:resultSet>
S: </iris:response>

C: (Bloc de demande)
C: 0x20      (en-tête de bloc : V=0,KO=oui)
C: 0x0B      (longueur d'autorité = 11)
C:           (authority="exemple.com")
C: 0x65 0x78 0x61 0x6D 0x70 0x6C 0x65 0x23 0x63 0x6F 0x6D
C: (tronçon 1)
C: 0x07      (LC=no,DC=no,CT=ad)
C: 0x01 0x4E (longueur de tronçon = 339)
C:  (Demande XML IRIS)
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:  xsi:schemaLocation="urn:ietf:params:xml:ns:iris1 iris.xsd" >
C:   <searchSet>
C:     <lookupEntity
C:       registryType="urn:ietf:params:xml:ns:dchk1"
C:       entityClass="domain-name"
C:       entityName="milo.exemple.com" />
C:  </searchSet>
C:  (tronçon 2)
C: 0x07      (LC=non,DC=non,CT=ad)
C: 0x00 0xA9 (longueur de tronçon=169)
```

```
C: (Demande XML IRIS)
C:  <searchSet>
C:    <lookupEntity
C:      registryType="urn:ietf:params:xml:ns:dchk1"
C:      entityClass="domain-name"
C:      entityName="felix.exemple.com" />
C:  </searchSet>
C:   (tronçon 3)
C: 0xC7      (LC=oui,DC=oui,CT=ad)
C: 0x00 0xB5 (longueur de tronçon=181)
C:   (Demande XML IRIS)
C:  <searchSet>
C:    <lookupEntity
C:      registryType="urn:ietf:params:xml:ns:dchk1"
C:      entityClass="domain-name"
C:      entityName="hobbes.exemple.com" />
C:  </searchSet>
C:</request>

S: (Bloc de réponse)
S: 0x00      (en-tête de bloc : V=0,KO=no)
S: (tronçon 1)
S: 0x07      (LC=non,DC=non,CT=ad)
S: 0x01 0xDA (longueur de tronçon=474)
S:   (Réponse XML IRIS)
S: <iris:response xmlns:iris="urn:ietf:params:xml:ns:iris1">
S:   <iris:resultSet>
S:     <iris:answer>
S:       <domain authority="exemple.com" registryType="dchk1"
S:         entityClass="domain-name" entityName="milo.exemple.com-1"
S:         temporaryReference="vrai"
S:         xmlns="urn:ietf:params:xml:ns:dchk1">
S:         <domainName>milo.exemple.com</domainName>
S:         <status>
S:           <assignedAndActive/>
S:         </status>
S:       </domain>
S:     </iris:answer>
S:   </iris:resultSet>
S:  (tronçon 2)
S: 0x07      (LC=non,DC=non,CT=ad)
S: 0x01 0xA2 (longueur de tronçon=418)
S: (Réponse XML IRIS
S:  <iris:resultSet>
S:    <iris:answer>
S:      <domain authority="exemple.com" registryType="dchk1"
S:        entityClass="domain-name" entityName="felix.exemple.com-1"
S:        temporaryReference="vrai"
S:        xmlns="urn:ietf:params:xml:ns:dchk1">
S:        <domainName>felix.exemple.com</domainName>
S:        <status>
S:          <assignedAndActive/>
S:        </status>
S:      </domain>
S:    </iris:answer>
S:  </iris:resultSet>
S: (tronçon 3)
S: 0xC7      (LC=oui,DC=oui,CT=ad)
S: 0x01 0xB5 (longueur de tronçon=437)
S: (Réponse XML IRIS)
S:  <iris:resultSet>
```

```
S:    <iris:answer>
S:      <domain authority="exemple.com" registryType="dchk1"
S:        entityClass="domain-name"
S:  entityName="hobbes.exemple.com-1"
S:        temporaryReference="vrai"
S:        xmlns="urn:ietf:params:xml:ns:dchk1">
S:        <domainName>hobbes.exemple.com</domainName>
S:        <status>
S:          <assignedAndActive/>
S:        </status>
S:      </domain>
S:    </iris:answer>
S:   </iris:resultSet>
S: </iris:response>
```

Exemple 1

Dans l'exemple suivant, un client IRIS demande des informations d'état de domaine pour "milo.exemple.com", "felix.exemple.com", et "hobbes.exemple.com" dans une demande. La demande est envoyée avec un tronçon ; cependant, la réponse est retournée en trois tronçons.

```
S:  (Bloc de réponse de connexion)
S: 0x20   (en-tête de bloc : V=0,KO=oui)
S: ( tronçon 1)
S: 0xC1      (LC=oui,DC=oui,CT=vi)
S: 0x01 0xBF (longueur de tronçon=447)
S:  (Informations de version )
S: <?xml version="1.0"?>
S: <versions xmlns="urn:ietf:params:xml:ns:iris-transport">
S:   <transferProtocol protocolId="iris.xpc1"
S:     authenticationIds="PLAIN EXTERNAL">
S:     <application protocolId="urn:ietf:params:xml:ns:iris1"
S:       extensionIds="http://exemple.com/SIMPLEBAG">
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dchk1"/>
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dreg1"/>
S:     </application>
S:   </transferProtocol>
S: </versions>

C: (Bloc de demande)
C: 0x00      (en-tête de bloc: V=0,KO=no)
C: 0x0B      (longueur d'autorité=11)
C:           (authority="exemple.com")
C: 0x65 0x78 0x61 0x6D 0x70 0x6C 0x65 0x23 0x63 0x6F 0x6D
C: ( tronçon 1)
C: 0xC7      (LC=oui,DC=oui,CT=ad)
C: 0x02 0xAB (longueur de tronçon=683)
C: (Demande XML IRIS)
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
C:   xsi:schemaLocation="urn:ietf:params:xml:ns:iris1 iris.xsd" >
C:    <searchSet>
C:     <lookupEntity
C:       registryType="urn:ietf:params:xml:ns:dchk1"
C:       entityClass="domain-name"
C:       entityName="milo.exemple.com" />
C:   </searchSet>
C:   <searchSet>
C:     <lookupEntity
C:       registryType="urn:ietf:params:xml:ns:dchk1"
C:       entityClass="domain-name"
```

```
C:      entityName="felix.exemple.com" />
C:   </searchSet>
C:   <searchSet>
C:    <lookupEntity
C:      registryType="urn:ietf:params:xml:ns:dchk1"
C:      entityClass="domain-name"
C:      entityName="hobbes.exemple.com" />
C:   </searchSet>
C: </request>

S: (Bloc de réponse)
S: 0x00      (en-tête de bloc: V=0,KO=no)
S: (tronçon 1)
S: 0x07      (LC=non,DC=non,CT=ad)
S: 0x01 0xDA (longueur de tronçon=474)
S:  (Réponse XML IRIS)
S: <iris:response xmlns:iris="urn:ietf:params:xml:ns:iris1">
S:   <iris:resultSet>
S:    <iris:answer>
S:      <domain authority="exemple.com" registryType="dchk1"
S:       entityClass="domain-name" entityName="milo.exemple.com-1"
S:       temporaryReference="vrai"
S:       xmlns="urn:ietf:params:xml:ns:dchk1">
S:       <domainName>milo.exemple.com</domainName>
S:       <status>
S:         <assignedAndActive/>
S:       </status>
S:      </domain>
S:    </iris:answer>
S:   </iris:resultSet>
S: (tronçon 2)
S: 0x07      (LC=non,DC=non,CT=ad)
S: 0x01 0xA2 (longueur de tronçon=418)
S:  (Réponse XML IRIS)
S:  <iris:resultSet>
S:    <iris:answer>
S:      <domain authority="exemple.com" registryType="dchk1"
S:       entityClass="domain-name" entityName="felix.exemple.com-1"
S:       temporaryReference="vrai"
S:       xmlns="urn:ietf:params:xml:ns:dchk1">
S:       <domainName>felix.exemple.com</domainName>
S:       <status>
S:         <assignedAndActive/>
S:       </status>
S:      </domain>
S:    </iris:answer>
S:  </iris:resultSet>
S:  (tronçon 3)
S: 0xC7      (LC=oui,DC=oui,CT=ad)
S: 0x01 0xB5 (longueur de tronçon=437)
S:  (Réponse XML IRIS)
S:  <iris:resultSet>
S:    <iris:answer>
S:      <domain authority="exemple.com" registryType="dchk1"
S:       entityClass="domain-name"
S:  entityName="hobbes.exemple.com-1"
S:       temporaryReference="vrai"
S:       xmlns="urn:ietf:params:xml:ns:dchk1">
S:       <domainName>hobbes.exemple.com</domainName>
S:       <status>
S:         <assignedAndActive/>
```

```
S:        </status>
S:      </domain>
S:    </iris:answer>
S:  </iris:resultSet>
S: </iris:response>
```

Exemple 2

Dans l'exemple suivant, un client IRIS envoie une demande contenant des données d'authentification SASL/PLAIN et une vérification d'état de domaine pour "exemple.com". Le serveur répond avec des informations de succès d'authentification et l'état de domaine de "exemple.com". Noter que le client demande que la connexion reste ouverte pour des demandes ultérieures, mais le serveur ne respecte pas cette demande.

```
S: (Bloc de réponse de connexion)
S: 0x20      (en-tête de bloc : V=0,KO=oui)
S: ( tronçon 1)
S: 0xC1      (LC=oui,DC=oui,CT=vi)
S: 0x01 0xBF (longueur de tronçon=447)
S:  (Informations de version)
S: <?xml version="1.0"?>
S: <versions xmlns="urn:ietf:params:xml:ns:iris-transport">
S:   <transferProtocol protocolId="iris.xpc1"
S:     authenticationIds="PLAIN EXTERNAL">
S:     <application protocolId="urn:ietf:params:xml:ns:iris1"
S:       extensionIds="http://exemple.com/SIMPLEBAG">
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dchk1"/>
S:       <dataModel protocolId="urn:ietf:params:xml:ns:dreg1"/>
S:     </application>
S:   </transferProtocol>
S: </versions>

C: (Bloc de demande)
C: 0x00      (en-tête de bloc : V=0,KO=no)
C: 0x0B      (longueur d'autorité=11)
C:           (authority="exemple.com")
C: 0x65 0x78 0x61 0x6D 0x70 0x6C 0x65 0x23 0x63 0x6F 0x6D
C: ( tronçon 1)
C: 0x44      (LC=no,DC=oui,CT=sd)
C: 0x00 0x12 (longueur de tronçon=18)
C: (données SASL)
C: 0x05      (longueur de mécanisme=5)
C:  (nom de mécanisme="PLAIN")
C: 0x50 0x4C 0x41 0x49 0x43
C: 0x00 0x0A (Longueur de données sasl PLAIN =10)
C:           (sasl PLAIN data: authcid="bob")
C:           (sasl PLAIN data: authzid=NULL)
C:           (sasl PLAIN data: password="kEw1")
C: 0x62 0x6F 0x62 0x20 0x00 0x20 0x6B 0x45 0x77 0x31
C:  ( tronçon 2)
C: 0xC7      (LC=oui,DC=oui,CT=ad)
C: 0x01 0x53 (longueur de tronçon=339)
C: (Demande XML IRIS)
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:   xsi:schemaLocation="urn:ietf:params:xml:ns:iris1 iris.xsd" >
C:   <searchSet>
C:     <lookupEntity
C:       registryType="urn:ietf:params:xml:ns:dchk1"
C:       entityClass="domain-name"
C:       entityName="exemple.com" />
C:   </searchSet>
C: </request>
```

```
S: (Bloc de réponse)
S: 0x00      (en-tête de bloc: V=0,KO=no)
S: (tronçon 1)
S: 0x45      (LC=non,DC=oui,CT=as)
S: 0x00 0xD0 (longueur de tronçon=208)
S: (Réponse de succès d'authentification)
S: <?xml version="1.0"?>
S: <authenticationSuccess
S:   xmlns="urn:ietf:params:xml:ns:iris-transport">
S:   <description language="fr">
S: l'utilisateur "Bob" s'authentifie avec un mot de passe
S:   </description>
S: </authenticationSuccess>
S:  (tronçon 2)
S: 0xC7      (LC=oui,DC=oui,CT=ad)
S: 0x01 0xE0 (longueur de tronçon=480)
S: (Réponse XML IRIS)
S: <iris:response xmlns:iris="urn:ietf:params:xml:ns:iris1">
S:   <iris:resultSet>
S:     <iris:answer>
S:       <domain authority="exemple.com" registryType="dchk1"
S:         entityClass="domain-name" entityName="exemple.com-1"
S:         temporaryReference="vrai"
S:         xmlns="urn:ietf:params:xml:ns:dchk1">
S:         <domainName>exemple.com</domainName>
S:         <status>
S:           <assignedAndActive/>
S:         </status>
S:       </domain>
S:     </iris:answer>
S:   </iris:resultSet>
S: </iris:response>
```

Exemple 3

Appendice B. Contributeurs

Des contributions substantielles au présent document ont été fournies par les membres du groupe de travail CRISP de l'IETF, en particulier Robert Martin-Legene, Milena Caires, et David Blacka.

Adresse de l'auteur

Andrew L. Newton
VeriSign, Inc.
21345 Ridgetop Circle
Sterling, VA 20166
USA

téléphone : +1 703 948 3382
mél : andy@hxr.us
URI : http://www.verisignlabs.com/

Déclaration complète de droits de reproduction

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