RFC 6885 : Stringprep Revision and PRECIS Problem Statement

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 14 mars 2013

Date de publication du RFC : Mars 2013

https://www.bortzmeyer.org/6885.html

Dans bien des protocoles, il y a besoin de comparer deux chaînes de caractères, pour voir si elles sont égales. Par exemple, un protocole d'authentification va comparer un identificateur à ceux enregistrés dans sa base, pour voir s'il est présent. Si les chaînes de caractères sont uniquement en US-ASCII, il n'y a guère de problèmes, à part décider si la comparaison est sensible à la casse. Mais si elles sont en Unicode ce qui, en 2013, est la moindre des choses ? Alors, c'est plus compliqué. Dans ce cas, on décide de **préparer** les chaînes avant la comparaison, pour les ramener à une forme de référence (en ASCII, un exemple simple de préparation serait de tout mettre en minuscules). Beaucoup de protocoles IETF utilisaient pour cette opération le stringprep du RFC 3454 [1], popularisé par son utilisation dans les noms de domaines internationalisés. Mais la mise à jour de ces noms de domaines, en 2010, a abandonné stringprep (cf. RFC 5890). Dans ces conditions, les autres protocoles se retrouvaient privés d'une référence bien pratique. Le groupe de travail PRECIS <http://tools.ietf.org/wg/precis> a été créé pour étudier la mise en place d'un cadre général de préparation des chaînes Unicode pour tous les protocoles. Ce RFC 6885 est son premier RFC et il décrit le problème, sans proposer encore de solution.

C'est que stringprep avait été un grand succès. Voici une liste incomplète des protocoles qui l'utilisent :

— NFS version 4 (RFC 8881),
— le protocole d'authentification EAP (RFC 3748),
— le protocole de communication XMPP (RFC 6120, profils Nodeprep et Resourceprep, car un identificateur XMPP comme stéphane@dns-oarc.net/Théâtre est légal),
— les noms dans les certificats X509 (RFC 4683),
— tous les protocoles qui passent par SASL comme l'authentification SMTP (RFC 4954) ou POP (RFC 5034),

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3454.txt

— le registre des algorithmes de comparaison (RFC 4790), et certains de ses algorithmes (comme celui du RFC 5051),
— iSCSI (RFC 3722) lorsqu'il s'agit de nommer l'initiateur (la machine) et la cible (le disque),
— et bien d'autres. Une évaluation très détaillée des différents profils figure dans l'annexe B de ce RFC.

Pour gérer plus facilement ce bestiaire, on peut les regrouper en quelques catégories (voir l'étude de Marc Blanchet <http://www.ietf.org/proceedings/78/slides/precis-2.pdf>), d'autant plus que beaucoup utilisent le même profil SASLprep du RFC 4013. L'annexe A de notre RFC liste ces catégories.

On l'a dit, la motivation immédiate pour le travail du groupe PRECIS <http://tools.ietf.org/wg/precis> était la perte de la référence que constituait les IDN, qui utilisent désormais d'autres méthodes. Puisque, de toute façon, il va falloir remettre à plat cette question de la préparation des identificateurs Unicode, c'était l'occasion de faire un bilan (il a commencé à la réunion IETF 77 <http://www.ietf.org/meeting/77/index.html> dans une Bof nommée « Newprep <https://datatracker.ietf.org/meeting/77/agenda/newprep/> » dont le compte-rendu est en ligne <http://www.ietf.org/proceedings/77/minutes/newprep.txt>). La section 4 étudie les profils de stringprep existants et leurs limites :

— Stringprep est lié à la version 3.2 d'Unicode. Or, plusieurs versions sont sorties depuis <https://www.bortzmeyer.org/unicode-6-0.html> et stringprep ne permet pas de les utiliser.
— Ce point est d'autant plus gênant qu'une application n'a en général pas le choix de la version Unicode : elle utilise ce que le système hôte lui fournit.
— Les chaînes de caractère à comparer (par exemple les identificateurs) sont souvent passés d'un protocole à l'autre et il serait donc souhaitable d'avoir des règles communes. Notez que, par exemple, XMPP utilise deux profils stringprep différents pour les composants du JID (l'identificateur XMPP) : la ressource, le domaine et la partie locale n'obéissent pas aux mêmes règles. Allez donc expliquer la syntaxe légale d'un JID dans ces conditions !

Bref, c'est sur la base de ce cahier des charges informel qu'a été créé le groupe PRECIS <http://tools.ietf.org/wg/precis> avec l'ambition de servir des protocoles très différents (mais qui ne bénéficiaient pas des compétences d'experts en internationalisation), en s'inspirant (sans forcément le copier aveuglément) de l'IDN bis <https://www.bortzmeyer.org/idnabis.html> du RFC 5890 et suivants.

La section 5 de ce RFC est le cahier des charges formel. D'abord, sur la question de la comparaison entre chaînes, un rappel du problème. Il y a trois catégories de chaînes, pour ce qui concerne la comparaison :

— Les absolues, qu'on compare octet par octet,
— Les définies, où la comparaison est plus complexe mais est bien définie, par un algorithme standard,
— Les indéfinies où il n'existe pas de règle standard (il n'y en a heureusement pas d'exemple connu dans les protocoles IETF).

Par exemple, deux chaînes Unicode identiques pour la deuxième catégorie (même séquence de points de code) peuvent ne pas l'être pour la première (par exemple si une est en UTF-8 et l'autre en UTF-32). Mais il peut exister un algorithme pour les comparer (tout convertir en UTF-8 puis comparer les octets). Le tableau en annexe A indique, pour les protocoles existants, à quelle catégorie ils appartiennent.

Ensuite, les caractères eux-même, un problème que la taille du bestiaire Unicode (plus de 100 000 caractères) rend difficile. Commençons par la casse. IDN v1 faisait une comparaison indépendante de la casse en convertissant tout en minuscules. IDN v2 a décidé de n'accepter que les minuscules, une éventuelle conversion des majuscules étant à faire en dehors du cadre IDN (RFC 5892). Le système défini par PRECIS doit donc gérer plusieurs cas, avec ou sans sensibilité à la casse, avec ou sans préservation de la casse.

Ensuite, la canonicalisation. Comme il existe plusieurs chaînes Unicode dont le rendu peut être identique (exemple classique : U+00C8, d'une part, et U+0045 U+0300, d'autre part, vont tous les deux donner É), Unicode prévoit des algorithmes de canonicalisation qui réduisent les chaînes de caractères à une forme canonique, avant la comparaison. La plupart des profils de stringprep se servent de NFKC. Est-ce le bon choix ? Dans tous les cas ? NFC peut être une meilleure idée (au fait, dans l'exemple ci-dessus, la chaîne deviendra U+00C8 avec NFC et avec NFKC).

Autre question délicate à considérer, les caractères interdits. Par exemple, un protocole peut avoir des caractères spéciaux comme @ ou / qu'il faudra interdire dans les identificateurs. Faut-il définir les caractères interdits (et autoriser donc le reste) ou au contraire les caractères permis (et donc interdire tous les autres) ?

Cette question est partiellement liée à la question de l'homographie. Ce RFC reprend la légende (pourtant bien démontée par UTR36 <http://www.unicode.org/reports/tr36/>) comme quoi les caractères pourraient avoir un rôle dans le hameçonnage <https://www.bortzmeyer.org/idn-et-phishing.html>. Mais il reconnait qu'il n'y a pas de solution technique à cette question.

Autre question sur les données, d'où viennent-elles ? Sont-elles typiquement saisies par un utilisateur (et, dans ce cas, il faut probablement lui laisser un grand choix de variantes possibles) ou bien sont-elles générées par un programme ? Derrière cette question se trouve la frontière oh combien mouvante entre ce qui relève de l'interface utilisateur et ce qui relève des protocoles (voir le RFC 5895 et aussi, même s'il n'est pas cité, le RFC 2277).

Après l'entrée, la sortie de ces chaînes de caractères soulève aussi des questions. Si les caractères sortent du monde « machine » et sont imprimés, pour se retrouver sur une carte de visite ou sur les flancs d'un autobus, les problèmes de risque de confusion visuelle seront plus aigus.

Et il y a le fait que certaines chaînes de caractères servent d'entrée au processus suivant, condensation cryptographique ou concaténation, par exemple.

Dernier problème à garder en mémoire, le risque de changements incompatibles lors de la sortie d'une nouvelle version d'Unicode. Idéalement, un caractère, une fois déclaré valide pour un protocole donné, devrait le rester éternellement, assurant ainsi la stabilité des identificateurs construits avec ce caractère. Mais Unicode ne fournit pas cette garantie (au passage, c'est pour cela que stringprep était lié à une version particulière d'Unicode). Un caractère peut changer de catégorie et, si sa validité était déterminée algorithmiquement à partir de sa catégorie, il peut passer de valide à invalide. Le cas s'est déjà produit (cf. RFC 6452).

La section 6 est ensuite le relevé des décisions (unanimes) concernant le travail du groupe PRECIS :

— Accord pour développer un remplaçant pour stringprep, ne gérant qu'Unicode (les autres jeux de caractères sont éliminés).
— Reprendre le mécanisme d'IDNA bis <https://www.bortzmeyer.org/idnabis.html> pour être indépendant de la version d'Unicode, à savoir faire dériver la validité ou nom d'un caractère d'un algorithme utilisant les propriétés Unicode, algorithme qu'il suffira de refaire tourner à chaque nouvelle version d'Unicode.
— Essayer de définir un petit nombre de profils, chacun pouvant convenir à une grande classe de protocoles.
— Garder en tête tous les problèmes posés en section 5.