RFC 5894 : Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 août 2010

Date de publication du RFC : Août 2010

https://www.bortzmeyer.org/5894.html

Dans l'ensemble des RFC sur IDNA bis <https://www.bortzmeyer.org/idnabis.html>, celui-ci joue le rôle du document non officiel mais qui éclaire, explique et justifie les autres. Il n'est pas nécessaire de le lire pour mettre en œuvre les IDN mais il peut aider à comprendre la démarche des auteurs de IDNAbis. Mon compte-rendu va être un peu plus polémique que d'habitude car 1) ce RFC n'explique pas grand'chose, en fait et 2) il contient beaucoup de rhétorique et de FUD.

Officiellement, IDNAbis est nommé IDNA2008 (section 1), car c'était la date (totalement irréaliste, et qui avait été pointée comme telle par de nombreux participants) à laquelle le projet devait se terminer. La section 1 rappelle aussi quelques points sur lesquels IDNA 1 posait des problèmes, le changement de la gestion de la casse (IDNA 1 était insensible à la casse, comme le DNS) alors que IDNAbis résout le problème autrement, en interdisant les majuscules) ou comme la dépendance de IDNA 1 à une version spécifique d'Unicode (ce qui interdisait les écritures qui sont entrées dans Unicode plus tard, comme le Tifinagh).

Les objectifs d'IDNAbis sont détaillés dans la section 1.4 :
— Indépendance vis-à-vis d'une version particulière d'Unicode (certainement l'objectif qui était le plus consensuel),
— Régler quelques limites d'IDNA 1 qui interdisaient, en pratique, certaines langues comme le Yiddish, qui avaient besoin de certains caractères, qui étaient exclus,
— Réduire (en fait, elle a même été supprimée) l'étape de canonicalisation (qui était faite avec nameprep, du RFC 3491 [1]),

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3491.txt

— Changer les règles du bidi (cf. le nouveau RFC 5893).

Pour le reste, le RFC est plutôt une collection assez décousue de divers points qui furent discutés lors du projet IDNAbis, points sur lesquels John Klensin tenait à faire connaître son opinion personnelle. Cela tient du blog plutôt que du RFC. Voyons quelques-uns de ces points.

Le terme même de « nom de domaine » peut entraîner des confusions car un nom dans le DNS n'est pas forcément une phrase ou même un mot dans une langue naturelle (section 1.3.1). Plusieurs règles du DNS interdisent certaines phrases ou certains mots (par exemple, la société C&A ne peut pas avoir de domaine à son nom, l'esperluette n'étant pas autorisée, avec ou sans IDN ; même chose avec le nom local de l'archipel d'Hawai'i, l'apostrophe n'étant pas acceptée). L'argument de Klensin est que, finalement, l'ancienne restriction à ASCII n'est pas si terrible puisque, de toute façon, on ne peut pas « écrire un roman dans le DNS ». Dommage qu'il se sente obligé de ridiculiser ceux qui ont une autre approche en les accusant de vouloir écrire du Klingon.

La section 1.5 concerne les limites d'IDNA, les points qu'il ne résoudra pas. Par exemple, le DNS ne permet pas de recherche floue, il faut indiquer le nom de domaine exact et cela ne change pas. L'augmentation du nombre de caractères admissibles, grâce à IDNA, peut rendre ce problème plus palpable (par exemple si on lit un peu trop vite un nom de domaine qu'on a vu sur le côté d'un bus).

Comme IDNA 1, IDNAbis ne nécessite pas de changement de l'infrastructure, comme l'aurait nécessité un hypothétique DNS Unicode <`https://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html`>. Une des conséquences est qu'il existe une forme ASCII de chaque IDN, le "*A-label*" et que ce nom (par exemple `xn--pgbs0dh` pour "*[Caractère Unicode non montré* [2] *][Caractère Unicode non montré][Caractère Unicode non montré][Caractère Unicode non montré]*") peut toujours être utilisé comme solution de secours.

La section 1.6 raconte qu'un des buts de IDNAbis était d'améliorer la « comprehensibilité » d'IDN. Qu'une personne normale, n'ayant pas forcément lu les RFC, comprenne mieux le fonctionnement d'IDN. D'où la suppression de l'étape de canonicalisation, jugée trop déroutante. Avec IDNAbis, les noms de domaine en Unicode ("*U-labels*") seront tous déjà sous forme canonique (les autres étant interdits), ce qui devrait augmenter ladite compréhensibilité.

En fait, la canonicalisation (par exemple de `CAFÉ` vers `café`) sera faite dans l'interface utilisateur et plus dans le protocole et je ne crois pas que cela rende les choses plus prévisibles, d'autant plus que deux interfaces utilisateur différentes pourront canonicaliser différemment.

Traditionnellement, le DNS avait des règles différentes pour la **résolution** (qui utilisait des règles standard, comme l'insensibilité à la casse, mais aussi l'acceptation de tous les caractères, au delà de ASCII, cf. RFC 2181, section 11) et l'**enregistrement** (où les règles sont décidées localement par le registre et sont typiquement bien plus sévères, par exemple limitation à l'ASCII, interdiction des gros mots, etc). IDNAbis formalise cette distinction, pour refléter cette pratique (section 2 et RFC 5891).

Une des caractéristiques nouvelles et importantes de IDNAbis est que les caractères Unicode sont désormais interdits par défaut, alors qu'avec IDNA 1, ils étaient autorisés, sauf quand c'était indiqué explicitement. La section 3 revient sur ce choix. L'algorithme exact figure dans le RFC 5892. Une autre différence entre IDNA 1 et IDNAbis est que la validité d'un caractère dépend désormais du contexte. En effet, certains caractères notamment les « gluons », les caractères qui contrôlent le fait que deux caractères soient séparés ou pas, peuvent être jugés raisonnables ou pas, selon les caractères autour

2. Car trop difficile à faire afficher par LaTeX

d'eux. C'est le cas par exemple de U+200D - le gluon de largeur nulle. En IDNA 1, tous ces caractères de la catégorie Unicode "*Join_Controls*" étaient interdits. Ils sont désormais autorisés conditionnellement (section 3.1.2)

Par défaut, en IDNAbis, les caractères tombent dans la catégorie IDN `DISALLOWED` (section 3.1.3). On y trouve des caractères qui ne sont typiquement pas considérés comme lettre ou chiffres et donc ne correspondent pas aux traditionnels critères pour les identificateurs. C'est le cas du [Caractère Unicode non montré] (U+2044) ou du [Caractère Unicode non montré] (U+2665).

Les règles du protocole IDNAbis ne sont pas la totalité des règles applicables. En effet, le registre peut ajouter des règles supplémentaires. Par exemple, la plupart des registres interdisent l'enregistrement de noms comportant des caractères parfaitement valides dans le DNS <https://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html> comme le _ (U+005F). La section 3.2 discute des politiques des registres. Il serait plus juste de dire que cette section aboie des ordres, sur un ton paternaliste. On y trouve beaucoup d'opinions personnelles non étayées, déguisées en « bonnes pratiques » par exemple l'idée qu'il ne faut pas accepter les noms composés de plusieurs écritures (alors la plupart des utilisateurs d'alphabets non-latins acceptent en plus les caractères latins). Certaines recommandations sont intéressantes (comme celle du système des variantes, cf. RFC 3743, RFC 4290 et RFC 4713) mais d'autres relèvent plutôt d'une volonté de bras de fer, qui avait souvent été exprimée ouvertement dans les réunions du groupe de travail IDNAbis (par exemple, Klensin disant que « les registres pensaient uniquement à l'argent et qu'il fallait les contraindre à respecter l'intérêt général », intérêt qu'il exprimait, lui).

De même, cette section justifie la règle (section 4.1 du RFC 5891) comme quoi le demandeur doit envoyer à la fois le "*U-label*" (forme Unicode) et le "*A-label*" (forme Punycode) au registre, en prétendant que c'est pour vérifier la cohérence des deux. Pourtant, la forme Punycode n'est qu'une astuce technique pour déployer les IDN, ce que l'utilisateur veut, c'est la forme Unicode et il est tout à fait anormal de prétendre que le "*A-label*" aie un quelconque intérêt pour l'utilisateur !

Ces règles et bien d'autres ont souvent été justifiées au début du processus IDNAbis par de vagues arguments de sécurité (section 2.2.6 du RFC 4690). Cette baudruche s'est dégonflée assez vite <https://www.bortzmeyer.org/idn-et-phishing.html> et, aujourd'hui, la section 3.3 note à juste titre qu'il n'y a pas de solution technique à des questions comme celle de la confusabilité de deux caractères (par exemple le 0 - U+0030 - et le O - U+004F).

Traditionnellement, l'IETF travaille uniquement sur ce qui se passe sur le câble, loin des utilisateurs. Ici, toutefois, on ne peut pas ignorer les problèmes des applications, IDNA est faite pour elles (le A final du sigle). La section 4 se penche donc sur les logiciels que verra l'utilisateur. D'abord, contrairement au réseau, où les noms de domaines sont toujours transmis dans le même ordre (celui de la saisie au clavier), l'affichage des IDN peut être de droite à gauche ou de gauche à droite (section 4.1). Cela peut être d'autant plus complexe à gérer pour, par exemple, un navigateur Web, que le nom complet peut avoir des composants de gauche à droite et d'autres de droite à gauche (par exemple `www.[Caractère Unicode non montré ][Caractère Unicode non montré ][Caractère Unicode non montré ][Caractère Unicode non montré ].com`). Ce cas est encore pire pour un IRI (RFC 3987) puisque celui-ci a forcément au moins un composant ASCII, le **plan** (par exemple `http`).

Saisir et afficher des IDN nécessite donc des choix délicats et des codes complexes (alors qu'un serveur de noms, par exemple, ou même une base de données d'un registre qui stocke des noms de domaine, n'ont rien de particulier à faire). La section 4.2 donne quelques conseils. Elle rappelle par exemple que les noms de domaine peuvent apparaitre dans un contexte où le fait qu'il s'agisse d'un nom de domaine est évident (par exemple lors d'un dialogue SMTP) mais aussi dans du texte libre, où le logiciel ne comprend pas forcément ce qu'il transmet. Les protocoles devraient donc bien préciser quand un nom

de domaine est une partie du protocole (commande `MAIL FROM` de SMTP) et quand il ne l'est pas (le corps d'un message envoyé en SMTP).

Un problème en soi est celui de la **canonicalisation** ("*mapping*", mise en correspondance) des caractères saisis par l'utilisateur avec ce que IDNAbis accepte. Dans IDNA 1, il existait une canonicalisation officielle, "*nameprep*", normalisée dans le RFC 3491. Elle a été supprimée et, désormais, la canonicalisation est laissée à l'initiative de l'application. La seule obligation est de produire un nom conforme au protocole IDNAbis. Comme celui-ci, par exemple, exclus les majuscules (contrairement à la tradition du DNS, où majuscules et minuscules sont autorisées, tout en étant équivalentes), l'application doit mettre le nom en minuscules (une tâche triviale en ASCII mais bien plus complexe en Unicode, elle peut même dépendre de la langue). Le RFC 5895 décrit un exemple d'une telle canonicalisation.

A priori, cette canonicalisation locale, chacun faisant comme il veut, risque de dérouter et d'entrainer bien des problèmes : la même chaîne de caractères en Unicode, saisi dans deux navigateurs différents, pourra donner des noms de domaines ("*U-label*") différents. L'idée est que chaque application prendra une décision conforme au contexte local (l'application connait l'utilisateur, connait sa langue) et que cela sera finalement moins surprenant. En outre, cela permet d'avoir, contrairement à IDNA 1, un aller-retour sans perte dans les conversions entre "*U-label*" et "*A-label*".

D'autres attentes linguistiques peuvent poser un problème au développeur de logiciel IDNAbis. La section 4.3 fournit quelques exemples pittoresques :

— Norvégiens et suédois peuvent considérer que `[Caractère Unicode non montré ]` et `[Caractère Unicode non montré ]` sont équivalents, ce qui dérouterait un utilisateur anglophone. Même chose avec `oe` et `[Caractère Unicode non montré ]` pour un allemand.
— Un chinois regarde en général les formes simplifiées et traditionnelles d'un caractère comme équivalentes, or ces caractères sont aussi utilisés pour le japonais où cette opération n'aurait pas de sens.
— Tout simplement, un anglophone pourrait s'étonner que `theatre` et `theater`, ou bien `color` et `colour` soient deux noms de domaine distincts... La frontière entre l'équivalence que doit faire le protocole et celle qui est laissée à la responsabilité de l'utilisateur n'est pas évidente à placer.

Autre cas de canonicalisation délicat : la distinction entre majuscules et minuscules. Pour les utilisateurs de l'alphabet latin, les deux casses sont souvent considérées comme sémantiquement équivalentes et c'est ce point de vue que reflète le DNS lorsque le RFC 1034 section 3.1 dit « "*domain name comparisons for all present domain functions are done in a case-insensitive manner*" ». Avec Unicode, un tel principe n'est plus tenable, comme le rappelle la section 4.4. Ni IDNA 1, ni IDNAbis n'obligent les serveurs à être insensibles à la casse et pour de bonnes raisons <`https://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html`>. Par exemple, certains caractères n'ont pas de majuscule propre comme le sigma final [Caractère Unicode non montré]. Une conversion en majuscules en fait un [Caractère Unicode non montré] dont la forme minuscule est le sigma ordinaire [Caractère Unicode non montré]. IDNA 1 résolvait la question en imposant aux applications une canonicalisation en minuscules via "*nameprep*" (RFC 3491), IDNAbis choisit d'ignorer le problème en n'acceptant que les minuscules et en laissant l'application canonicaliser à sa guise, en suivant les règles locales. À noter que ce changement peut entraîner quelques incompatibilités entre les deux versions d'IDN. Ainsi, en IDNA 1, `strasse.de` et `stra[Caractère Unicode non montré ]e.de` sont le même nom de domaine (puisque nameprep canonicalise le second dans le premier) alors qu'ils sont différents en IDNAbis.

Le cas des écritures qui s'affichent de droite à gauche fait l'objet de la section 4.5. Pour limiter les risques d'ambiguité, IDNA 1 et IDNAbis imposent que, dans chaque composant d'un nom de domaine, il n'y aie que des caractères de la même directionnalité (ou des caractères neutres). C'est une des rares règles d'IDN qui examinent le composant entier, pas juste des caractères individuels.

La règle exacte dans IDNA 1 était trop stricte pour certaines langues qui ont besoin de marques vocaliques comme le yiddish écrit en hébreu ou le divehi écrit en thaana. Ces marques n'ont pas de directionalité et étaient interdites à la fin d'un composant. Ce n'est plus le cas et IDNAbis permet donc des noms qui étaient interdits en IDNA 1 (cf. RFC 5893).

En revanche, l'idée d'avoir des règles entre composants (étendre la règle ci-dessus à tout le FQDN) a été abandonnée. Elle aurait imposé une partie de bras de fer avec les registres de noms. (Le RFC ne mentionne pas cette très mauvaise idée, qui était pourtant promue par son auteur...) Il est donc toujours possible d'avoir un nom de domaine dont certains composants sont de gauche à droite et d'autres de droite à gauche.

La section 5 se réclame du fameux principe de robustesse (« soyez strict dans ce que vous envoyez et tolérant dans ce que vous recevez ») pour culpabiliser les registres de noms de domaine en leur faisant porter la responsabilité de contrôles plus étendus. En pratique, l'application qui accède à un IDN ne peut pas compter dessus, à la fois parce que tous les registres n'ont pas forcément les obsessions de John Klensin, mais aussi parce que des techniques comme les jokers (RFC 1034, section 4.3.3) font qu'un nom peut être résolu bien qu'il n'aie jamais été enregistré.

Encore plus délicate, la question des interfaces utilisateur (section 6). Il n'est évidemment pas possible de donner des règles applicables à tous les cas, vue la grande variété de ces interfaces. La section 6 se concentre surtout sur la nouveauté de IDNAbis ayant le plus d'implication pour l'interface : le fait que la canonicalisation standard (nameprep) du RFC 3491 aie disparu. Cette canonicalisation standard avait des avantages, notamment une meilleure interopérabilité, due au caractère plus prédictible des noms. Mais d'un autre côté, elle menait à des canonicalisations qui ne tenaient pas compte de la "*locale*" et pouvaient donc être surprenantes pour l'utilisateur humain. La nouvelle règle (chaque application canonicalise comme elle veut) permet aux applications (qui connaissent l'utilisateur, la "*locale*", la langue...) de produire, à partir de ce qu'a tapé ou sélectionné l'utilisateur, des noms qui seront moins déroutants.

Pour ceux qui avaient déjà déployé les IDN avec l'ancienne norme IDNA 1, la section 7 fournit une description détaillée des mécanismes de migration possibles, et des pièges qui peuvent les guetter. Ainsi, comme le précise la section 7.2, des chaînes de caractères identiques en IDNA1 ne le sont plus en IDNAbis ("*strasse*" et "*stra[Caractère Unicode non montré]e*" ou bien les noms utilisant les gluons Unicode comme le U+200D) et donc deux noms de domaines qui étaient équivalents en IDNA 1 ne le sont plus (et ont donc des "*A-labels*" différents). Le choix n'a pas été évident. La section 7.2.3 résume les possibilités qui s'offraient au groupe de travail, dont certaines étaient très lourdes (changer le préfixe de l'encodage, actuellement `xn--`, par exemple, pour marquer clairement l'incompatibilité). Finalement, après de très longues discussions, le choix a été fait d'accepter cette légère incompatibilité. Une fois cette décision prise, quelle stratégie adopter pour accueillir ces « nouveaux » caractères ? La section 7.2.4 en décrit plusieurs, du refus des nouveaux, qui empêchera les anciens noms de fonctionner mais évitera toute confusion, à la période de transition ("*sunrise*") pendant laquelle les titulaires des anciens noms auraient priorité pour enregistrer le nom incluant les nouveaux caractères. Par exemple, un registre germanophone pourrait donner la priorité au titulaire de `strasse.de` pour qu'il enregistre `stra[Caractère Unicode non montré ]e.de` avant tout le monde. Il y a aussi l'éventualité d'un système de « variantes » où les noms « anciens » et « nouveaux » seraient liés en permanence et enregistrables uniquement par le même titulaire. Et la toute simple possibilité de ne pas s'en préoccuper, en considérant que le problème est mineur et disparaitra peu à peu. À l'heure actuelle (juin 2010), le registre du `.AT` prévoit de ne pas utiliser de variantes et de ne pas accepter immédiatement les nouveaux caractères <`http://www.nic.at/uebernic/aktuelles/nicat_news/news_ansicht/article/81/sonderzeichen-ss-in-domainnamen/`>. Lorsque ce sera fait, je suppose qu'il y aura une période de transition avec priorité aux anciens titulaires. C'est ce qu'a fait le registre du `.DE`, DENIC, qui fournit une période de transition <`http://www.denic.de/en/denic-in-dialogue/news/2985.html`>.

La question du changement de préfixe fait même l'objet d'une section spécifique, 7.4. En effet, la question avait été sérieusement envisagée. Ce changement aurait créé deux familles d'IDN complètement séparées, celle dont les "*A-labels*" commencent par `xn--` et la nouvelle. Le groupe de travail a considéré que le changement de préfixe aurait été nécessaire si les changements avaient créé un risque de confusion. La section 7.4.1 énumère précisément les cas où cela se serait produit, par exemple si la conversion d'un "*A-label*" en "*U-label*" avait renvoyé deux chaînes Unicode différentes ; le cas inverse était jugé moins grave, et il s'est effectivement produit dans de rares cas, comme dans l'exemple du [Caractère Unicode non montré] ci-dessus. Autre exemple, celui où l'encodage du "*A-label*" aurait été drastiquement modifié, par exemple pour inclure de l'information sur la langue utilisée.

En revanche, le groupe de travail avait considéré que des changements comme l'interdiction de caractères autrefois autorisés (section 7.4.2) ne justifiait pas un changement de préfixe. Cette interdiction rend des noms enregistrés inutilisables dans le DNS mais l'idée est qu'un refus clair est moins grave qu'un changement de sémantique et ne justifie donc pas de changement de préfixe. Un tel changement aurait eu des coûts considérables, détaillés en section 7.4.3. En effet, si un registre peut toujours changer tous les "*A-labels*" facilement (`UPDATE Domains SET alabel TO idnabis_alabel(ulabel)`), il n'aurait pas pu synchroniser ce changement avec celui des applications qui ont l'encodage en Punycode.

Et l'algorithme nameprep, normalisé dans le RFC 3491, que devient-il ? Il est complètement abandonné en IDNAbis mais nameprep n'est qu'un profil particulier de stringprep, normalisé dans le RFC 3454, et celui-ci est utilisé par bien d'autres normes IETF et il continue donc sa vie (section 7.5), sans que IDNAbis ne l'affecte.

Un des grands changements théoriques entre IDNA 1 et IDNAbis est l'interdiction des symboles comme U+2665 ([Caractère Unicode non montré]), U+2605 ([Caractère Unicode non montré]) ou U+2798 ([Caractère Unicode non montré]). C'est un changement surtout théorique car, en pratique, ils étaient souvent interdits par les règles d'enregistrement et on ne les trouve pratiquement pas en pratique. (Voir, par exemple, le "*IESG Statement on IDN*" <`http://www.ietf.org/iesg/statement/idn.html`>.) Cette méfiance vis-à-vis des symboles vient entre autre du fait qu'il n'existe pas de nommage standard pour les désigner et que les variations de forme entre polices sont encore plus marquées que pour les lettres. Il n'est donc pas facile d'épeler un nom de domaine comme `I[Caractère Unicode non montré ]NY.us` sans ambiguité... D'autant plus que certains symboles ont beaucoup de caractères Unicode correspondants (comme le carré).

La même section 7 couvre le cas de la migration interne à IDNAbis lorsqu'une nouvelle version d'Unicode est publiée (section 7.7). En effet, IDNAbis n'est plus lié à une version spécifique d'Unicode et l'enregistrement de nouveaux caractères, ou bien certains changements dans les caractères déjà enregistrés, peuvent faire apparaitre « automatiquement » de nouveaux caractères légaux dans IDN.

Parmi les innombrables questions qu'avaient soulevé l'introduction des IDN en 2003, le comportement des logiciels serveurs de noms comme BIND (section 8). Le souci de ne pas leur faire faire des canonicalisations Unicode est la principale raison pour laquelle nous avons IDNA ("*IDN in Applications*") et pas un DNS Unicode <`https://www.bortzmeyer.org/pourquoi-idn-et-pas-un-dns-unicode.html`>). Si le DNS a toujours prévu une canonicalisation minimale (les noms de domaine sont insensibles à la casse), celle-ci n'a jamais été normalisée pour Unicode (cf. RFC 4343). Ce point ne change pas en IDNAbis.

Donc, une des rares conséquences réelles pour les serveurs de noms est la longueur plus importante de beaucoup d'IDN. Ce point est mentionné en section 8.2, mais sans préciser que DNSSEC, bien plus gourmand, a été déployé sans trop de problèmes. Depuis la rédaction du RFC, l'introduction de quatre TLD IDN dans la racine du DNS a bien montré qu'il n'y avait aucun problème technique, malgré le FUD qui a été répandu à ce sujet.

Vu le sujet du RFC, la section facultative sur l'internationalisation se devait d'être présente (section 9). Elle rappelle que les noms dans le DNS, quoique mnémoniques, ne sont pas écrits selon les règles des langues humaines (par exemple, l'espace n'y est pas permis) et qu'il ne faut donc pas demander aux IDN de permettre l'écriture de n'importe quelle phrase, en respectant toutes les règles de la langue. Ceci dit, la même section oublie par contre de dire que, si les noms de domaine ne sont pas du texte en langue naturelle, ils ne sont pas non plus de purs identificateurs (c'est pour cela qu'ils ont une utilisation mnémonique) et c'est bien cela qui justifie IDN (personne ne demande l'internationalisation des adresses IP qui sont, elles, de purs identificateurs).

Une autre raison pour laquelle les règles des langues humaines ne peuvent pas être intégralement respectées dans le DNS est que le DNS est mondial et qu'un nom de domaine doit pouvoir être utilisé partout. D'autre part, le RFC rappelle également qu'un nom de domaine n'est pas écrit dans une langue particulière. Quelle est la langue de `coca-cola.com`? (Certainement pas de l'anglais.)

Le développement de IDNAbis a nécessité la création de certains nouveaux registres à l'IANA. La section 10 revient sur ces registres. Elle rappelle que la liste des caractères autorisés n'est pas normative, seul l'est l'algorithme du RFC 5892. En revanche, la liste des règles contextuelles <`https://www.iana.org/assignments/idnabis-tables/idnabis-tables.xhtml#idnabis-tables-context`> est normative. Plus délicate, la liste des politiques d'enregistrement <`https://www.iana.org/domains/idn-tables/`>, dont la section 10.3 rappelle qu'elle n'a aucune valeur, c'est juste un dépôt volontaire de leurs politiques par certains registres. Elle n'est spécifiée dans aucun RFC et ne découle que de considérations politiciennes à l'ICANN.