Chapitre 14

Zones spéciales et caractères de formatage

Ce chapitre décrit différents types de caractères aux propriétés singulières ainsi que des zones de l'espace de codage réservées à des fins particulières :

- les commandes générales ;
- les commandes de mise en page ;
- les caractères de formatage à éviter ;
- les seizets d'indirection ;
- la zone à usage privé ;
- les caractères spéciaux.

En plus des caractères normaux, Unicode comprend un certain nombre de caractères qui ne s'affichent pas directement[1], mais qui influencent la mise en page du texte ou modifient le comportement des processus de traitement de texte. On les appelle les caractères de formatage.

Le standard Unicode a prévu 65 positions de code destinées aux 64 caractères de commande et au caractère de suppression (DEL) présents dans les normes ISO et de nombreux jeux de caractères propriétaires.

Les commandes de mise en page n'ont pas d'apparence propre, mais elles influent sur le comportement des algorithmes de coupure de ligne, de coupure de mot, de sélection de glyphe et de mise en ordre bidirectionnel.

Les seizets d'indirection s'utilisent par paire ; ils sont destinés à permettre l'accès en UTF-16 à 1 048 544 caractères supplémentaires.

La zone à usage privé est réservée à des caractères dont la signification est définie par convention de gré à gré entre les utilisateurs.

Le bloc des caractères spéciaux comprend les caractères qui ne sont ni des caractères graphiques ni des commandes classiques.

[1] Pour des raisons didactiques, cet ouvrage représente cependant souvent ces caractères par un carré pointillé. Unicode a également défini des pictogrammes pour les noms de caractères de commande (U+2400..U+2426).

14.1 Commandes

Commandes C0 : U+0000 – U+001F

Commandes ASCII C0 et Suppression. Le standard Unicode n'attribue pas de sens particulier à ces codes de commande, mais les prévoit uniquement afin de permettre leur transmission sans perte, Unicode n'ajoute ni ne soustrait rien à leur sémantique. En règle générale, le sens des commandes C0 (U+0000..U+001F) et de *suppression* (U+007F) dépend de l'application qui les utilise. Toutefois, en absence de contexte particulier, on peut les interpréter conformément à la sémantique établie par la norme ISO/CEI 6429. U+0009 TABULATION HORIZONTALE s'utilise souvent munie d'une même signification, c'est pourquoi Unicode la prend en compte. Pour plus de détails sur les codes de commande, voir la *Section 2.8*, *Commandes et suite de commandes*.

Il existe une correspondance biunivoque simple entre les codes de commande à 7 bits (ou 8 bits) et les codes de commande Unicode : on ajoute le nombre nécessaire de zéros au début de chaque code de commande à 7 bits (ou 8 bits). Ainsi, si on désire utiliser U+000A PASSAGE À LA LIGNE (<PL>) pour commander un terminal, on transmettra alors le texte « WX<PL> YZ » en UTF-16 sous la forme suivante : « 0057 0058 000A 0059 005A ». L'interprétation de ces codes de commande ne tombe pas dans le champ d'application d'Unicode ; les développeurs doivent se référer à la norme pertinente (par exemple, ISO/CEI 6429) qui précise le sens des codes de commande.

Fonction changement de ligne. Un ou plusieurs codes de commande U+000A PASSAGE À LA LIGNE, U+000D RETOUR DE CHARIOT ou l'équivalent Unicode du NL EBCDIC représentent la *fonction changement de ligne*. Une fonction de changement de ligne peut, selon l'application, se comporter comme un séparateur de lignes ou de paragraphes. Voir la *Section 14.2*, *Commandes de mise en page*, pour plus de renseignements sur la manière d'interpréter les séparateurs de lignes et de paragraphes. Le codage précis de la fonction de changement de ligne dépend du domaine d'application, voir le *Rapport technique d'Unicode n° 13*, « *Unicode Newline Guidelines »,* présent sur le cédérom ou, pour une version tenue à jour, sur le site Internet du consortium Unicode.

Commandes C1 : U+0080 – U+009F

Lors du passage du système de codage à 7 bits ASCII à un système à 8 bits, l'ISO/CEI 4873 (sur lequel repose la famille des normes de caractères 8859) a incorporé 32 codes de commande supplémentaires dans l'intervalle 80-9F hexadécimal. À l'instar des codes de commande C0, le standard Unicode n'attribue pas de sens particulier à ces codes de commande, mais les prévoit uniquement afin de permettre leur transmission sans perte, Unicode n'ajoute ni ne soustrait rien à leur sémantique. En règle générale, le sens des commandes C1 (U+0080..U+009F) dépend de l'application qui les utilise. Toutefois, en absence d'un contexte particulier, on peut les interpréter conformément à la sémantique établie dans la norme ISO/CEI 6429.

14.2 Commandes de mise en page

Mise en page

L'effet des commandes de mise en page dépend des processus de texte en cause. Dans la mesure du possible, les commandes de mise en page sont sans effet pour les processus textuels non pertinents. En d'autres termes, leurs effets sont mutuellement orthogonaux.

Coupure de ligne et de mot

U+00A0 ESPACE INSÉCABLE à la même chasse que U+0020 ESPACE, cependant U+00A0 ESPACE INSÉCABLE indique que, d'ordinaire, il n'est pas permis de couper la ligne entre celui-ci et les caractères voisins, à moins que le caractère suivant ou précédent soit un séparateur de lignes ou de paragraphes. U+00A0 ESPACE INSÉCABLE se comporte comme la suite de caractères codés <U+FEFF ESPACE INSÉCABLE SANS CHASSE, U+0020 ESPACE, U+FEFF ESPACE INSÉCABLE SANS CHASSE>. Pour une liste complète des espaces Unicode, voir le *Tableau 7-1*, *Caractères d'espacement Unicode*.

U+00AD TRAIT D'UNION VIRTUEL indique un point de coupure de mot potentiel. La réalisation graphique de ce caractère, si une coupure de mot y survient, varie selon le système d'écriture : certaines écritures, par exemple, le représentent à l'aide d'un trait d'union « - », alors que pour d'autres il restera invisible. Il faut le distinguer de U+2027 • POINT DE COUPURE DE MOT qui sert à indiquer de manière visible le point de coupure de mot dans certains dictionnaires. Pour une liste complète des traits Unicode, y compris les traits d'union, veuillez consulter le *Tableau 7-2*, *Tirets d'Unicode*.

Unicode comprend deux traits d'union insécables : U+2011 ‑ TRAIT D'UNION INSÉCABLE et U+0F0C ་ SIGNE DÉLIMITEUR TIBÉTAIN TSHEG BSTAR (ts'ek tar). Voir la *Section 10.13*, *Tibétain*, pour un examen de la coupure de ligne propre au tibétain.

Espace insécable sans chasse. En sus de son rôle d'indicateur d'ordre des octets, le numéro de caractère U+FEFF possède également la sémantique d'une ESPACE INSÉCABLE SANS CHASSE.

En tant qu'ESPACE INSÉCABLE SANS CHASSE, U+FEFF se comporte comme une U+00A0 ESPACE INSÉCABLE par l'absence de frontière de mot qu'elle indique ; toutefois U+FEFF a une largeur nulle. On insérera donc ce caractère après le quatrième caractère du texte « base+delta » pour s'assurer que l'on ne coupe pas la ligne entre le « e » et le « + ». On utilise également ESPACE INSÉCABLE SANS CHASSE en compagnie d'autres caractères qui non pas de variante insécable. C'est ainsi qu'on peut encadrer U+2009 ESPACE FINE ou U+2015 ― BARRE HORIZONTALE de U+FEFF pour les rendre insécables.

Espace sans chasse. U+200B ESPACE SANS CHASSE indique une frontière de mot, mais il est à chasse nulle. Les espaces sans chasse sont conçues pour les langues qui ne séparent pas les mots à l'aide d'espaces visibles, comme le thaï ou le japonais. Quand un texte est justifié, l'espace sans chasse n'a aucun effet sur l'interlettrage (cf. la tradition française ou japonaise).

Il se peut qu'à la suite d'une justification il faille avec certaines écritures, comme le thaï, blanchir un peu plus autour de l'espace sans chasse (voir la *Figure 14-1*). Cette méthode se distingue de l'utilisation de caractères à chasse fixe, comme U+2002 ESPACE DEMI-CADRATIN dont la chasse est constante et qu'il ne faut pas automatiquement élargir pendant la justification (voir la *Section 7.1*, *Ponctuation générale*).

Figure 14-1. Interlettrage

Type	Justification	Explication
Mémoire	the ISP®ZWSPCharts	On a inséré un ZWSP afin de permettre une coupure après le ®
Affichage 1	the ISP®Charts	Justification en pavé sans interlettrage
Affichage 2	t h e I S P ® C h a r t s	Interlettrage normal
Affichage 3	t h e I S P ® C h a r t s	Interlettrage à la thaïe
Affichage 4	t h e I S P ®C h a r t s	® empêche à tort l'interlettrage

Espaces à chasse nulle et caractères liants. Il ne faut pas confondre les espaces sans chasse et les caractères liants sans chasse. U+200C ZWNJ ANTILIANT SANS CHASSE et U+200D ZWJ LIANT SANS CHASSE n'affectent pas la frontière des mots, alors que ESPACE INSÉCABLE SANS CHASSE et ESPACE SANS CHASSE n'ont pas d'effet sur la liaison des lettres. En d'autres termes, il faut ignorer les liants sans chasse lors de l'établissement des frontières de mot alors que l'ESPACE SANS CHASSE doit être ignorée lors du calcul des formes liées. Voir *Liaison cursive* ci-dessous.

Séparateur de lignes et de paragraphes. Le standard Unicode prévoit deux caractères univoques U+2028 LSEP SÉPARATEUR DE LIGNES et U+2029 PSEP SÉPARATEUR DE PARAGRAPHES pour signaler la séparation des lignes et des paragraphes. On les considère comme les formes canoniques d'indication de frontière de ligne et de paragraphe dans les textes Unicode bruts. Une nouvelle ligne suit chaque SÉPARATEUR DE LIGNES, un paragraphe chaque SÉPARATEUR DE PARAGRAPHES. Ces caractères étant des séparateurs, il n'est pas nécessaire qu'un d'entre eux précède le premier paragraphe ou la première ligne ou encore suive le dernier paragraphe ou la dernière ligne. En effet, cette insertion signifierait qu'une ligne vide précède ou suit. On peut séparer les paragraphes d'un texte à l'aide de SÉPARATEUR DE PARAGRAPHES. Il est de la sorte possible de créer des fichiers de texte brut qui, à la réception, peuvent être composés avec des largeurs de ligne différentes selon la zone d'affichage disponible. Le SÉPARATEUR DE LIGNES peut servir à signaler une fin de ligne obligatoire.

Un séparateur de paragraphes indique le début d'un nouveau paragraphe. On pourra y effectuer la mise en page interparagraphe nécessaire. Ce formatage pourrait comprendre, par exemple, la coupure de la ligne courante, l'application de l'interlignage interparagraphe et le renfoncement de la première ligne du nouveau paragraphe.

Liaison cursive

Antiliant et liant. Les caractères adjacents de certaines écritures, écrits à l'aide de certaines polices, s'affichent sous la forme des glyphes reliés à leurs voisins de manière cursive et imitent ainsi l'écriture à la main courante. Ces liaisons cursives sont mises en œuvre dans la quasi-totalité des polices arabes, alors que pour les écritures latines elles ne le sont que pour les polices imitant l'écriture manuscrite.

Pour mettre en œuvre la liaison cursive, il faut relier les glyphes présents dans la police et sélectionner, à l'aide d'une fonction de contextualisation, la forme appropriée de chaque caractère en fonction de la nature jointive des caractères voisins. La logique de sélection est répartie, d'une certaine manière, entre le moteur de rendu et la police de caractères.

Quand on lie les lettres en cursive, il se peut qu'on veuille à l'occasion déroger à la sélection automatique habituelle des glyphes liés. D'ordinaire, cette dérogation a pour but :

- la sélection d'un œil lié non implicite (cas parfois obligatoire en persan écrit au moyen de l'écriture arabe) ;
- l'affichage en contexte isolé des différentes variantes d'œil lié.

Le standard Unicode permet d'ajuster le processus de sélection de glyphes liés à l'aide de deux caractères, U+200C ZWNJ ANTILIANT SANS CHASSE et U+200D ZWJ LIANT SANS CHASSE. Ces caractères ne modifient pas le processus de sélection contextuelle, ils changent plutôt le contexte d'un caractère particulier. L'insertion d'un caractère antiliant à la place d'un voisin liant, ou *vice versa*, permet de leurrer le processus de rendu et de lui faire choisir un autre glyphe lié. On peut utiliser cette méthode des deux manières décrites ci-dessous.

1. Empêcher la liaison. Ainsi, la suite

U+0635	ص	LETTRE ARABE ÇAD
U+200C	ZWNJ	ANTILIANT SANS CHASSE
U+0644	ل	LETTRE ARABE LAM

   s'affichera صل (en d'autres termes, la liaison cursive habituelle entre le *çad* et le *lam* n'a pas lieu). Sans antiliant, ces deux caractères auraient été affichés صلْ .

2. Afficher les glyphes jointifs de manière isolée. Ainsi, la suite

U+200D	ZWJ	LIANT SANS CHASSE
U+200C	غ	LETTRE ARABE GHAÏN
U+200D	ZWJ	LIANT SANS CHASSE

   s'affichera غ (en d'autres mots, la forme médiale du *ghaïn* apparaît dans un contexte isolé). Sans le LIANT SANS CHASSE qui le précède et le suit, le *ghaïn* aurait été affiché sous la forme isolée غْ .

Les exemples ci-dessus sont repris et adaptés de la norme nationale iranienne de jeux de caractères codés, ISIRI 3342. Ces caractères s'y nomment respectivement « pseudo-espace » et « pseudo-raccord ».

Comme le précise ce standard, le LIANT SANS CHASSE a des effets particuliers pour certaines écritures. Il fournit aux écritures brahmies, par exemple, un voisin invisible auquel une consonne morte peut se joindre afin de révéler une consonne mi-formée (voir la *Section 10.1*, *Dévanâgarî*). Il ne faut pas l'utiliser pour « coller » un élément textuel arbitraire à un autre, comme dans le cas de caractères soudés.

Le LIANT SANS CHASSE ou l'ANTILIANT SANS CHASSE sont des caractères de commande de formatage. Comme les autres caractères de cette catégorie, les processus d'analyse de contenu textuel doivent les ignorer. Un correcteur orthographique ou une opération *rechercher-remplacer* devront donc les éliminer (voir la *Section 2.7*, *Valeurs de non-caractères et de caractères spéciaux*).

L'effet de ces caractères à l'affichage dépend du contexte dans lequel ils apparaissent. Ajouter un LIANT SANS CHASSE entre deux caractères qui sont déjà liés de manière cursive

n'aura pas d'effet. Ajouter un ANTILIANT SANS CHASSE entre deux caractères qui ne sont pas reliés n'en aura pas plus. C'est ainsi qu'un nombre quelconque de LIANT SANS CHASSE ou d'ANTILIANT SANS CHASSE saupoudrés dans un texte français ne produira aucun changement à son apparence, s'ils sont rendus à l'aide d'une police latine non cursive typique.

Choisir les ligatures

Afin de pouvoir commander plus précisément la formation des ligatures, la définition des caractères suivants a été élargie, à partir d'Unicode 3.0.1 ; ils s'appliquent aussi bien aux ligatures qu'aux liaisons cursives.

U+200C ANTILIANT SANS CHASSE

- Ce caractère a pour but à la fois de dissocier les ligatures et d'empêcher la formation de liaisons cursives.

U+200D LIANT SANS CHASSE

- Ce caractère a pour but d'afficher, si possible, les caractères adjacents sous une forme plus liée qu'il ne l'aurait été en son absence. On trouvera ci-dessous plus de détails sur la sémantique du LSC (liant sans chasse, *ZWJ* en anglais).

    1. Si deux caractères peuvent former une ligature dans des cas inhabituels, un LSC indique qu'on désire obtenir cette ligature.

    2. Sinon, quand un des deux caractères peut se lier de manière cursive dans des cas inhabituels, le LSC indique qu'on désire que les deux caractères prennent une forme liée et cursive là où cela est possible.

        - Soit une suite <X, LSC, Y>, où une forme cursive existe pour X mais non pour Y, la présence d'un LSC a pour effet de demander la forme cursive de X.

    3. Enfin, si aucune ligature ni liaison cursive n'existent pour ces deux caractères, le LSC est sans effet.

En d'autres termes, si on considère les trois grandes catégories ci-dessous,

1. non relié,
2. lié de manière cursive,
3. ligaturé,

un LSC indique qu'on désire obtenir le glyphe appartenant à la catégorie la plus haute possible (pour une police donnée). Un ALSC (antiliant sans chasse, *ZWNJ* en anglais) indique qu'on désire obtenir le glyphe appartenant à la catégorie la plus basse possible (pour une police donnée).

Pour les situations inhabituelles où l'on veut empêcher la formation d'une ligature dans une suite <X,Y>, mais forcer une liaison cursive, on peut utiliser la suite <X, LSC, ALSC, LSC, Y>. L'ALSC empêche la formation d'une ligature, alors que les deux liants adjacents font en sorte que X et Y prennent des formes cursives liées adjacentes (si elles existent). De même, si on désire que X prenne une forme cursive mais que Y demeure isolée, on peut utiliser la suite <X, LSC, ALSC, Y>. Le tableau ci-après reprend quelques exemples.

Remarque : le LIANT SANS CHASSE (LSC, *ZNJ*) a des effets particuliers dans le cas des écritures brahmies. Voir la *Section 10.1*, *Dévanâgarî*.

Exemples. Le tableau ci-dessous illustre l'apparence désirée lorsqu'on insère un liant ou un antiliant entre deux caractères. Dans les exemples arabes, les caractères à gauche sont déjà dans l'ordre visuel, mais ne sont pas encore contextualisés. Ces exemples présupposent que tous les glyphes sont disponibles dans la police choisie. Si, par exemple, les ligatures sont absentes de la police, on choisira comme solution de repli les formes non ligaturées.

Tableau 14-1. Effets sur le rendu

Aucune	ZW NJ	ZW J ZW NJ ZW J	ZW J
fi ou fi	fi	fi	fi
لا	ل‌ا	ل‌ا	لا
مج ou FC45	م‌ج	مج	FC45 !
وج	و‌ج	و‌ج	و‌ج

Mise en œuvre. Les fondeurs doivent ajouter le LSC, le cas échéant, aux tables de ligatures de leurs polices modernes (de type Opentype ou AAT). Ainsi, si une police associait à « f » + « i » la ligature fi, le fondeur devrait également associer «f » + LSC + « i » à la ligature fi. En revanche, l'ALSC aura, en règle générale, l'effet recherché pour la plupart des polices sans qu'on doive les modifier, car il ne fait qu'empêcher la formation habituelle des ligatures ou des liaisons cursives. Comme pour tous les autres caractères de formatage, les polices devraient représenter l'ALSC et le LSC à l'aide d'un glyphe sans œil (invisible) et sans chasse.

Impact sur les données existantes. Les données actuelles ne contiennent guère de LSC entre les caractères qui se lient habituellement de manière cursive, car ce caractère aurait été simplement superflu à cet endroit dans les versions précédentes d'Unicode. Un tel LSC superfétatoire pourrait avoir produit une rupture de liaison cursive dans les mises en œuvre médiocres ; il est presque certain qu'aucun LSC sans effet sur la forme des caractères ne se trouve dans les données créées pour ces mises en œuvre. L'immense majorité des données existantes peut être rendue à l'aide des nouvelles mises en œuvre sans aucun changement apparent.

Impact sur les mises en œuvre existantes. Les algorithmes de rendu, avant Unicode 3.0.1, prennent en charge le LSC dans la mesure où celui-ci affecte la forme des caractères. Si cette mise en œuvre reçoit des données plus récentes, le LSC est sans effet ou, dans les mises en œuvre médiocres de l'algorithme de formage, il peut occasionner la rupture inopinée d'une liaison cursive. Toutefois, la présence du LSC n'a jamais été limitée à certains contextes, de sorte que même les anciens algorithmes devraient pouvoir le traiter de manière élégante.

Commandes bidirectionnelles

Ces codes servent dans l'algorithme bidirectionnel écrit au *Chapitre 3*, *Conformité*. Les systèmes qui traitent les écritures bidirectionnelles (arabe, hébreu ou syriaque) devraient être sensibles à ces codes. Le *Tableau 14-2* énumère ces codes.

Tableau 14-2. Commandes bidirectionnelles

N° Code	Nom	Abréviation
U+200E	MARQUE GAUCHE-À-DROITE	MGAD *(LRM)*
U+200F	MARQUE DROITE-À-GAUCHE	MDAG *(RLM)*
U+202A	ENCHÂSSEMENT GAUCHE-À-DROITE	EGAD *(LRE)*
U+202B	ENCHÂSSEMENT DROITE-À-GAUCHE	EDAG *(RLE)*
U+202C	DÉPILEMENT DE FORMATAGE DIRECTIONNEL	DFD *(PDF)*
U+202D	FORÇAGE GAUCHE-À-DROITE	FGAD *(LRO)*
U+202E	FORÇAGE DROITE-À-GAUCHE	FDAG *(RLO)*

Tout comme les autres codes à chasse nulle, ces caractères de mise en ordre bidirectionnel peuvent être ignorés par les logiciels de traitement, sauf pour leur effet sur la composition du texte où ils apparaissent. Les traitements autres que ceux de mise en page, comme le tri ou le repérage peuvent ignorer les caractères de mise en page sans chasse. Néanmoins, les opérations qui modifient du texte doivent conserver ces caractères intacts, car les séries couplées de caractères de formatage à chasse nulle doivent rester synchronisées (voir le *Chapitre 3*, *Conformité*).

U+200E MARQUE GAUCHE-À-DROITE et U+200F MARQUE DROITE-À-GAUCHE ont le même sens qu'un caractère invisible de chasse nulle, si ce n'est que ces caractères ont une forte directionalité. Ils sont conçus pour dissiper les incertitudes quant à la directionalité de certains passages dans les textes bidirectionnels. Contrairement à U+200B ESPACE SANS CHASSE, ces caractères n'ont aucun impact sur la coupure de mots (voir la *Section 3.12*, *Comportement bidirectionnel*, pour plus de renseignements).

14.3 Caractères de formatage à éviter

Caractères de formatage à éviter : U+206A – U+206F

Ce bloc comprend trois paires de caractères de formatage à éviter[2].

- Les caractères d'échange symétrique qui servent à désigner les glyphes représentant des caractères comme « (» . Leur état[3] implicite est *activé*.
- Les sélecteurs de formes de caractères qui servent à commander la contextualisation des caractères arabes de compatibilité. Leur état implicite est *inhibé.*
- Les sélecteurs de formes numérales qui permettent de changer la forme habituelle des chiffres occidentaux. Leur état implicite est *de référence*.

Unicode déconseille fortement l'utilisation de ces sélecteurs de formes. On utilisera à leur place les caractères appropriés avec l'état implicite mentionné ci-dessus. Ainsi, si on désire obtenir les formes contextuelles des lettres arabes, on utilisera les caractères de référence et non les formes de présentation et leurs sélecteurs de formes. De même, si on veut afficher les chiffres arabes[4], on choisira explicitement les chiffres arabes (par exemple, U+0660 ٠ CHIFFRE ARABE-HINDI ZÉRO).

Échange symétrique. Les caractères d'échange symétrique s'utilisent avec les caractères symétriques (encore appelés caractères spéculaires ou miroirs) comme les parenthèses ou les accolades. La *Section 4.7*, *Caractères miroirs – normatif*, énumère les caractères concernés. Les caractères d'échange symétrique précisent si les termes DROITE et GAUCHE dans les noms de caractères signifient respectivement *ouvrant* et *fermant*. Ils ne s'imbriquent pas. Un protocole ou une norme de niveau supérieur (comme l'ISO 6429) peuvent préciser l'état implicite d'échange symétrique. En l'absence d'un tel protocole, l'état implicite est activé.

À partir du caractère de formatage U+206A INHIBITEUR D'ÉCHANGE SYMÉTRIQUE jusqu'à l'U+206B ACTIVATEUR D'ÉCHANGE SYMÉTRIQUE suivant (le cas échéant), on interprète et affiche les caractères symétriques sous leurs formes gauche et droite.

À partir du caractère de formatage U+206B ACTIVATEUR D'ÉCHANGE SYMÉTRIQUE jusqu'à l'U+206A INHIBITEUR D'ÉCHANGE SYMÉTRIQUE suivant (le cas échéant), on interprète et affiche les caractères symétriques sous leurs formes ouvrante et fermante. Cet état *(activé*) est l'état implicite en absence de tout code d'échange symétrique ou d'un protocole de niveau supérieur.

Sélecteur de formes de caractères. Les caractères d'échange symétrique s'utilisent avec les formes de présentation arabes. Lors du processus de rendu, certaines lettres peuvent former des ligatures ou des liaisons cursives. Les codes de sélection de formes indiquent si le processus de choix des formes de caractères (la sélection de glyphe) doit être activé ou inhibé. Les sélecteurs de formes ne s'imbriquent pas.

À partir du caractère U+206C INHIBITEUR DE FORMAGE ARABE jusqu'au U+206D ACTIVATEUR DE FORMAGE ARABE suivant (le cas échéant), le processus de formage des caractères est bloqué. Si la mémoire contient des formes de présentation arabes (U+FE80..U+FEFC), on affichera ces caractères sans les modifier. Cet état *(inhibé*) est l'état implicite en absence de tout code de sélection de formes de caractère ou d'un protocole de niveau supérieur.

[2] Il faut bien distinguer les caractères à éviter des caractères désuets. Les premiers sont fortement déconseillés alors que les seconds ne s'utilisent plus. La désuétude d'un caractère dépend de son contexte, la lettre *grand yousse* est désuète en russe, mais est toujours utilisée en bulgare moderne.

[3] On associe à chaque paire de caractères un état binaire qui détermine la forme des caractères subséquents.

[4] Plus précisément les chiffres arabo-hindis et non nos chiffres européens, communément appelés arabes.

À partir du caractère U+206D ACTIVATEUR DE FORMAGE ARABE jusqu'au U+206C INHIBITEUR DE FORMAGE ARABE suivant (le cas échéant), toutes les formes de présentation arabes présentes en mémoire devront être affichées à l'aide du processus de formage des caractères (la contextualisation et la sélection de glyphe).

Les sélecteurs de formes n'ont pas d'effet sur les caractères arabes de référence (U+0660..U+06FF) qui sont toujours contextualisés et ignorent ces caractères de formatage.

Sélecteur de formes numérales. Les codes de sélection de formes numérales permettent de préciser la forme que devraient prendre les chiffres U+0030..U+0039. Ces caractères de formatage ne s'imbriquent pas.

À partir du caractère U+206E SÉLECTEUR DE FORMES NUMÉRALES NATIONALES jusqu'au U+206F SÉLECTEUR DE FORMES NUMÉRALES DE RÉFÉRENCE suivant (le cas échéant), on affichera les chiffres européens (U+0030..U+0039) sous une forme nationale précisée par des accords appropriés. Ils pourraient, par exemple, être affichés sous la forme de chiffres arabo-hindis (U+0660..U+0669). La forme précise des caractères (les glyphes) utilisée pour rendre les chiffres nationaux n'est pas précisée par le standard Unicode.

À partir du caractère U+206F SÉLECTEUR DE FORMES NUMÉRALES DE RÉFÉRENCE jusqu'au U+206E SÉLECTEUR DE FORMES NUMÉRALES NATIONALES suivant (le cas échéant), on affichera les chiffres européens (U+0030..U+0039) à l'aide de glyphes qui illustrent les formes numérales de référence dans les tableaux de caractères. Cet état *(de référence*) est l'état implicite en absence de tout code de sélection de formes numérales ou d'un protocole de niveau supérieur.

14.4 Zone d'indirection

Zone d'indirection : U+D800 – U+DFFF

La zone d'indirection se compose des 1 024 valeurs de code d'indirection inférieures (aussi appelées seizets d'indirection inférieurs) et des 1 024 valeurs de code d'indirection supérieures (seizets d'indirection supérieurs). Ces seizets d'indirection, interprétés par paire, permettent d'accéder en UTF-16[5] à plus d'un million de points de code appartenant aux *plans complémentaires*[6]. Pour une définition formelle des *paires d'indirection* et leur rôle dans les clauses de conformité Unicode, veuillez consulter les sections *3.7*, *Seizets d'indirection*, et *5.4*, *Traitement des paires d'indirection*.

Demi-zone haute. Les seizets d'indirection supérieurs sont affectés à l'intervalle U+D800..U+DBFF. Le premier élément de chaque paire d'indirection correspond à un seizet d'indirection supérieur.

Demi-zone basse. Les seizets d'indirection inférieurs sont affectés à l'intervalle U+DC00..U+DFFF. Le second élément de chaque paire d'indirection correspond à un seizet d'indirection inférieur.

Seizets d'indirection supérieurs à usage privé. Les seizets d'indirection supérieurs de l'intervalle U+DB80..U+DBFF (128 valeurs de codes) sont à usage privé. Les caractères représentés à l'aide de paires d'indirection, dont le seizet supérieur est à usage privé, sont des caractères à usage privé. Ce mécanisme permet de représenter 131 068 (= 128 x 1024 – 4) valeurs de code à usage privé. Pour plus de renseignements sur les caractères à usage privé, veuillez consulter la section consacrée ci-dessous à la zone à usage privé.

Aucun tableau de glyphes et aucun nom de caractères n'est associé à la zone U+D800..U+DFFF, car les seizets non appariés ne représentent pas des caractères, mais de simples unités de stockage.

[5] Ces seizets d'indirection ne sont pas utiles en UTF-32 (ou UCS-4) où ces valeurs sont directement accessibles.
[6] Plans aux valeurs de code supérieures à U+FFFF ; depuis Unicode 3.1 des valeurs publiques y sont affectées.

14.5 Zone à usage privé

Zone à usage privé : U+E000 – U+F8FF

La zone à usage privé est réservée aux programmeurs et aux utilisateurs qui ont besoin d'un jeu de caractères particulier. Les points de code de cette zone sont réservés à un usage privé, ils n'ont pas de sémantique précise interprétable si ce n'est par accord entre les parties concernées.

Il n'existe pas de tableaux de caractères pour cette zone, car les caractères codés dans cette zone sont définis de gré à gré.

Des valeurs scalaires d'Unicode (points de code) appartenant aux plans complémentaires (U+F0000..U+FFFFD et U+100000..U+10FFFD[7]) sont également réservés à l'usage privé. (Cf. la définition D12 à la *Section 3.4*, *Propriétés simples*.) En UTF-16, on y accède à l'aide des seizets d'indirection (voir la section précédente).

Structure de codage. On divise par convention la zone à usage privé en une sous-zone réservée aux entreprises (elle commence à U+F8FF et s'étend vers le bas de la zone) et une sous-zone réservée aux utilisateurs (elle commence à U+E000 et s'étend vers le haut de la zone).

Sous-zone dédiée aux entreprises. Il se peut que les fabricants de systèmes et les développeurs de logiciels aient besoin de caractères privés pour leurs propres logiciels. La sous-zone dédiée aux entreprises est le meilleur endroit pour ce faire. L'affectation d'une sémantique aux caractères de cette sous-zone peut être entièrement confidentielle et cachée aux utilisateurs ou, au contraire, être publiée sous la forme d'affectations de caractères propres à un fabricant mises à la disposition des programmeurs et des utilisateurs. Dans le premier cas, on peut imaginer l'affectation d'un numéro de caractère à une opération de soutien du système comme <DÉPLACER> ou <COPIER> ; dans le second cas, il se peut qu'on affecte un code à un caractère logotype comme le caractère *pomme* d'Apple.

Sous-zone dédiée aux utilisateurs. La sous-zone dédiée aux utilisateurs est destinée aux définitions de caractères effectuées par les utilisateurs ou comme zone d'affectation temporaire de caractères par les applications**.**

Attribution des sous-zones. Les fabricants peuvent décider de réserver des codes privés dans la sous-zone réservée aux entreprises et de mettre une portion particulière de la sous-zone réservée aux utilisateurs entièrement à la disposition des utilisateurs. Cette convention existe par commodité pour les fabricants de systèmes et des concepteurs de logiciels. Aucune ligne de démarcation fixe n'est définie entre les deux sous-zones, car des utilisateurs différents peuvent avoir des besoins différents. Unicode ne prévoit pas de mécanisme pour éviter une « collision pile-tas» entre les deux sous-zones de la zone à usage privé.

Promotion des caractères à usage privé. Il se peut que des versions ultérieures du standard Unicode codent parmi les caractères normaux des caractères définis auparavant par un fabricant dans la sous-zone réservée aux entreprises, si leur utilisation est devenue assez répandue. Les positions de code dans cette zone sont définitivement réservées à un usage privé – le consortium Unicode n'avalisera jamais l'affectation de ces points de code à un jeu de caractères particulier.

[7] Depuis Unicode 3.1.

14.6 Caractères spéciaux

Caractères spéciaux : U+FEFF, U+FFF0 – U+FFFF

Le bloc des caractères spéciaux se compose des valeurs de code qui ne correspondent ni à des commandes ni à des caractères graphiques, mais qui permettent de mettre en œuvre des usages logiciels bien établis. Les 14 valeurs Unicode U+FFF0..U+FFFD sont réservées à la définition de caractères spéciaux. En sus de ces 14 positions, ce bloc comprend également 2 valeurs de codes par plan qui ne sont pas des caractères mais des indicateurs (voir ci-dessous).

Indicateur d'ordre des octets (IOO, en anglais *BOM*)

Il existe deux cas où les caractères U+*n*FEFF (où *n* est compris entre 0 et 10_{16}) ont un sens particulier qui les différencient de l'espace insécable sans chasse (voir la *Section 14.2*, *Commandes de mise en page*) :

1. Ordre des octets non précisé. Certaines architectures d'ordinateur rangent les octets dans ce qu'il est convenu d'appeler l'ordre *gros-boutien* ; d'autres dans l'ordre *petit-boutien*. Selon l'architecture, un texte Unicode peut aussi bien se sérialiser en ordre gros-boutien que petit-boutien. Toutefois, cet ordre de sérialisation n'est pas toujours signalé ce qui pose problème dans l'échange de données entre des systèmes d'architectures différentes.

2. Jeu de caractères non précisé. Parfois, le jeu de caractères utilisé dans un flux de caractères codés (comme dans un fichier) n'est pas précisé. On sait seulement que le flux comprend du texte, mais le jeu de caractères précis est inconnu.

Dans ces deux cas, on peut se servir des caractères U+*n*FEFF comme d'une signature pour indiquer l'ordre des octets et le jeu de caractères en utilisant la sérialisation UTF-16 décrite à la *Section 3.8*, *Transformations*. Sa version à octets inversés U+FFFE est toujours une valeur Unicode illégale ; on peut donc, quand on est confronté à cette valeur, en conclure que le fichier a été lu dans le mauvais ordre de sérialisation ou que le fichier n'est pas un texte Unicode bien formé.

En sérialisation UTF-16, U+FEFF dès le début d'un fichier ou d'un flux indique explicitement l'ordre des octets.

On peut se servir de la suite d'octets $FE_{16}FF_{16}$ comme d'une signature indiquant que le fichier contient du texte Unicode. En effet, cette suite est excessivement rare au début des fichiers de texte qui utilisent d'autres codages, qu'il s'agisse de jeux à un, à deux ou à multiples octets ; on ne risque donc pas d'être induit en erreur par de vraies données textuelles. Ainsi, pour les systèmes qui utilisent l'ISO Latin-1 (ISO/CEI 8859-1) ou la page de code 1252 ANSI de MS Windows la suite d'octets $FE_{16}FF_{16}$ correspond à "þÿ" (*thorn, y tréma*) ; pour les systèmes qui emploient le jeu de caractères romain Macintosh d'Apple ou le codage standard d'Adobe, cette suite représente "¸ ˇ" (*ogonek, hacek*) ; enfin pour les systèmes qui utilisent d'autres pages de code habituelles IBM PC (par exemple, CP 437 ou CP 850), cette suite correspond à "■. " (*carré noir, espace insécable*).

En UTF-8, l'IOO (l'indicateur d'ordre des octets) correspond à la suite d'octets $EF_{16}BB_{16}BF_{16}$. Bien que la sérialisation ne pose pas de problème en UTF-8, cette suite peut servir à détecter le jeu de caractères quand celui-ci n'est pas explicitement signalé. Tout comme dans le cas de l'IOO en UTF-16, cette suite est extrêmement rare au début des fichiers de texte codés dans les autres jeux de caractères. Ainsi, dans les systèmes qui emploient la page de code 1252

ANSI de MS Windows, la suite $EF_{16}BB_{16}BF_{16}$ correspond-elle à la suite "ï«¿" (*i tréma, guillemet ouvrant, point d'interrogation renversé*).

Quand le jeu de caractères est explicitement indiqué, comme c'est le cas pour UTF-16BE ou UTF-16LE, on interprète alors tous les caractères U+FEFF, même ceux au début du texte, comme des *espaces insécables sans chasse*. De même quand on connaît l'ordre de sérialisation d'un texte Unicode, il n'est pas nécessaire d'insérer un U+FEFF au début d'un texte et ceux qui s'y trouveraient doivent être interprétés comme des *espaces insécables sans chasse*. Pour les chaînes d'une interface de programmation (une API), par exemple, l'architecture de la mémoire du processeur fournit l'ordre explicite des octets, il n'est donc pas nécessaire d'utiliser un IOO. Pour les bases de données et les structures similaires, il est nettement plus efficace et prudent d'utiliser une sérialisation uniforme pour un même champ (voire la base de données en entier), évitant de la sorte d'avoir recours à un indicateur d'ordre des octets.

Les systèmes qui emploient l'*indicateur d'ordre des octets* doivent interpréter un U+FEFF initial comme un indicateur de sérialisation, il ne fait pas partie du contenu textuel. Il faut l'éliminer avant tout traitement, sinon il pourrait être interprété comme une *espace insécable sans chasse*. Pour représenter une U+FEFF ESPACE INSÉCABLE SANS CHASSE initiale dans un fichier UTF-16, utilisez deux U+FEFF l'un après l'autre. Le premier sert d'*indicateur d'ordre des octets* ; le second est l'*espace insécable sans chasse* initiale. Voir le *Tableau 14-3*, pour un résumé des signatures des formes de stockage Unicode.

Tableau 14-3. Signatures des formes de stockage Unicode

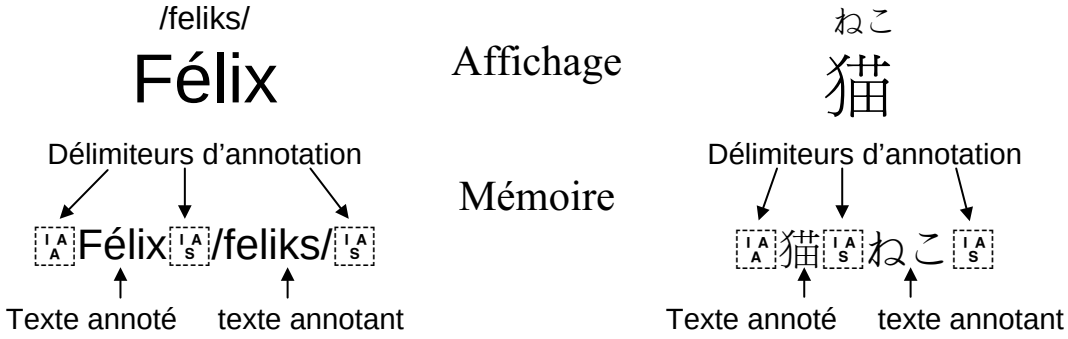
Forme de stockage	Signature
UTF-8	EF BB BF
UTF-16 gros-boutien	FE FF
UTF-16 petit-boutien	FF FE
UTF-32 gros-boutien	00 00 FE FF
UTF-32 petit-boutien	FF FE 00 00

Délimiteurs d'annotation

Une *annotation interlinéaire* est composée d'un texte annotant (l'annotation même) et d'une série correspondante de caractères annotés. Pour toutes les opérations de traitement de texte ordinaires, on considère les caractères annotés comme faisant partie à part entière du texte. L'annotation fait également partie du contenu, cependant pour certaines — voire toutes — les opérations de traitement de texte, on considère qu'elle ne fait pas partie du corps du texte. Toutefois, les caractères annotants peuvent être mis en page et édités de la même manière que le texte de base. Les délimiteurs d'annotation séparent le texte annoté du texte annotant et identifient ces textes comme faisant partie d'une annotation. Voir la *Figure 14-2*.

Les délimiteurs d'annotation s'utilisent quand, lors du traitement interne, on associe à ce flux de caractères des informations *hors bande* (ou *hors texte*), une situation analogue à l'utilisation du U+FFFC CARACTÈRE DE REMPLACEMENT D'OBJET. Néanmoins, contrairement aux objets opaques cachés par le caractère de remplacement d'objet, les annotations sont toujours textuelles.

Figure 14-2. Délimiteurs d'annotation



Conformité. Une mise en œuvre conforme prenant en charge les délimiteurs d'annotation est tenue d'interpréter le texte de base comme s'il faisait partie d'un flux textuel non annoté. Cette mise en œuvre doit interpréter l'annotation même (le texte annotant) à la lumière de la sémantique Unicode habituelle des caractères la constituant.

U+FFF9 ANCRE D'ANNOTATION INTERLINÉAIRE représente un repère de début d'annotation. La nature et le formatage précis de l'annotation dépendent d'informations supplémentaires qui ne font pas partie du texte brut. Cette situation est analogue à celle du U+FFFC CARACTÈRE DE REMPLACEMENT D'OBJET.

U+FFFA SÉPARATEUR D'ANNOTATION INTERLINÉAIRE sépare les caractères de base (annotés) des caractères annotants subséquents. L'interprétation précise de ce séparateur dépend de la nature de l'annotation. Plus d'un séparateur peut être présent, chacun délimitant une annotation supplémentaire.

U+FFFB TERMINATEUR D'ANNOTATION INTERLINÉAIRE met fin à un objet d'annotation ; on revient au flux textuel normal.

Utilisation dans les textes bruts. L'utilisation des délimiteurs d'annotation dans les textes bruts est fortement déconseillée sans accord préalable entre l'expéditeur et le destinataire car, sinon, le contenu risque d'être mal interprété. L'élimination pure et simple des délimiteurs produira un texte illisible ou, pire, un contresens. À la réception, un destinataire de texte brut doit conserver tous les caractères ou éliminer à la fois les délimiteurs d'annotation et les caractères annotants compris entre le U+FFFA SÉPARATEUR D'ANNOTATION INTERLINÉAIRE et le U+FFFB TERMINATEUR D'ANNOTATION INTERLINÉAIRE.

Quand un expéditeur désire transmettre un texte brut annoté et que le destinataire lui est inconnu, il doit ôter les délimiteurs d'annotation ainsi que le texte annotant compris entre le U+FFFA SÉPARATEUR D'ANNOTATION INTERLINÉAIRE et le U+FFFB TERMINATEUR D'ANNOTATION INTERLINÉAIRE.

Cette restriction n'exclut pas l'utilisation de délimiteurs d'annotation dans les échanges de textes bruts, mais elle exige une entente préalable entre le récepteur et le destinataire afin que les annotations soient bien interprétées.

Restrictions lexicales. Si une coupure de paragraphe survient entre une *ancre* et son *terminateur* correspondant, celle-ci met fin à toute annotation encore ouverte. Les ancres doivent précéder leurs terminateurs correspondants. Les ancres et terminateurs non appariés doivent être ignorés. Les séparateurs d'annotation qui se présentent à l'extérieur d'une paire de délimiteurs ne doivent pas rentrer en ligne de compte. Les annotations peuvent être imbriquées.

Formatage. Toute l'information de formatage destinée aux annotations est fournie par des protocoles de niveau supérieur. Les détails de la mise en page des annotations dépendent de la mise en œuvre. Il est possible que la mise en page correcte des annotations nécessite des renseignements supplémentaires qui ne font pas partie du flux de caractères, mais qui sont stockés hors texte. Les caractères d'annotation servent alors de repères pour une mise en œuvre ayant accès à l'information nécessaire à partir d'une autre source.

Tri. Sauf dans le cas particulier où une annotation servirait de clé de tri, on ignore d'ordinaire le texte annotant lors des tris ou, facultativement, on ne l'utilise qu'en cas d'égalité comme clé subsidiaire. Il est important de noter que les caractères de base (annotés) ne sont pas ignorés, mais qu'on les traite comme le reste du texte.

> **Remarque :** dès leur conception, les délimiteurs d'annotation n'étaient pas destinés à être échangés entre processus, mais devaient servir de repères temporaires internes. Depuis la version 3.1, il existe une bien meilleure manière d'aborder le problème de l'annotation d'un texte et d'autres semblables comme les *Caractères de remplacement* : 32 positions consécutives du PMB ont été désignées comme « non-caractères » (voir ci-dessous). Par définition, ces caractères ne pouvant jamais faire partie d'un texte échangé avec un processus externe, ils peuvent donc servir de repère sans qu'on puisse les confondre avec des données[8].
>
> Employer les délimiteurs d'annotation dans un texte brut pose problème : les processus d'affichage existants pourraient les ignorer ou les remplacer et changer ainsi le sens du texte.
>
> L'utilisation des délimiteurs d'annotation dans un texte balisé est également problématique, car l'information de formatage (où placer l'annotation, par exemple) ne fait pas partie des annotations Unicode. Dans le cas de texte XML, il est préférable d'utiliser les balises définies à cet effet[9].

Caractères de remplacement.

U+FFFC. U+FFFC CARACTÈRE DE REMPLACEMENT D'OBJET indique dans le flux textuel le point où insérer un objet. Tous les renseignements reliés à cet objet se situent en dehors du flux des données de caractères. Ce caractère factice joue le rôle d'un repère ou d'un pointeur vers l'information de formatage qui se rapporte à cet objet. En plus de permettre le placement correct d'un objet dans le flux de données, le caractère de remplacement d'objet permet d'utiliser, pour tous les aspects textuels des objets imbriqués, des algorithmes généraux prévus pour des flux.

> **Remarque :** la remarque de la sous-section consacrée aux délimiteurs d'annotation ci-dessus s'applique également aux caractères de remplacement d'objet : il vaut mieux utiliser des non-caractères ou des balises appropriées.

U+FFFD. U+FFFD CARACTÈRE DE REMPLACEMENT est le caractère de substitution général Unicode. Ce caractère peut remplacer tout caractère « inconnu » dans un autre codage et qui ne peut être transformé en une valeur de code Unicode (voir la *Section 5.3*, *Caractères inconnus ou manquants*).

[8] Voir <http://www.unicode.org/unicode/reports/tr20/tr20.html>.
[9] Voir <http://www.w3.org/TR/ruby/>.

Non-caractères

U+*n*FFFE. Les valeurs U+*n*FFFE (où *n* est compris entre 0 et 10_{16}) ne représentent pas des valeurs de caractère Unicode. Leur présence dans un flux de données Unicode suggère fortement que les octets doivent être inversés avant toute interprétation. U+*n*FFFE ne peut être interprété que comme un U+FEFF ESPACE INSÉCABLE SANS CHASSE (indicateur d'ordre des octets) dont les octets sont inversés (c'est-à-dire sont mal sérialisés).

U+*n*FFFF. Les valeurs U+*n*FFFF (où *n* est compris entre 0 et 10_{16}) ne représentent pas des valeurs de caractère Unicode ; elles peuvent être utilisées par une application comme un code d'erreur interne ou un autre non-caractère. Ces valeurs ne peuvent jamais être échangées avec un autre processus. Unicode n'attribue pas de sens particulier à ces valeurs, on peut donc les considérer comme des non-caractères d'usage privé.

U+FDD0..U+FDEF. Unicode 3.1 a déclaré trente-deux nouvelles positions comme non-caractères. Voir la définition D7b de la *Section 3.1*, *Exigences de conformité*.

14.7 Étiquettes

Étiquettes : U+E0000 – U+E007F

Les caractères de ce bloc fournissent un mécanisme d'étiquetage linguistique au sein des textes bruts Unicode. *Toutefois, leur utilisation est fortement déconseillée.* Les caractères de ce bloc doivent s'utiliser avec des protocoles particuliers. Il est interdit de les employer en l'absence de ces protocoles ou avec n'importe quel protocole qui fournirait un autre mécanisme d'étiquetage linguistique, comme HTML ou XML. On exagère souvent le besoin d'insérer des informations linguistiques dans les données textuelles brutes. Voir la *Section 5.11*, *Étiquettes linguistiques et l'unification han*.

Ce bloc comprend un jeu de 95 caractères d'étiquetage spéciaux qui permettent d'écrire en toutes lettres des étiquettes linguistiques à l'aide de caractères que l'on peut sans ambiguïté distinguer des caractères utilisés pour coder le contenu textuel habituel. Ces étiquettes peuvent être insérées dans le texte brut par d'autres protocoles. Les mises en œuvre utilisant des algorithmes triviaux peuvent facilement les identifier ou les ignorer, car ces caractères ne s'utilisent qu'en tant qu'étiquettes et jamais en tant que contenu textuel.

Outre ces 95 caractères, ce bloc inclut également un identificateur d'étiquette linguistique et un caractère d'annulation d'étiquette. Comme son nom l'indique, l'identificateur d'étiquette linguistique précise qu'une chaîne d'étiquetage est de type linguistique ; l'étiquette linguistique même se sert des indicatifs linguistiques du RFC 3066 (ou de ses successeurs) en les écrivant en toutes lettres à l'aide des étiquettes linguistiques de ce bloc.

Cinq termes (*balise*, *étiquette*, *annotation*, *hors texte*, *à même le texte*) à l'acception particulière ont été ajoutés à l'*Annexe G*, *Glossaire*.

Syntaxe des étiquettes imbriquées

Afin de pouvoir imbriquer n'importe quelle étiquette dérivée de l'ASCII au sein d'un texte brut Unicode, on écrit l'étiquette linguistique en toutes lettres à l'aide des caractères-étiquettes correspondants, préfixés de l'identificateur d'étiquette approprié. La chaîne produite de la sorte est directement insérée dans le texte.

Identificateur d'étiquette. L'identificateur d'étiquette permet de distinguer différents types d'étiquette. À l'avenir, ce mécanisme devrait permettre la coexistence et l'imbrication de différents types d'étiquette dans un texte brut.

Fin d'étiquette. Aucun caractère de fin d'étiquette n'est nécessaire pour l'étiquette même car tous les caractères qui forment l'étiquette ont un numéro distinct des caractères habituels (non-étiquettes). Une étiquette se termine au premier caractère non-étiquette (c.-à-d. toute autre valeur Unicode ordinaire) ou au prochain identificateur d'étiquette. Une grammaire BNF (forme Backus-Naur) décrit ci-dessous la syntaxe détaillée des étiquettes.

Étiquettes linguistiques. On définit une étiquette linguistique comme étant formée d'une U+E0001 ÉTIQUETTE LINGUISTIQUE suivie d'une chaîne de caractères-étiquettes. Au demeurant, les valeurs d'étiquette linguistique[10] doivent s'écrire en toutes lettres conformément au RFC 3066 et ne recourir qu'à des valeurs d'étiquette agréées ou à des valeurs privées préfixées par les caractères « x- ».

10 En d'autres termes, la langue désignée.

Considérons, par exemple, l'insertion d'une étiquette linguistique désignant le japonais. L'indicatif RFC 3066 pour le japonais est « ja » (conformément à l'indicatif linguistique de l'ISO 639) ou encore « ja-JP » (indicatif linguistique ISO 639 suivi de l'indicatif de pays ISO 3166). Le RFC 3066 précise que les indicatifs linguistiques ne sont pas sensibles à la casse, on conseille donc leur mise en minuscules avant de les convertir en caractères d'étiquette linguistique.

Ainsi l'indicatif complet « ja-JP » sera converti dans les caractères d'étiquette suivants :

U+E0001 U+E006A U+E0061 U+E002D U+E006A U+E0070

L'étiquette linguistique, dans sa forme abrégée « jp », s'exprimera de la manière suivante :

U+E0001 U+E006A U+E0061

Portée et imbrication des étiquettes. La valeur d'une étiquette prend effet à partir du point où l'étiquette est insérée dans le texte jusqu'à ce que :

A. le texte lui-même sorte de portée pour l'application (c'est-à-dire, pour des protocoles orientés ligne, à la fin d'une ligne ou d'une chaîne ; pour les flux textuels à la fin d'un flux, etc.)

ou

B. l'étiquette est explicitement annulée à l'aide d'un caractère U+E007F ÉTIQUETTE D'ANNULATION.

Les étiquettes de même type ne peuvent en aucune façon s'imbriquer. Ainsi, une étiquette linguistique se présentant dans un texte déjà étiqueté linguistiquement remplace l'étiquette précédente pour le texte subséquent.

Des étiquettes de type différent peuvent s'entremêler, elles ne forment pas pour autant une hiérarchie. En effet, les étiquettes de types différents s'ignorent mutuellement, de telle sorte que l'utilisation des étiquettes linguistiques est totalement indépendante de celle d'éventuels nouveaux types d'étiquette.

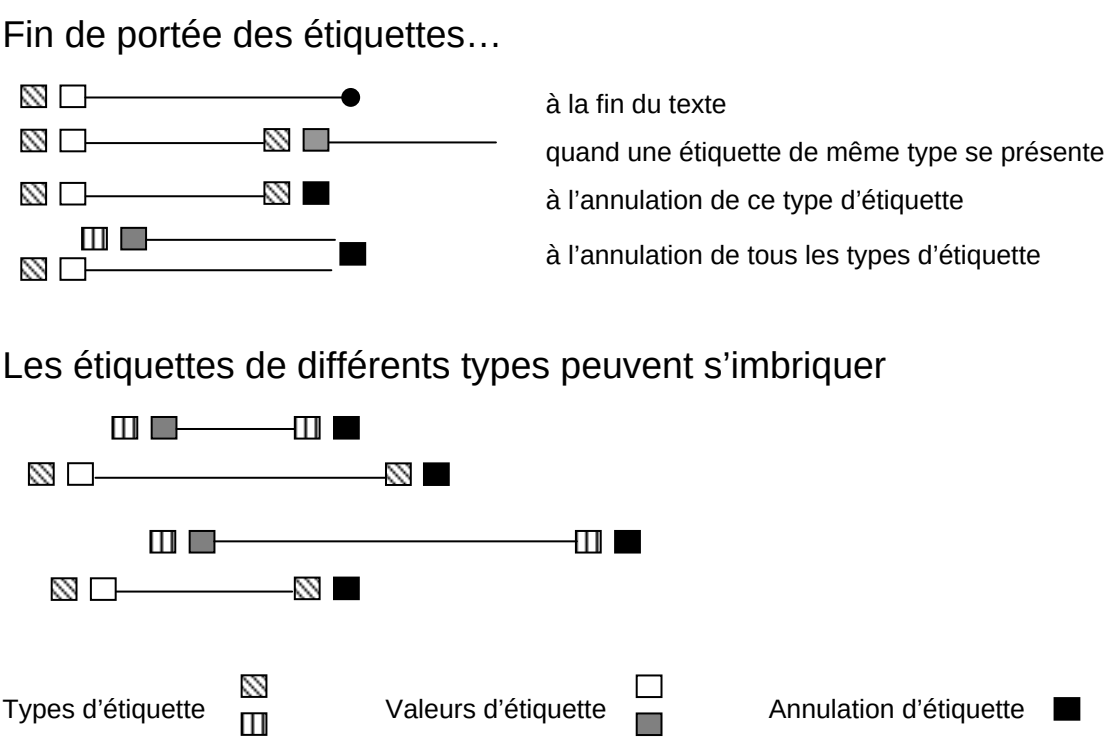
Annuler des valeurs d'étiquette. La fonction principale de l'ÉTIQUETTE D'ANNULATION est de permettre des opérations comme la concaténation aveugle de chaînes dans un contexte étiqueté sans propagation inopinée des valeurs d'étiquette au-delà des frontières de chaîne. L'ÉTIQUETTE D'ANNULATION connaît deux emplois.

Pour annuler une valeur d'étiquette d'un type particulier, préfixez l'ÉTIQUETTE D'ANNULATION de l'identificateur d'étiquette du type approprié. Par exemple, la chaîne complète pour annuler une étiquette linguistique est U+E0001 U+E007F. Le type d'étiquette en question reprend alors sa valeur implicite, en d'autres mots : aucune valeur d'étiquette définie, à l'instar d'un texte non étiqueté.

Pour annuler toutes les valeurs d'étiquette en vigueur, utilisez l'ÉTIQUETTE D'ANNULATION sans la préfixer d'un identificateur d'étiquette.

> **Remarque.** À l'heure actuelle, il n'existe pas de différence visible entre les deux emplois de l'ÉTIQUETTE D'ANNULATION, car un seul identificateur d'étiquette (et donc un seul type d'étiquette) est défini. N'insérer qu'une simple ÉTIQUETTE D'ANNULATION, quand on ne désire qu'annuler l'étiquetage linguistique, pourrait entraîner des effets de bord inattendus à l'avenir si ce texte devait être inséré dans un texte qui prend en charge plus d'un type d'étiquette.

Figure14-3. Étiquettes



Utilisation des étiquettes linguistiques

Éviter l'utilisation d'étiquette linguistique. Les textes bruts doivent éviter la charge de mise en œuvre supplémentaire associée aux étiquettes linguistiques à moins que cette information soit cruciale et que l'on soit assuré que les destinataires de ces textes interpréteront correctement et conserveront ces étiquettes. Néanmoins, lorsque ces étiquettes s'imposent, il faut alors se pencher sur les problèmes suivants liés à leur utilisation et décider comment traiter celles-ci quand elles ne sont pas complètement prises en charge. Cet exposé vaut pour tous les mécanismes de balisage linguistique inséré dans des textes bruts.

Protocoles de niveau supérieur. Il faut s'abstenir d'insérer des étiquettes linguistiques dans le texte Unicode si un protocole de niveau supérieur, comme XML ou MIME, fournit déjà cette information. De la sorte, on prévient non seulement des complications inutiles, mais on évite également les conflits entre les étiquettes linguistiques du texte et les étiquettes (ou balises) linguistiques du protocole de niveau supérieur. Voir le *Rapport technique d'Unicode* n° 20 « *Unicode in XML and other Markup Languages* ».

Impact des étiquettes sur l'interprétation des textes. Les mises en œuvre qui prennent en charge les étiquettes linguistiques seront peut-être tenues de les prendre en compte pour certains traitements, comme la coupure de mot ou le choix de police. Toutefois, les étiquettes elles-mêmes n'ont pas d'œil et n'ont aucun effet sur la coupure de ligne, sur la forme ni sur la liaison des caractères ou sur toute autre propriété de formatage ou de mise en page. Tout processus qui interprète une étiquette peut décider d'imposer un comportement particulier en fonction de la valeur de cette étiquette.

Rendu. Les étiquettes linguistiques ne s'affichent pas. Cette décision ne signifie pas nécessairement qu'il faille modifier le programme de rendu si les polices de la plate-forme en question associent les caractères d'étiquetage linguistique à des glyphes sans chasse, c'est-à-dire invisibles. Pour la mise au point ou pour d'autres opérations qui nécessitent l'affichage des

étiquettes sous une forme visible, il est conseillé de rendre les caractères d'étiquette à l'aide de leurs glyphes ASCII correspondants (peut-être modifiés systématiquement afin de les différencier des caractères ASCII normaux). On a cependant choisi les valeurs de caractères étiquettes de telle sorte que les caractères d'étiquette soient interprétables par la plupart des débogueurs sans qu'ils ne doivent être affichés.

Traitement. D'ordinaire, l'accès séquentiel au texte est facile. Si une étiquette linguistique ne s'applique pas à une opération précise, il faut alors l'ignorer. Dans le cas d'un accès direct, la nature des étiquettes pose problème. En effet, l'état linguistique d'un passage dépendant des étiquettes qui le précèdent, il faut reculer dans le texte pour en déterminer l'état, parfois jusqu'au début du texte. À part pour ces exceptions, les étiquettes ne posent pas de difficultés particulières pour autant qu'on ne modifie pas le texte.

Contrôle des limites des caractères d'étiquette. Les caractères d'étiquette sont codés dans le plan 14 afin d'en faciliter le repérage et le contrôle. Les courts extraits de code C/C++ ci-dessous illustrent, pour les trois principales formes de stockage Unicode, la manière dont on peut efficacement vérifier l'appartenance à l'intervalle U+E0000..U+E007F. Ce type de contrôle permet aux mises en œuvre qui ne désirent pas prendre en charge ces caractères d'étiquette de les éliminer efficacement.

Contrôle des limites exprimé en UTF-32 :

```
if (((unsigned) *s)  - 0xE0000 <= 0x7F)
```

Contrôle des limites exprimé en UTF-16 :

```
if ((*s == 0xDB40) && (((unsigned) *(s+1)) – 0xDC00 <= 0x7F))
```

Contrôle des limites exprimé en UTF-8 :

```
if ((*s == 0xF3) && (*(s+1)) == 0xA0) && ((*(s+2) & 0xFE) == 0x80))
```

Le contrôle des limites UTF-32 et UTF-16 peut également s'écrire à l'aide de masques binaires. Les deux méthodes de contrôle devraient être d'efficacité égale.

Contrôle des limites exprimé en UTF-32 :

```
if (((*s) & 0xFFFFFF80) == 0xE0000)
```

Contrôle des limites exprimé en UTF-16 :

```
if ((*s == 0xDB40) && ((*(s+1) & 0xDC80) == 0xDC00))
```

Édition et modification. L'édition de texte et les étiquettes qui y sont imbriquées présentent des difficultés particulières car ses étiquettes sont associées à un état. Toute modification au texte est rendue plus complexe car ces modifications doivent prendre en compte la langue courante et veiller au bon maintien des paires étiquettes ouvrantes et fermantes. Si un programme d'édition ne se rend pas compte que certaines étiquettes ont un état et qu'il ne peut les traiter correctement, il pourrait alors fort bien modifier le texte et le corrompre. Il se pourrait alors qu'un utilisateur supprime une partie de l'étiquette ou qu'il colle — dans un contexte — erroné du texte contenant une étiquette.

Risques liés à une prise en charge partielle. Même les programmes qui n'interprètent pas les étiquettes sont tenus de conserver les paires d'étiquettes intactes. Les étiquettes non appariées doivent être éliminées lors d'une sauvegarde ou d'une transmission.

Toutefois, il se peut que du texte mal formé soit produit et transmis par un éditeur qui ne comprend pas les étiquettes. C'est pourquoi les mises en œuvre qui prennent en compte les

étiquettes linguistiques doivent être prêtes à recevoir des étiquettes mal formées. À la réception d'une étiquette mal formée ou non appariée, une mise en œuvre qui analyse ces étiquettes doit réinitialiser la langue à AUCUNE et ignorer l'étiquette.

Conformité Unicode

Les règles de conformité d'Unicode qui s'appliquent aux caractères-étiquettes sont identiques à celles qui s'appliquent à tous les autres caractères Unicode. Un processus conforme n'est pas tenu d'interpréter les caractères-étiquettes. S'il les interprète, il doit le faire conformément au standard, c'est-à-dire sous la forme d'étiquettes écrites en toutes lettres. Cependant, l'étiquetage linguistique d'un texte ne signifie pas qu'il faille l'interpréter d'une façon particulière. Si une application n'interprète pas les caractères étiquettes, elle doit les laisser intacts et faire ce qu'elle fait habituellement avec les caractères non interprétés.

La présence d'une étiquette bien formée ne garantit pas l'étiquetage correct d'un texte. C'est ainsi qu'une application pourra par erreur désigner un texte français à l'aide d'une étiquette espagnole. Les mises en œuvre d'Unicode qui utilisent déjà des mécanismes hors-texte pour identifier la langue d'un texte ou des mécanismes *dans le texte* lourds comme XML ou HTML doivent continuer de se comporter comme elles le font déjà et faire abstraction des caractères d'étiquette ; elles peuvent en interdire l'utilisation afin d'éviter des conflits avec leur propre balisage correspondant.

Description syntaxique des étiquettes

Cette section décrit, à l'aide d'une grammaire de forme Backus-Naur étendue, la syntaxe des étiquettes définies ci-dessus. Cette description formelle comprend les extensions suivantes :

1. On définit les contraintes sémantiques à l'aide de règles prenant la forme d'une assertion exprimée entre des paires d'accolades doubles ; la variable $$ représente la chaîne constituée par tous les symboles terminaux qui correspondent au non-terminal courant.

   Exemple : `{{ Assertion ($$[0] == '?'); }}`

   Signification : Le premier caractère de la chaîne qui correspond à ce non-terminal doit être un « ? ».

2. Les règles de contrainte emploient un certain nombre de fonctions prédicats non définies par ailleurs ; leur nom suffit à définir leur nature prédicative.

   Exemple : `EstIndicatifDeLangueRFC3066( argument-étiquette )`

   Signification : *argument-étiquette* est un indicatif de langue valable selon le RFC 3066.

3. La fonction d'expansion lexicale, ÉTIQUETTE, sert à représenter la forme étiquette d'un caractère ASCII ; cette fonction prend comme argument un caractère ou un ensemble de caractère exprimés sous la forme d'un intervalle ou d'une énumération.

   Exemple : `ÉTIQUETTE ('-')`

   Signification : U+E002D ÉTIQUETTE TRAIT D'UNION-SIGNE MOINS

   Exemple : `ÉTIQUETTE ([A-Z])`

   Signification : U+E0041 ÉTIQUETTE LETTRE MAJUSCULE LATINE A..U+E005A ÉTIQUETTE LETTRE MAJUSCULE LATINE Z

4. On utilise une macro afin de représenter les symboles terminaux qui ne peuvent être représentés directement en ASCII. Cette macro prend comme argument le nom ISO 10646 du caractère.

   Exemple : `'${ÉTIQUETTE DE LANGUE}'`

   Signification : le littéral caractère dont le numéro est égal à `U+E0001.`

5. Cette grammaire utilise les indicateurs d'occurrence suivants : « + » (une fois ou plus) et « * » (zéro fois ou plus). On indique la présence optionnelle d'une expression en l'entourant de crochets « [» et «] ».

Syntaxe formelle des étiquettes

```
étiquette                       :   étiquette-linguistique
                                |   annuler-toutes-étiquettes
                                ;

étiquette-linguistique          :   intro-étiquette-linguistique
                                    argument-étiquette-linguistique
                                ;

argument-étiquette-linguistique :   argument-étiquette
              {{Assertion (EstIndicatifDeLangueRFC3066( $$ )); }}
                                |   annuler-étiquette
                                ;

annuler-toutes-étiquettes       :   annuler-étiquette
                                ;

argument-étiquette              :   caractère-étiquette+
                                ;

caractère-étiquette             :   { c : c dans
    ÉTIQUETTE({a : a parmi caractères ASCII imprimables ou ESPACE })}
                                ;

intro-étiquette-linguistique    :   '${ÉTIQUETTE DE LANGUE}'
                                ;

annuler-étiquette               :   '${ANNULER ÉTIQUETTE}'
                                ;
```