(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 933 793**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **08 54788**

(51) Int Cl[8] : ***G 06 F 17/22*** *(2006.01)*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 11.07.08.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 15.01.10 Bulletin 10/02.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *CANON KABUSHIKI KAISHA* — JP.

(72) **Inventeur(s) :** FABLET YOUENN et DENOUAL FRANCK.

(73) **Titulaire(s) :** CANON KABUSHIKI KAISHA.

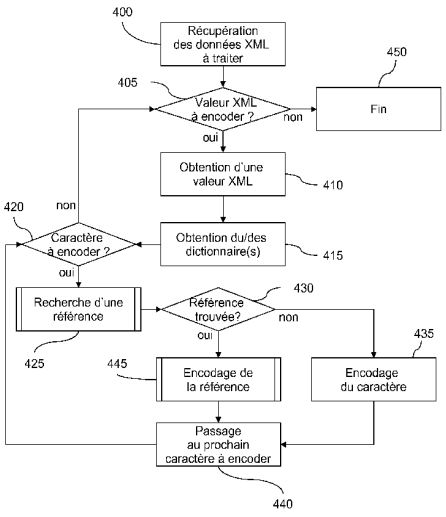
(74) **Mandataire(s) :** SANTARELLI.

(54) PROCEDES DE CODAGE ET DE DECODAGE, PAR REFERENCEMENT, DE VALEURS DANS UN DOCUMENT STRUCTURE, ET SYSTEMES ASSOCIES.

(57) La présente invention concerne un procédé de codage d'un document structuré type XML, un procédé de décodage correspondant ainsi que des systèmes associés.

Le procédé de codage d'un document structuré (502) comprenant des valeurs à coder, comprend plus particulièrement une étape de codage (445), à partir d'au moins un dictionnaire d'indexation (508) comprenant des entrées, d'au moins une valeur à coder, procédé dans lequel on procède:

- à l'identification (425, 630) d'au moins une valeur d'entrée du dictionnaire liée à la valeur à coder par correspondance entre au moins une partie de l'une des valeurs et une sous-partie de l'autre valeur; et
- au codage (740, 750, 755, 760, 765) de ladite valeur à coder par référence, selon ladite correspondance, à au moins l'entrée identifiée du dictionnaire.



FR 2 933 793 - A1

La présente invention concerne un procédé de codage d'un document structuré, un procédé de décodage correspondant ainsi que des systèmes associés.

Les documents structurés sont des documents qui contiennent de l'information à propos de leur structure logique, sémantique et intellectuelle. Des exemples classiques sont le format HMTL (pour "*Hypertext Markup Language*", langage de balisage) ou le format XML dont on fera référence pour la suite de la description.

Le langage XML (acronyme de "*Extensible Markup Language*" pour langage de balisage extensible) est une syntaxe pour définir des langages informatiques adaptés à des utilisations différentes mais pouvant être traités par les mêmes outils.

De façon connue en soi, un document XML se compose d'éléments définis par une balise ouvrante comportant le nom de l'élément (par exemple: <balise>) et par une balise fermante comportant, elle aussi, le nom de l'élément (par exemple: </balise>). Ces éléments peuvent contenir d'autres éléments de façon hiérarchique ou des données textuelles.

D'autres types de données XML sont connus en soi. Les données XML sont décrites généralement par des items, chaque item pouvant être un début d'élément, une fin d'élément, un « attribut », une « donnée textuelle », un « commentaire », une « instruction de traitement » et une « section d'échappement ».

Afin de pouvoir mélanger plusieurs langages XML différents ayant parfois des éléments portant le même nom, un ajout a été effectué à la syntaxe XML permettant de définir des espaces de nommage ("*Namespace*" selon la terminologie anglo-saxonne) en associant un préfixe à une URI (acronyme de "*Uniform Resource Identifier*", pour Identifiant uniforme de ressource). En utilisation, l'espace de nommage d'un élément ou d'un attribut est précisé en

faisant précéder son nom par le préfixe associé à l'espace de nommage suivi de « : » (par exemple: « <ml:balise ml:attribut="valeur"> » indique que l'élément *balise* découle de l'espace de nommage *ml* et qu'il en est de même pour l'attribut *attribut*).

Le format XML présente de nombreux avantages et est devenu un langage de référence pour stocker des données dans un fichier ou pour échanger des données. XML permet en particulier de disposer de nombreux outils pour traiter les fichiers générés. D'autre part, un document XML peut être édité manuellement avec un simple éditeur de texte. En outre, comme un document XML contient sa structure intégrée aux données, ce document est très lisible même sans en connaître la spécification.

Néanmoins, le principal inconvénient des langages structurés, ici la syntaxe XML, est d'être très prolixe. Ainsi la taille d'un document XML peut être plusieurs fois supérieure à la taille intrinsèque des données. Cette taille importante des documents XML induit aussi un temps de traitement important lors de la génération et de la lecture de documents XML.

Pour remédier à ces inconvénients, des méthodes pour encoder un document XML ont été recherchées. Le but de ces méthodes est de coder le contenu du document XML sous une forme plus efficace, tout en permettant de reconstruire facilement le document XML. Cependant, la plupart de ces méthodes ne conservent pas l'ensemble des avantages du format XML.

Parmi ces méthodes, la plus simple consiste à coder les données de structure dans un format binaire au lieu d'utiliser un format textuel. En outre, la redondance des informations structurelles dans le format XML peut être supprimée ou au moins diminuée. Par exemple, il n'est pas forcément utile de préciser le nom de l'élément dans la balise ouvrante et la balise fermante.

Une autre méthode est d'utiliser une ou plusieurs tables d'index, en particulier pour les chaînes de caractères, par exemple les noms d'éléments et d'attributs, qui sont généralement répétés dans un document XML. Ainsi, lors de la première occurrence d'un nom d'élément, celui-ci est codé normalement dans le fichier et un index lui est associé. Puis, pour les occurrences suivantes de ce nom d'élément, l'index est utilisé à la place de la chaîne complète,

réduisant la taille du document généré, mais facilitant aussi la lecture. En effet, il n'y a plus besoin de lire la chaîne complète dans le fichier, et en outre, la détermination de l'élément lu peut être réalisée par une comparaison d'entiers au lieu d'une comparaison de chaînes de caractères.

Enfin, au-delà de ces méthodes élémentaires, il existe des méthodes plus évoluées consistant notamment à prendre en compte un nombre plus important d'informations structurelles du document afin de compresser davantage les données.

On peut citer, entre autres, le cas du format « Efficient XML » qui prend en compte l'ordre d'apparition des différents items au sein d'un document pour construire, de façon évolutive, une grammaire de productions qui permet d'encoder les items les plus fréquents sur un faible nombre de bits. Chaque production comprend une description d'événement (ou item) XML, une valeur de codage associée et l'indication de la grammaire suivante à utiliser. Le codage d'un événement XML est alors réalisé en utilisant la valeur de codage correspondant à la production contenant la description la plus précise de l'événement XML.

Les informations contenues dans l'événement et non décrites dans la production sont codées. Ces informations généralement sous la forme de chaîne de caractères sont codées à l'aide de tables ou dictionnaires d'index comme expliqué précédemment: la première occurrence d'une chaîne de caractères est codée avec sa longueur puis insérée dans les dictionnaires afin de représenter les occurrences futures par l'index de la chaîne dans les dictionnaires plutôt que par sa valeur.

La grammaire est évolutive en ce qu'après l'occurrence d'un événement XML déjà décrit par une production de la grammaire, la grammaire est modifiée pour inclure une nouvelle production plus efficace correspondant à cet événement XML. Ainsi, cette production peut soit contenir une description plus précise de l'événement, diminuant le nombre d'informations à coder pour représenter l'événement, soit avoir une valeur de codage plus compacte.

Les règles de grammaires ou productions utilisées peuvent soit être des règles génériques, communes à tous les documents XML et construites à

partir de la syntaxe XML, soit être des règles spécifiques à un type de document, construites à partir d'un Schéma XML décrivant la structure de ce type de document (en décrivant l'ensemble des éléments et des attributs pouvant être présents dans le document XML, ainsi que les relations entre ces éléments et ces attributs).

Lors du décodage, le processus inverse est utilisé : la valeur de codage est extraite et permet d'identifier l'événement XML codé, ainsi que les informations complémentaires à décoder.

En outre, lors du décodage, les mêmes règles d'évolution des grammaires sont utilisées, permettant d'avoir à tout moment un ensemble de règles de grammaires identique à celui qui était utilisé lors du codage au même endroit du document XML.

Une alternative au format *Efficient XML* est connue sous le nom de format *Fast Infoset* permettant une représentation plus compacte d'un document XML en utilisant des codes d'événements binaires pour les données structurelles et des tables d'indexation pour les valeurs sous forme de chaînes de caractères (valeurs de contenu ou de structures, typiquement les données textuelles et les attributs).

Dans ce format, les types d'événements sont décrits en tant que liste qui utilise des codes binaires de longueur variable. Ce format utilise les techniques d'indexation en créant des tables pour des ensembles de valeurs XML précises. Ces tables permettent d'encoder une valeur donnée de manière littérale, par exemple selon les formats UTF8 ou UTF16 (où UTFx est l'acronyme de "*UCS transformation format*" x bits), la première fois que cette information est rencontrée durant l'encodage du document. Cette information est ensuite ajoutée dans la table d'indexation et associée à un index.

Ultérieurement, lorsque cette information est détectée à nouveau dans le document XML, l'index correspondant est récupéré dans la table d'indexation et on encode alors la valeur de cet index à la place de l'information. On peut ainsi obtenir une compression notable des données.

On note un certain nombre de tables d'indexation, parmi lesquelles :

- deux tables indexant respectivement les préfixes et les URI afin de définir les espaces de nommage ;
- deux tables spécifiques indexant respectivement les valeurs d'attributs et les valeurs de nœuds textes ;
- une table indexant les noms locaux d'attributs et d'éléments ;
- deux tables spécifiques indexant respectivement, d'une part les noms qualifiés (qui regroupent par exemple un préfixe, une URI et un nom local) d'éléments, et d'autre part les noms qualifiés d'attributs.

On peut noter que la norme *Fast Infoset* permet à l'encodeur de décider si telle ou telle valeur d'attribut ou de nœud texte est à indexer, par exemple en fonction de la longueur de la valeur ou de la chaîne de caractère. Ceci permet notamment de limiter la taille de la mémoire utilisée par l'encodeur. La décision d'indexer ou non une valeur d'attribut ou un nœud texte est alors encodée dans le flux *Fast Infoset* pour permettre au décodeur associé d'indexer ou non les valeurs à décoder.

Ainsi, pour ces différentes solutions de codage, il est généralement procédé à un traitement particulier des données de structure (par exemple utilisation de productions) et à un traitement séparé des valeurs d'attributs, d'éléments, de données textuelles, de préfixes, etc. à l'aide de dictionnaires d'index tels que les tables précitées.

Or, toutes ces techniques de codage à l'aide de dictionnaires se basent, pour le codage d'une valeur donnée, sur la recherche à l'intérieur de ces dictionnaires d'une valeur identique: soit la valeur est présente en entier et on utilise l'index correspondant pour coder, soit la valeur est absente des dictionnaires et on code alors sa longueur et sa valeur. Ces techniques exploitent donc les répétitions potentielles des valeurs sous forme de chaînes de caractères, par exemple une valeur de structure représentant le nom d'attribut ou d'élément qui se répète, une valeur de contenu correspondant à la valeur d'attribut et à une donnée textuelle de l'arbre XML qui se répète également.

L'invention vise à améliorer ces techniques de codage à l'aide de dictionnaires en se basant sur la corrélation qui peut exister entre différentes

chaînes de caractères et notamment celles à l'intérieur d'un même document XML. En effet, ce dernier est généralement formé de mots ou chaînes de caractères issues d'une même langue, par exemple le français. Cette corrélation peut notamment s'illustrer par des sous-parties de chaînes de caractères identiques d'un mot à l'autre.

A cet effet, l'invention vise notamment un procédé de codage d'un document structuré comprenant des valeurs à coder, le procédé comprenant une étape de codage, à partir d'au moins un dictionnaire de codage, également appelé ci-après d'indexation, comprenant des entrées, d'au moins une valeur à coder, caractérisé en ce qu'il comprend les étapes suivantes :

- l'identification d'au moins une valeur d'entrée du dictionnaire liée à la valeur à coder par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur ;
- le codage de ladite valeur à coder par référence, selon ladite correspondance, à au moins l'entrée identifiée du dictionnaire.

On entend ici par sous-partie d'une valeur une partie non entière de cette valeur. Ainsi, selon l'invention, deux situations se présentent: d'une part, lorsqu'une sous-partie uniquement de la valeur à coder correspond à tout ou partie d'une valeur d'entrée du dictionnaire, et d'autre part, lorsque la valeur à coder dans son intégralité correspond à une sous-partie uniquement d'une valeur d'entrée du dictionnaire.

Grâce au procédé selon l'invention, on détermine dans un premier temps des similarités entre la valeur à coder et les valeurs déjà codées dans le dictionnaire de telle sorte que l'on utilise, pour le codage de la première, les informations de codage de la deuxième alors même que les deux valeurs sont différentes.

On obtient ainsi une diminution de la taille des dictionnaires utilisés car soit la valeur à coder dans son intégralité correspond à une ou plusieurs sous-partie de valeurs d'entrée du dictionnaire et on peut éviter d'insérer une nouvelle entrée dans le dictionnaire, soit l'entrée insérée pour cette nouvelle valeur à coder présente un nombre de bits d'information réduit parce que la

référence à une autre entrée du dictionnaire est moins coûteuse que la présence de la valeur entière.

Ce codage ne modifie pas l'organisation des données du document structuré et reste donc compatible avec un traitement de codage ou décodage à la volée comme on l'illustrera par la suite.

On comprend aisément qu'en disposant, pour une sous-partie seulement de la valeur à coder, d'une référence à une valeur déjà codée, on procède également au codage du reste de la valeur à coder. On peut utiliser soit un codage classique, soit appliquer l'invention pour plusieurs sous-parties comme évoqué ci-après.

En particulier, lesdites valeurs comprennent des chaînes de caractères et ladite identification comprend la recherche de la chaîne ou d'une sous-chaîne de caractères de la valeur à coder dans une pluralité de valeurs d'entrée du dictionnaire. Ainsi, pour le codage, la référence comprend l'identification de la valeur d'entrée identifiée ainsi que la localisation de la chaîne ou sous-chaîne de caractères reconnue dans cette valeur d'entrée.

En pratique, on veillera à initier la recherche à l'aide d'une sous-chaîne de taille préfixée, trois caractères par exemple. Cependant, la similitude entre les chaînes de caractères peut être supérieure à trois caractères. Ainsi, on prévoit que, lorsque ladite identification comprend la recherche d'une sous-chaîne et en cas de recherche positive (c'est-à-dire lorsqu'une entrée dont la valeur comprend ladite sous-chaîne a été trouvée), on ajoute au moins un caractère à ladite sous-chaîne de recherche de sorte à rechercher la sous-chaîne ainsi augmentée dans ladite pluralité de valeurs d'entrée du dictionnaire. Ainsi, de façon progressive, on identifie l'entrée présentant la plus grande similarité avec la valeur à coder. Pour ce faire, on itère, jusqu'à ce qu'un critère soit atteint et tant que la recherche est positive :

- l'ajout d'au moins un caractère à ladite sous-chaîne de recherche ;
- ladite recherche de la sous-chaîne ainsi augmentée dans ladite pluralité de valeurs d'entrée du dictionnaire.

Un tel critère peut, par exemple, porter sur la taille de la sous-chaîne (jugée suffisamment grande) ou sur le temps passé de recherche.

Au final, on peut être amené à chercher l'ensemble de la chaîne de la valeur à coder dans les entrées du dictionnaire.

Des chaînes de caractères à coder peuvent être suffisamment grandes pour qu'une même sous-chaîne se répète plusieurs fois dans la chaîne entière à coder. Il peut donc s'avérer efficace d'effectuer un codage par référence "interne" en s'appuyant sur une auto-corrélation. Dans ce dessein, on prévoit que ledit dictionnaire comprend une entrée correspondant à une sous-chaîne déjà codée de ladite valeur à coder de telle sorte qu'une sous-chaîne ultérieure de ladite valeur à coder peut être codée en référence à ladite sous-chaîne déjà codée. On comprend ici que le dictionnaire doit être compris au sens large comme l'ensemble des valeurs déjà codées associées à des informations, codes ou index de codage, et ce même si cet ensemble est réparti dans différentes mémoires ou tables.

Dans un mode de réalisation, lorsque plusieurs valeurs d'entrée trouvées comprennent ladite sous-chaîne de recherche, on sélectionne une desdites entrées trouvées selon un critère prédéterminé, par exemple selon un critère de compression, un critère de rapidité de décodage ou un critère de proximité, de sorte que ledit codage est réalisé en référence à ladite entrée sélectionnée.

Au vu du fonctionnement précédemment décrit, on comprend que, lorsqu'aucune valeur d'entrée n'est identifiée pour la sous-chaîne de recherche, on peut insérer, dans ledit dictionnaire, une nouvelle entrée associant ladite sous-chaîne de recherche à des données de codage de ladite sous-chaîne de recherche non trouvée. On conserve ainsi des tables de dictionnaire évolutives.

Dans un mode de réalisation, ledit au moins un dictionnaire comprend une pluralité d'entrées correspondant à des valeurs prédéfinies avant codage. Ainsi, il est possible de privilégier, pour le codage, certaines valeurs ou chaînes de caractères indépendamment du document XML à traiter.

En variante ou en combinaison, le procédé peut comprendre au moins une étape préalable de codage d'une valeur antérieure précédent ladite valeur à coder dans le document et une étape préalable d'ajout, dans ledit au moins un dictionnaire, d'une entrée correspondant à la valeur antérieure,

procédé dans lequel la référence lors dudit codage par référence porte alors sur ladite entrée ajoutée. Cette formation progressive du dictionnaire au fur et à mesure du codage correspond à l'élaboration classique des dictionnaires de codage par l'insertion, la plupart du temps, de couples (index, valeur) pour chaque valeur codée.

Selon cette réalisation de l'invention, on réutilise le dictionnaire formé classiquement pour en retirer des références de codage.

Dans un mode de réalisation, ledit codage par référence comprend le codage d'une information d'identification du dictionnaire d'indexation et de ladite entrée identifiée dans ce dictionnaire, ainsi qu'une information de localisation de la sous-chaîne qui peut comprendre une information de positionnement, dite "offset", dans ladite valeur d'entrée identifiée et une information de longueur, correspondant généralement à la taille de la sous-partie ou sous-chaîne de caractères utilisée lors de l'étape d'identification.

Ainsi, on prévoit également que le procédé comprend une étape, préalable à ladite identification, de sélection d'au moins un dictionnaire parmi une pluralité de dictionnaires, ladite sélection étant fonction du type de la valeur à coder. Le type s'entend ici dans un sens large : il comprend notamment le type d'évènement (début de balise, nœud texte) et le contexte de la valeur (valeur d'un attribut 'a', nom de balise dont le parent est 'b'…). On sélectionne de la sorte le dictionnaire d'indexation des attributs uniquement lorsque l'on encode un attribut (de même pour les autres types de valeurs à coder).

En variante, ledit codage par référence comprend le codage d'une donnée de codage correspondant à une entrée d'au moins un dictionnaire de références, une entrée dudit dictionnaire de références renseignant une information d'identification d'une entrée d'un dictionnaire de codage, une information de positionnement et une information de longueur. En utilisant de tels dictionnaires de références, on obtient une meilleure compression et efficacité de codage, car toutes les informations des références sont substituées par exemple par l'index uniquement de l'entrée correspondante dans le dictionnaire de références. Ce dictionnaire de références peut être créé

sur la même base que les dictionnaires de codage, avec notamment l'insertion d'une référence comme entrée à la première occurrence d'apparition.

Quelque soit le mode choisi, on peut prévoir que ladite information d'identification comprend un identifiant d'un dictionnaire parmi une pluralité de dictionnaires de codage (par exemple local, interne ou global) et un identifiant de ladite entrée identifiée à l'intérieur dudit dictionnaire. En pratique, on privilégie les dictionnaires locaux (un dictionnaire local correspond par exemple à l'ensemble des valeurs d'un attribut 'a'. A l'opposé, un dictionnaire global correspond à l'ensemble des valeurs d'attributs et/ou d'éléments déjà encodées). En effet, les références de types locales sont fréquentes car les différentes valeurs d'éléments XML partageant le même nom sont généralement similaires. Du fait que les dictionnaires locaux ont moins d'entrées qu'un dictionnaire global, on peut encoder l'index de l'entrée identifiée de manière plus compacte.

En particulier, il est prévu, dans ce cas, que l'on associe, à chaque dictionnaire de codage, une plage d'indices pour l'identification de leurs entrées, et l'on encode alors l'identifiant de ladite entrée identifiée relativement à ladite plage, en particulier en encodant la différence entre cet identifiant et le premier indice. Ainsi, on minimise le nombre de bits utilisés pour l'encodage de cette information d'identification.

Il a été observé que dans de nombreux cas de valeurs XML, on retrouve des sous-parties correspondant soit au début, soit à la fin d'une valeur du dictionnaire. Ces deux cas particuliers correspondent donc a des encodages particuliers de la valeur d'offset. A cet effet, on prévoit que l'information de positionnement comprend un premier code lorsque ladite partie ou sous-partie de la valeur à coder (i.e. la sous-chaîne de recherche) correspond au début de la valeur d'entrée identifiée, d'un deuxième code lorsque ladite partie ou sous-partie de la valeur à coder correspond à la fin de la valeur d'entrée identifiée, et une valeur d'offset de la position de ladite partie ou sous-partie de la valeur à coder dans la valeur d'entrée identifiée, dans les autres cas. Généralement cette valeur d'offset est établie relativement au début de la valeur d'entrée. On peut ainsi utiliser des codes compacts spécifiques de début et de fin au lieu de

coder un offset sur un nombre de bits important (qui est généralement fonction du nombre de caractères de la valeur d'entrée).

On prévoit aussi de déterminer des positions préférentielles dans la valeur d'entrée identifiée, notamment en fonction d'au moins un caractère prédéterminé type ponctuation par exemple, et ladite information de positionnement est établie, au moins dans lesdits autres cas, relativement à au moins une desdites positions préférentielles déterminées. Encore une fois, ces positions préférentielles permettent d'utiliser des codes plus compacts pour identifier le positionnement.

Selon une caractéristique de l'invention, l'information de longueur comprend un code parmi un ensemble fini de codes de longueur correspondant respectivement à un ensemble fini de longueurs, et un troisième code combiné à une valeur pour les longueurs non comprises dans ledit ensemble. Cette réalisation permet d'optimiser de façon simple le codage en associant aux longueurs faibles (les références aux sous-chaînes courtes sont plus fréquentes que les références aux sous-chaînes longues) des codes courts.

En particulier, lesdites informations de positionnement et de longueur sont encodées conjointement par un encodage adaptatif desdits premier code, deuxième code, troisième code et codes de longueur. On obtient ainsi un codage plus compact.

Dans un mode de réalisation, le procédé comprend en outre une étape d'ajout, aux données codées correspondant à la valeur codée, d'une indication renseignant la fin de la valeur codée, en particulier au travers soit de l'encodage du nombre d'octets utilisés pour encoder ladite valeur permettant de la sorte de sauter aisément le décodage de cette valeur si elle n'est pas utile à l'application ou au décodage, soit de l'encodage du nombre de caractères de la valeur, soit enfin de l'encodage d'un symbole de fin de valeur encodée. Cette indication permet d'effectuer, comme expliqué ci-après, un décodage efficace du flux codé résultant du procédé de codage objet de l'invention.

Selon une caractéristique de l'invention, le codage des références à une entrée du dictionnaire de codage met en œuvre au moins une table d'encodage fixe, par exemple de Huffman. Il s'agit donc d'un codage statique où

les valeurs de codage utilisées, une fois définies, restent inchangées. Cela permet notamment de les définir de façon anticipée. Selon cette réalisation de l'invention, on obtient un décodage rapide quelle que soit la position des données à décoder dans le document.

Selon une autre caractéristique de l'invention, le codage des caractères et références s'effectue par suite d'octets avec un codage statique. Un caractère est représenté sous la forme d'un ou plusieurs octets, les caractères les plus fréquents (les caractères ASCII notamment) étant codés sur un seul octet. De même, les références sont codées sur des suites d'octets, les types de références fréquents étant codés sur un petit nombre d'octets (deux octets par exemple) et les types de références moins fréquents étant codés sur un nombre plus important d'octets (trois octets et plus). L'intérêt de cette mise en œuvre est notamment une intégration aisée avec Fast Infoset (format orienté octet), des gains en terme de compression moyennant un surcoût de traitement minime car le décodage est effectué sur des séries de bits et non bit par bit.

En variante, le codage desdites références met en œuvre au moins une table d'encodage adaptative, le procédé comprenant alors une étape de mise à jour de ladite au moins une table d'encodage suite au codage de ladite valeur à coder. On accroît alors la compacité et l'efficacité du codage du document structuré.

On note ici que les tables d'encodage peuvent être les dictionnaires de codage tels qu'évoqués précédemment, si ces derniers contiennent explicitement les valeurs de codage utilisées dans le flux binaire final et non seulement des symboles associés aux entrées, symboles qu'il faut ensuite coder.

En combinaison du codage de valeurs du document structuré, on procède au codage des informations de structure de ce même document.

Ainsi, on prévoit, dans une mode de réalisation, que ledit document structuré comprend des données de structure à coder, et le procédé comprenant alors les étapes suivantes de codage d'une donnée de structure:

- la prédiction d'au moins une partie des informations composant ladite donnée de structure, à partir d'une grammaire de codage ;

- si la prédiction est positive, l'encodage d'une information dite « de conformité » représentative de ce résultat positif, et l'encodage des autres informations de ladite donnée de structure non comprise dans ladite partie.

Ainsi, le nombre de bits codés utilisés peut être réduit par rapport aux méthodes classiques de codage des priorités de productions par exemple.

En particulier, on effectue une pluralité de prédictions pour des données de structure successives, ledit procédé comprend alors :

- une étape de détermination s'il existe une succession de prédictions correctes pour des données de structure, et
- si le résultat de ladite étape de détermination est positif, une étape de codage du nombre de prédictions correctes successives.

On améliore ainsi la compacité du codage.

En particulier, lorsqu'une prédiction pour une donnée de structure est positive, on enregistre les données encodées correspondant auxdites autres informations dans une mémoire tampon, et lorsqu'une prédiction est négative, on réalise ladite étape de codage du nombre de prédictions correctes successives et on copie lesdites données encodées stockées dans la mémoire tampon.

On a ainsi encodé l'ensemble des données nécessaires à la reconstruction, du côté du décodeur, de la structure du fichier XML par exemple.

Corrélativement, l'invention vise également un système de codage d'un document structuré comprenant des valeurs à coder, le système comprenant des moyens de codage, à partir d'au moins un dictionnaire de codage comprenant des entrées, d'au moins une valeur à coder, caractérisé en ce qu'il comprend un moyen d'identification d'au moins une valeur d'entrée du dictionnaire liée à la valeur à coder par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur,

et les moyens de codage sont aptes à coder ladite valeur à coder par référence, selon ladite correspondance, à au moins ladite entrée identifiée du dictionnaire.

De façon optionnelle, le système de codage peut comprendre des moyens se rapportant aux caractéristiques du procédé de codage exposé précédemment.

L'invention a également trait à un procédé de décodage d'un flux codé binaire d'un document structuré comprenant des valeurs décodées, le procédé comprenant une étape de décodage, à partir d'au moins un dictionnaire de décodage comprenant des entrées, de données codées correspondant à une valeur décodée, caractérisé en ce que ledit décodage comprend les étapes suivantes :

- l'identification, dans lesdites données codées, et le décodage d'au moins une information de référence liant la valeur d'une entrée du dictionnaire de décodage à la valeur décodée par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur ;
- la récupération de ladite partie ou sous-partie de la valeur de l'entrée à partir de ladite référence décodée de sorte à former, au moins en partie, la valeur décodée.

De façon similaire au codage, soit l'ensemble des données codées d'une valeur forment une référence à une sous-partie d'une entrée du dictionnaire; soit plusieurs parties des données codées permettent de former plusieurs sous-parties de la valeur décodée.

Dans ce dernier cas, il est entendu que, pour décoder l'ensemble des données codées correspondant à la valeur décodée, il convient de procéder, soit de la même façon, soit d'une façon classique, au décodage des autres parties desdites données codées.

Ainsi, le décodage d'une nouvelle valeur s'appuie sur les valeurs déjà décodées ou présentes dans le dictionnaire sans pour autant être égale à celle-ci. En utilisant des références basées sur des sous-chaînes de caractères en commun, on réduit ainsi la taille des dictionnaires ou tables de décodage, et donc les ressources utilisées.

De façon optionnelle, le procédé de décodage peut comprendre des caractéristiques images de celles se rapportant aux caractéristiques du procédé de codage exposé précédemment.

En outre, dans un mode de réalisation, ledit décodage des données codées correspondant à la valeur décodée comprend une succession de décodages partiels de parties desdites données codées pour former des sous-parties décodées de la valeur décodée et la concaténation desdites sous-parties de sorte à former ladite valeur décodée,

chaque décodage partiel comprenant l'un parmi :

- lesdits identification et décodage d'une information de référence suivis de ladite récupération ; et
- un décodage direct d'au moins un caractère.

Dans un mode de réalisation, on identifie et on stocke dans ledit dictionnaire, préalablement audit décodage des données codées correspondant à la valeur décodée, une pluralité de données codées correspondant à une pluralité d'autres valeurs codées dudit document, généralement celles précédent la valeur à décoder dans le sens du document,

et ladite récupération comprend le décodage des données codées associées à l'entrée référencée.

Ainsi, lorsque l'on souhaite accéder à une partie seulement du document structuré, on ne décode les données codées du début du document que lorsque celles-ci sont utiles pour la partie à accéder, c'est-à-dire référencées par cette partie. Le décodage du document structuré est ainsi partiel, plus rapide et requiert moins de ressources.

Une alternative consiste à décoder chacune desdites données codées au fur et à mesure de la lecture du flux codé binaire et à enregistrer les valeurs décodées dans ledit dictionnaire.

Ainsi, l'invention a également trait à un procédé d'accès à une partie d'un document structuré comprenant le décodage dudit document comme évoqué ci-dessus, dans lequel on identifie et stocke ladite pluralité de données codées pour les valeurs codées situées, dans ledit document, avant ladite partie à accéder,

et on décode ladite partie à accéder, ledit décodage de la partie à accéder comprenant, pour au moins une valeur codée de ladite partie à accéder, lesdits

identification et décodage d'une information de référence suivis de ladite récupération.

De retour au procédé de décodage, selon un autre mode de réalisation, le procédé comprend une étape de détermination, dans ledit flux codé binaire, desdites données codées correspondant à une valeur décodée à l'aide d'au moins une indication renseignant la fin de ladite valeur codée.

Corrélativement, l'invention vise également un système de décodage d'un flux codé binaire d'un document structuré comprenant des valeurs décodées, le dispositif comprenant des moyens de décodage, à partir d'au moins un dictionnaire de décodage comprenant des entrées, de données codées correspondant à une valeur décodée, caractérisé en ce qu'il comprend :

- un moyen d'identification, dans lesdites données codées, et de décodage d'au moins une information de référence liant la valeur d'une entrée du dictionnaire de décodage à la valeur décodée par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur ;

- un moyen de récupération de ladite partie ou sous-partie de la valeur de l'entrée à partir de ladite référence décodée de sorte à former, au moins en partie, la valeur décodée.

De façon optionnelle, le système de décodage peut comprendre des moyens se rapportant aux caractéristiques du procédé de décodage exposé précédemment.

Un moyen de stockage d'informations, éventuellement totalement ou partiellement amovible, lisible par un système informatique, comprend des instructions pour un programme informatique adapté à mettre en œuvre le procédé de codage, de décodage ou d'accès conforme à l'invention lorsque ce programme est chargé et exécuté par le système informatique.

Un programme d'ordinateur lisible par un microprocesseur, comprend des portions de code logiciel adaptées à mettre en œuvre le procédé de codage, de décodage ou d'accès conforme à l'invention, lorsqu'il est chargé et exécuté par le microprocesseur.

Les moyens de stockage d'information et programme d'ordinateur présentent des caractéristiques et avantages analogues aux procédés qu'ils mettent en œuvre.

La présente invention apporte ainsi des gains en termes de compression des informations de contenu de document XML, qui constituent souvent la majeure partie des données du document.

L'invention offre également des gains en termes de performance de vitesse et de mémoire, notamment en permettant l'encodage et le décodage à la volée avec un gain de compression notable sans augmenter l'utilisation mémoire, car les dictionnaires sont construits par le format XML binaire sous jacent et ne sont donc pas spécifiquement construits pour l'encodage par référencement.

L'invention permet également le décodage partiel du document plus rapide que dans l'art antérieur et en utilisant moins de mémoire. En effet, une partie des données peut être conservée sous forme compressée.

Enfin, un autre avantage de l'invention réside dans la compatibilité de l'encodage/décodage proposé avec les formats XML binaire sous jacents, tels *Fast Infoset* et EXI, de telle sorte que le surcoût pour la mise en œuvre de l'invention se limite à un nouveau codec (codeur-décodeur) uniquement.

D'autres particularités et avantages de l'invention apparaîtront encore dans la description ci-après, illustrée par les dessins ci-joints, dans lesquels :

- la **figure 1** représente un exemple de document XML à encoder ;
- la **figure 2** illustre le codage des valeurs du document de la **figure 1** selon les techniques classiques de l'état de la technique ;
- la **figure 3** illustre le codage des valeurs du document de la **figure 1** selon le procédé de la présente invention ;
- la **figure 4** illustre l'algorithme général d'encodage selon la présente invention ;
- la **figure 5** est une représentation schématique d'un encodeur pour la mise en œuvre de l'algorithme de la **figure 4** ;

- la **figure 6** représente, sous forme d'un logigramme, des étapes de recherche de références mise en œuvre dans le processus de la **figure 4** ;

- la **figure 7** représente, sous forme d'un logigramme, des étapes d'encodage d'une référence obtenue à la suite des étapes de la **figure 6** ;

- la **figure 8** est une représentation schématique d'un décodeur selon l'invention ;

- la **figure 9** illustre le décodage d'un flux binaire codé généré à l'issu du codage selon la **figure 3**, décodage selon le procédé de la présente invention ;

- la **figure 10** représente, sous forme d'un logigramme, des étapes pour le décodage partiel d'un document XML codé selon l'invention ; et

- la **figure 11** montre une configuration matérielle particulière d'un dispositif de traitement d'information apte à une mise en œuvre des procédés selon l'invention.

On se propose d'illustrer tout d'abord l'apport de l'invention pour l'encodage du document XML proposé en **figure 1**. Ce document XML comprend trois éléments "*person*" ayant chacun trois éléments enfants "*firstname*", "*lastname*" et "*city*" dotés d'un champ textuel.

Bien que l'on illustre l'invention ici par l'encodage de champs textuels, il est entendu que l'invention s'applique à l'encodage de toute valeur qui est classiquement effectuée via des tables d'index: préfixes, URI, valeurs d'attributs et de nœuds, noms locaux d'attributs et de nœuds, noms qualifiés ou non-qualifiés, …

Sur la **figure 2**, on a représenté le codage des chaînes de caractères par référence à un index, selon l'art antérieur, et sur la **figure 3**, un codage différentiel des chaînes de caractères à l'aide de la présente invention.

De façon classique, la deuxième occurrence de *« Liverpool »* utilise l'identifiant attribué à la première valeur de *« Liverpool »*.

Par les techniques connues, Il apparaît également que les redondances entre *« Mary »* et *« Mary Lisa »* ne sont pas exploitées, de même que les redondances entre « *London* » et « *Londasle* ».

Au contraire, sur la **figure 3** illustrant la présente invention, on a mis en évidence la prise en compte des redondances inter-chaînes (soulignées) et la nouvelle façon de les représenter. La chaîne *« Mary Lisa »* référence ainsi la chaîne *« Mary »* et la chaîne *« Londasle »* référence les 4 premiers caractères de la chaîne *« London ».*

On illustre maintenant les mécanismes de l'invention permettant d'aboutir à un tel résultat, notamment en références aux **figures 4** à **11**.

La **figure 4** illustre l'algorithme général d'encodage selon la présente invention. Cet algorithme est notamment mis en œuvre dans un module appelé codeur 501, illustré par la **figure 5**.

L'encodeur 501 prend en entrée un document ou un flux de données XML (502), par exemple celui de la **figure 1**, et génère en sortie un document ou un flux de données binaires (503). Pour cela, le codeur 501 utilise un analyseur XML 504 chargé d'extraire et d'identifier des évènements XML du document XML 502. Il possède également un module de gestion 505 des grammaires 506 et productions telles que définies par exemple par la norme Efficient XML. Ce gestionnaire est également responsable de l'encodage des priorités associées aux différentes productions des grammaires 506.

Egalement, le codeur contient un module de gestion 507 des dictionnaires de valeurs 508. Ce gestionnaire de valeurs 507 insère et récupère des valeurs dans les dictionnaires. Il prend également en charge le codage des index représentant les valeurs indexées ainsi que le codage des valeurs non encore présentes dans les dictionnaires 508.

Enfin un générateur de train binaire 509 est en charge de récupérer les différentes données encodées et de les insérer au fur et à mesure de leur production dans le flux binaire de sortie 503.

De retour à la **figure 4**, l'encodage débute par récupérer des données XML 502 à encoder à l'étape 400. Lors de l'encodage de ces données 502, on peut être amené à encoder une valeur XML. Une valeur XML peut être une valeur d'attribut, une valeur de nœud texte, un nom d'élément XML, une URI d'espace de nommage, un préfixe, une valeur de commentaire… généralement une chaîne de caractères.

On teste si une valeur XML est à encoder à l'étape 405. Si tel est le cas, le gestionnaire de valeurs 507 récupère cette valeur à l'étape 410 et récupère le ou les dictionnaires à utiliser pour l'encodage de cette valeur à l'étape 415. Plusieurs récupérations sont possibles en fonction du format binaire sous-jacent. On récupère en général un ou deux dictionnaires.

Ainsi, dans le format *EXI*, on peut récupérer pour une valeur d'élément ou d'attribut, le dictionnaire local à l'élément/l'attribut et le dictionnaire global des valeurs. Pour un nom local d'élément, on peut récupérer la table des noms locaux. Dans le cas du format *Fast Infoset*, on récupère la table des URI d'espace de nommage pour encoder une URI d'espace de nommage.

Il est à noter que l'on peut aussi partitionner un dictionnaire en deux ou plusieurs sous-dictionnaires, le but étant de permettre un meilleur encodage ultérieur des références à ces dictionnaires comme on le verra en référence à la **figure 7** ci-après.

Ainsi, pour *Fast Infoset* et *EXI*, on peut définir pour un nom local d'élément, un sous-ensemble du dictionnaire des noms locaux en retenant un ensemble composé des noms locaux des éléments parents de l'élément à encoder. Ce partitionnement peut aussi consister en une utilisation d'un dictionnaire local d'un autre élément, le partitionnement étant alors basé sur une étape de détection de redondance d'information entre les deux dictionnaires, cette étape pouvant être effectuée durant l'encodage ou auparavant lors de la configuration de l'encodeur après avoir effectué des analyses statistiques.

Une fois les dictionnaires récupérés, le gestionnaire de valeurs 507 procède à l'encodage proprement dit de la valeur. Cette étape débute par tester s'il reste un caractère de la valeur à encoder à l'étape 420.

Si tel est le cas (sortie OUI de l'étape 420), on effectue une recherche de référence sur la valeur à partir du caractère à encoder à l'étape 425. Cette étape est détaillée par la suite en référence à la **figure 6**.

Suite à cette recherche, on teste si une référence a été trouvée à l'étape 430. Si ce n'est pas le cas (sortie NON de l'étape 430), on encode le

caractère de la valeur à l'étape 435, puis on passe au caractère à encoder suivant à l'étape 440.

Si une référence a été trouvée (sortie OUI à l'étape 430), on encode cette référence à l'étape 445. Cet encodage est décrit par la **figure 7** ci-après. On passe ensuite au caractère suivant à encoder à l'étape 440. Il est à noter que l'étape 435 a permis l'encodage de plusieurs caractères et que l'étape 440 a bien entendu sauté ces caractères.

Lorsque l'ensemble des caractères de la valeur XML a été encodé (sortie NON de l'étape 420), l'encodage de la valeur se termine par l'encodage de la fin de valeur et on peut passer à la valeur suivante à l'étape 405.

On note ici que pour permettre le décodage complet d'une valeur XML comme on le verra par la suite en référence à la **figure 9** par exemple, le codeur doit insérer un symbole spécial ("de fin de valeur") indiquant la fin des données binaires correspondant à cette valeur. Ce symbole peut prendre, sans que ce soit limitatif, les formes suivantes :

- l'encodage du nombre d'octets utilisés pour encoder la valeur ;
- l'encodage du nombre de caractères de la valeur ;
- l'encodage d'un symbole particulier, typiquement 0 à la fin de la valeur.

La première alternative a l'avantage de permettre de sauter très rapidement le décodage d'une valeur si elle n'est pas utile. La troisième alternative permet d'encoder la fin d'une valeur XML de manière adaptative (via un code particulier). On peut aussi envisager d'intégrer la recherche de ce symbole de fin dans l'étape de recherche de référence auquel cas le symbole de fin est parfois encodé via une référence.

Lorsque toutes les données et valeurs XML ont été encodées (sortie NON de l'étape 405), le processus d'encodage se termine à l'étape 450.

On décrit maintenant plus en détail l'étape 425 de recherche de références, illustrée par la **figure 6**.

De façon générale ici, la recherche d'une référence correspond à la recherche d'une sous-chaîne appartenant, à la fois, à la valeur XML à encoder et à une valeur déjà présente dans un dictionnaire de valeurs XML. Plusieurs

approches sont possibles qui se basent sur des algorithmes de recherche de sous-chaîne, par exemple les algorithmes Rabin-Karp, Knuth-Morris-Pratt. On pourra notamment noter la possibilité d'utiliser l'algorithme de Boyer Moore pour une recherche efficace.

L'algorithme général de recherche de référence de la **figure 6** s'appuie sur un algorithme de recherche de sous-chaîne. L'algorithme, mis en œuvre par le gestionnaire de valeurs 507, débute par récupérer une sous-chaîne à rechercher à l'étape 600. Typiquement, cette sous-chaîne est extraite d'une valeur XML à encoder. On commence par rechercher une sous-chaîne XML de petite longueur, trois caractères typiquement.

On récupère, à l'étape 610, les chaînes à partir desquelles on effectue la recherche. Ces chaînes (appelées entrées par la suite) sont récupérées à partir des dictionnaires obtenus à l'étape 415. On peut aussi inclure à cette recherche la partie de la valeur XML qui est déjà encodée. Si une référence est trouvée à partir de la partie déjà encodée de la valeur XML, on parlera par la suite de "référence interne".

On itère ensuite la recherche de la sous-chaîne pour chaque entrée.

On vérifie s'il y a une entrée à tester, à l'étape 620. Dans l'affirmative, on récupère cette entrée et on effectue la recherche de la sous-chaîne dans l'entrée, à l'étape 630, en utilisant par exemple l'algorithme de Boyer Moore. On teste ensuite si une sous-chaîne a été trouvée à l'étape 640. Si ce n'est pas le cas, on effectue la recherche avec l'entrée suivante.

Si une sous-chaîne a été trouvée (sortie OUI de l'étape 640), on effectue un test d'arrêt de la recherche à l'étape 650. Cette recherche peut être arrêtée (étape 670) si le résultat de la recherche est jugé comme suffisant, par exemple si la sous-chaîne a une taille importante. On peut aussi arrêter la recherche si le temps passé est trop important. Un apprentissage du critère d'arrêt est aussi possible : ce critère peut être global à l'ensemble des valeurs XML ou local pour chaque ensemble de valeurs de tel ou tel élément/attribut. On peut ainsi décider de ne tester que les entrées du dictionnaire local. Il est entendu que ce test d'arrêt est facultatif et qu'il peut être envisagé de

poursuivre les recherches tant qu'une sous-chaîne d'une entrée peut être trouvée.

Dans le cas où la recherche peut continuer soit parce qu'aucun test n'est effectué soit parce que le test est négatif (sortie OUI de l'étape 650), on augmente la taille de la sous-chaîne recherchée d'un ou plusieurs caractères à l'étape 660 et on continue la recherche sur les entrées non encore testées. Durant l'étape 660, on termine la recherche dans l'entrée courante de la sous-chaîne en l'augmentant éventuellement.

Une fois toutes les entrées testées (sortie NON de l'étape 620), on met fin à la recherche à l'étape 670. La dernière sous-chaîne trouvée est celle rendue en sortie de l'étape 425.

Dans ce processus, dès qu'une référence est trouvée, on augmente la sous-chaîne de recherche. Cela permet notamment de limiter la recherche et le temps de traitement associé.

En variante, on peut néanmoins continuer la recherche jusqu'à la fin du dictionnaire considéré avant d'augmenter la sous-chaîne. On dispose alors éventuellement d'un choix entre plusieurs entrées.

On illustre ce processus de la **figure 6** par l'exemple suivant :

- on dispose du dictionnaire comprenant les entrées: "marie", "voile", "voiturette" et "car" ;
- on a la valeur "voiture de marie" à encoder ;
- on débute par une recherche sur la sous-chaîne "voi" (étape 600) ;
- la première entrée "marie" ne possède pas cette sous-chaîne ;
- la deuxième entrée "voile" possède cette sous-chaîne ;
- on augmente alors la sous-chaîne d'un caractère "voit" (étape 660) ;
- la deuxième entrée "voile" ne possède pas cette sous-chaîne (toujours étape 660) ;
- la troisième entrée "voiturette" possède cette sous-chaîne ;
- on augmente successivement la sous-chaîne des caractères "u", "r" et "e" jusqu'à obtenir la référence "voiture" qui se trouve dans la troisième entrée ;

- la sous-chaîne suivante augmentée par l'espace " " est alors recherchée. Cette sous-chaîne "voiture " ne se trouve pas dans la troisième entrée "voiturette" ;

- la quatrième entrée "car" ne possède pas cette sous-chaîne ;

- la référence trouvée est alors "voiture" et il reste à encoder la fin de la valeur à encoder, c'est-à-dire " de marie" qui dans l'exemple sera encodé caractère par caractère pour " de ", puis en référence à l'entrée "marie" pour la fin.

Comme on le verra par la suite pour l'encodage de la référence à la troisième entrée, on obtient dans notre exemple la suite suivante de symboles à encoder:

Référence(3,0,7) " de " Référence(1)

Différentes approches pour l'encodage de ces symboles peuvent être proposées.

Il est possible d'avoir des encodages fixes via des tables de Huffman fixes par exemple.

On peut aussi envisager que l'encodeur 501 et le décodeur 801 possèdent un ensemble prédéfini de tables de Huffman et que l'encodeur détermine la table de Huffman utilisée.

On peut aussi, comme dans le format de compression ZIP, définir dans le flux binaire la table de Huffman à utiliser. Ces différentes méthodes ont l'avantage de permettre un décodage rapide, indépendant de la position de l'encodage ce qui peut être pratique lorsqu'on souhaite décoder une valeur une fois l'ensemble du document décodé par exemple.

On peut aussi envisager d'utiliser un algorithme de Huffman adaptatif auquel cas on conserve des statistiques récupérées lors de l'encodage des valeurs et utilisées progressivement pour améliorer l'efficacité de la table de Huffman. On peut ainsi décider de mettre à jour la table de Huffman après l'encodage de chaque symbole (étape 440) ou après l'encodage de chaque valeur (étape 410).

En variante, on peut noter que la même référence peut parfois être trouvée dans plusieurs entrées par le processus. Dans ce cas, différents critères de sélection peuvent être appliqués :

- un critère de compression. Dans ce cas, on avantage les références qui sont encodées plus efficacement : entrée récente, de petite taille, référence de début ou de fin comme on le verra ci-après en référence à la **figure 7** ;

- un critère de rapidité de décodage. Dans ce cas, on utilise l'entrée la plus ancienne de façon à ce que des références de sous-chaînes identiques pointent systématiquement vers la même entrée permettant ainsi un décodage partiel minimal comme on le verra par la suite en référence à la **figure 10** ;

- un critère de proximité, par lequel on privilégie les entrées provenant d'un dictionnaire local.

On décrit maintenant plus en détail l'étape 445 d'encodage des références trouvées précédemment, illustrée par la **figure 7**.

L'algorithme de recherche des références produit, comme on l'a vu ci-dessus, une suite de symboles qui sont encodées par des méthodes de codage, tel que codage de Huffman ou codage arithmétique par exemple. Ces différentes méthodes se basent une connaissance des probabilités d'apparition des différents symboles à encoder. Pour permettre un encodage à la volée, les probabilités sont soit connues à l'avance (approche statique avec des tables prédéfinies) soit mises à jour progressivement durant l'encodage (tables dynamiques), comme expliqué ci-dessus.

L'algorithme d'encodage des références représenté en **figure 7** s'appuie sur ces méthodes de codages tout en prenant en compte les spécificités des données à encoder, notamment l'indice, la longueur et l'offset de la référence.

L'encodage débute à l'étape 700 par récupérer la référence produite aux étapes 425 ou 670, afin de déterminer le type de référence.

On détermine tout d'abord à l'étape 705 s'il s'agit d'une référence interne, c'est-à-dire s'appuyant sur une autre sous-chaîne de la même valeur à encoder.

Dans l'affirmative (sortie OUI de l'étape 705), on encode un code indiquant qu'il s'agit d'une référence interne, à l'étape 710.

Dans la négative, s'il s'agit d'une référence locale (testée à l'étape 715), c'est-à-dire que la valeur XML référencée est indexée dans le dictionnaire local, on encode le symbole correspondant (indiquant qu'il s'agit d'une référence locale), lors de l'étape 720, à l'aide d'une technique de codage classique, type un codage arithmétique ou un codage de Huffman.

Sinon (sortie NON de l'étape 715), il s'agit d'une référence globale et on encode le symbole "référence globale" à l'étape 725.

Dans le cas d'une référence locale ou globale (après les étapes 720 et 725), on encode, à l'étape 730, la valeur d'indice de la référence à l'intérieur par exemple du dictionnaire associé

Afin de limiter le nombre de bits utilisés pour encoder cet indice, on peut effectuer cet encodage de manière simple en encodant l'indice relativement à la taille du dictionnaire: pour un dictionnaire ayant par exemple 213 entrées, on utilise 8 bits, alors qu'avec un dictionnaire ayant 8 entrées, on utilise seulement 3 bits.

En variante, on peut aussi définir des plages d'indices, calculées à partir de la taille T_i du dictionnaire "i" : on affecte à un premier dictionnaire 1 les premiers indices de 0 à T_1-1, puis au deuxième dictionnaire, les indices de T_1 à T_1+T_2-1, etc... Ainsi, on peut déterminer aisément la plage dans laquelle se trouve l'indice. Dans ce cas, on encode l'indice relativement à la plage : la plage définissant un indice minimum (I_{min}) et maximum (I_{max}), on encode alors l'indice I en encodant la valeur ($I-I_{min}$) relativement à la valeur ($I_{max}-I_{min}$). Cette approche permet de minimiser le nombre de bits utilisés pour l'encodage de l'indice.

Dans ce cas, il convient de renseigner dans le flux codé à quelle plage se rapporte l'indice différentiel codé. On peut prévoir l'indication directe du dictionnaire utilisé.

En variante, on définit des codes qui déterminent conjointement le type de référence et la plage d'indices (lorsqu'il s'agit d'une référence locale ou globale). Dans ce cas, on prévoit les codes suivants :

- un code pour un nouveau caractère à encoder ;

- autant de codes que de caractères précédemment encodés ;
- plusieurs codes pour le symbole référence locale, chaque code définissant une plage d'indices ;
- plusieurs codes pour le symbole référence globale, chaque code définissant une plage d'indices.

L'hypothèse sous-jacente à cette approche est que les valeurs XML ayant un indice récent ont plus de chance d'être référencées que les valeurs XML anciennes. Cette propriété peut notamment être amplifiée par la recherche qui peut se faire des valeurs les plus récentes aux valeurs les plus anciennes. En prenant cette hypothèse, on peut alors définir des intervalles de petite taille pour les valeurs récentes et de grande taille pour les valeurs anciennes. On peut alors déterminer les plages proportionnellement à la taille du dictionnaire et en essayant d'obtenir des tailles d'intervalle du type puissance de 2, notamment pour les intervalles de petite taille.

A la fin des étapes 710 et 730, l'identifiant du dictionnaire et l'indice de l'entrée de la référence, le cas échéant, ont été encodés et il reste à renseigner à quelle sous-chaîne de l'entrée correspond la référence, c'est-à-dire à encoder les valeurs de longueur et d'offset.

On observe à ce stade qu'il existe une grande redondance soit du début soit de la fin des valeurs XML. En effet, il est classique d'utiliser des espaces en début ou fin de valeurs, tout comme il est classique d'utiliser des noms préfixés.

A titre d'exemple, le concepteur d'un document XML répertoriant un carnet d'adresse utilise généralement des noms d'éléments préfixés par le terme "address". Ainsi, il y a, dans ce document, une forte récurrence pour des éléments tels <addressBook>, <addressEntry>, <addressName>, et les éléments fin de balise correspondant </addressBook>, etc.

Au regard de l'offset, on distingue différentes catégories de références :

- des références de type début lorsque la sous-chaîne débute au premier caractère de l'entrée référencée, ce qui est le cas dans l'exemple ci-dessus ;

- des références de type fin lorsque la sous-chaîne finit au dernier caractère de l'entrée référencée ; et

- toutes les autres références ne comprenant ni le premier ni le dernier caractère de l'entrée référencée.

Au regard de la longueur de la sous-chaîne, on note que, généralement, les références courtes sont beaucoup plus fréquentes que les références longues.

Ainsi, pour permettre un codage adaptatif efficace de l'offset et la longueur, on prévoit l'ensemble de codes suivant :

- un code de "référence de début", utilisé dans la première catégorie ci-dessus ;

- un code de "référence de fin", utilisé dans la deuxième catégorie ci-dessus ;

- un code "longueur = N", un code "longueur = N+1", ..., un code "longueur = N+M" ;

- un code "longueur grande", bien sûr dans les cas où la longueur de la sous-chaîne est supérieure à N+M, auquel on associera généralement une indication du complément de longueur au-delà de N+M.

La valeur N correspond à la sous-chaîne minimale, soit N=3 dans l'exemple proposé précédemment (voir étape 600). En pratique, on prend typiquement une petite valeur de M, par exemple M=3.

On peut alors effectuer un codage adaptatif de ces différents codes au fur et à mesure de l'apparition de leurs occurrences.

On peut aussi prévoir que les codes utilisés ci-dessus soient affectés statiquement et alignés sur un octet (des codes de longueur 8 bits donc) ce qui facilite le décodage octet par octet (qui est alors à peine plus lent qu'en l'absence de compression) et non bit par bit.

On procède alors à l'encodage à proprement parler des valeurs de longueur et d'offset en testant tout d'abord, à l'étape 735, s'il s'agit d'une référence de type début. Si tel est le cas, on encode, à l'étape 740, le code "référence de début".

Sinon (sortie NON de l'étape 735), on teste s'il s'agit d'une référence de type fin à l'étape 745. Si tel est le cas, on encode le code "référence de fin" à l'étape 750.

Suite aux étapes 740 et 750 ayant permis l'encodage d'une information d'offset, on encode la longueur de la référence (dans le cas d'une référence de type fin, on peut déduire l'offset des longueurs de la valeur et de la référence) à l'étape 755 en utilisant par exemple les codes de longueur prévus ci-dessus.

On peut encoder la longueur de référence relativement à la valeur maximale possible de cette longueur. En pratique, des études statistiques montrent que la probabilité d'apparition des longueurs de références décroît rapidement avec la valeur de la longueur de référence. On peut alors aussi décider d'encoder cette longueur en utilisant un encodage tel que celui de "Golomb exponentiel" qui privilégie fortement les petites valeurs au détriment des fortes valeurs.

On a ainsi codé les références de début et de fin et on met fin à ce codage de référence à l'étape 770, de sorte à poursuivre à l'étape 440.

S'il ne s'agit pas de référence de type début ou de référence de type fin (sorite NON de l'étape 745), on encode alors la longueur de référence directement à l'étape 760 via le code de longueur approprié.

Dans le cas d'une référence de longueur L supérieure à N+M, on encode le code "longueur grande" puis la valeur (L-N+M) via un encodage tel que celui de Golomb exponentiel.

En variante, on peut utiliser de façon successive les codes de longueur ci-dessus pour indiquer la longueur totale. Par exemple, la succession des codes "longueur grande", "longueur grande" et "longueur = N+1" correspond à la longueur (N+M)+(N+M)+(N+1) = 3N + 2M + 1.

On encode ensuite la valeur de l'offset à l'étape 765, correspondant à la distance entre le premier caractère de l'entrée référencée et le début de la sous-chaîne trouvée. Il est de nouveau possible d'effectuer un codage adaptatif de cette valeur, par exemple en adaptant l'encodage suivant la valeur de la longueur de la référence. Mais, la répartition de l'offset étant relativement

uniforme, on encode généralement cette valeur relativement à la valeur d'offset maximum calculée à partir des longueurs de la valeur et de la référence.

Selon une caractéristique visant à optimiser cet encodage, on détermine un ensemble de positions possibles dans la valeur de l'entrée référencée, ces positions étant fonction de critères divers tels que la ponctuation, des lettres capitales, un espace. On encode ainsi la valeur de l'offset relativement à cet ensemble de positions. L'algorithme de sélection des positions peut être similaire à un algorithme de séparation en syllabes de mots.

Deux exemples sont donnés ci-dessous où les lettres soulignées sont les positions sélectionnées :

- 'listeNumeroCible' → 'liste**Nu**mero**Ci**ble'
- ' Ceci est un exemple ' → ' **Ce**ci **es**t **u**n **ex**emp**le** '

On a ainsi codé les références qui ne sont ni de début ni de fin, et on met fin au codage de référence à l'étape 770, de sorte à poursuivre à l'étape 440.

En alternative à ce procédé d'encodage où l'on encode séparément chacune des informations de type de référence, d'indice de référence, de longueur et d'offset, on envisage d'utiliser un dictionnaire de référence, c'est-à-dire un dictionnaire dont chaque entrée associe un index à une référence composée des informations énumérées ci-dessus.

Ainsi, l'encodeur 501 conserve les dictionnaires courants et ajoute un tel dictionnaire de références stockant les informations de référence. Puis, l'encodage de la référence à l'étape 445 s'effectue de manière indexée à ce dictionnaire ou directement de façon standard si l'entrée n'existe pas encore dans le dictionnaire (auquel cas une nouvelle entrée est ajoutée au dictionnaire).

Dans ce cas, le décodeur construit également ce dictionnaire de références au fur et à mesure du décodage.

On note que l'encodeur 501 peut également utiliser un tel dictionnaire de références afin d'accélérer la recherche de références (voir **figure 6**) sans toutefois l'utiliser au niveau de l'encodage proprement dit des

références. Cette dernière possibilité réduit le temps d'encodage tout en limitant les besoins en ressource mémoire du décodeur.

Non représenté par les figures, l'encodage des valeurs typées d'un document XML peut être légèrement amélioré. Comme aucun dictionnaire n'est généralement construit lors de l'encodage standard de telles valeurs typées en raison de la spécialisation de cet encodage, il est difficile d'intégrer le système de références évoqué ci-dessus directement dans l'encodage typé.

Néanmoins, on peut prévoir au choix :

- d'intégrer des valeurs typées dans un ou plusieurs dictionnaires, et donc d'appliquer la présente invention également à ces valeurs typées ;

- d'ajouter une mémoire tampon (ou *buffer*) flottante pour le traitement de ces valeurs typées. Les valeurs typées sont encodées et envoyées en tant que suites d'octets et sont aussi conservées dans cette mémoire flottante, auquel cas, la recherche de références s'applique à cette mémoire flottante, en lieu et place ou en complément du dictionnaire.

Avec ces deux possibilités, on peut alors effectuer des références afin de réduire la redondance de ces valeurs typées.

On a vu ci-dessus l'encodage des valeurs d'un document XML à l'aide du procédé objet de la présente invention. Un tel document XML comprend en outre des données de structure dont on prévoit de réaliser l'encodage de façon parallèle. La combinaison des deux encodages assurant une compression efficace du document structuré.

Cet encodage peut être réalisé selon les mécanismes classiques d'encodage de Fast Infoset ou d'EXI, par exemple à l'aide des grammaires et productions associées.

Afin d'améliorer la compression des données, on envisage de générer, à partir des grammaires décrivant les éléments du document XML, une séquence d'événements, et de comparer cette séquence d'événements aux événements à coder.

Pour cela, on considère successivement les différentes grammaires concernées (par exemple les grammaires décrivant les éléments "person", "firstName", "lastName" et "city" dans l'exemple ci-dessus), et l'on liste les

productions ayant une priorité de niveau 0. On peut également établir une succession de productions qui s'enchaînent dans le document en introduisant, par exemple dans celles-ci, une référence ou un pointeur vers la production suivante.

On compare ainsi la séquence à coder, par exemple deux événements (données de structure) consécutifs, à la séquence générée, sans nécessairement chercher à avoir des événements strictement identiques mais simplement que les informations contenues dans les grammaires soient bien vérifiées (c'est-à-dire à ce que les événements soient de même type et possèdent le même nom dans le cas d'événement « début d'élément » ou « attribut »).

En cas de comparaison positive, on se trouve dans une situation où l'on est capable de prédire, à l'aide des grammaires générées à l'endroit courant du document, les événements suivants.

Ainsi, le résultat de l'évaluation est alors encodé sur 1 bit : en cas d'évaluation positive (les données sont conformes aux grammaires), il suffit alors de coder les informations qui ne sont pas présentes dans les grammaires (par exemple les valeurs des événements de contenu textuel) ; en revanche, en cas d'évaluation négative, on code les données de manière basique, c'est-à-dire en codant les productions et les valeurs (comme on le ferait avec Efficient XML). Dans le premier cas, on a codé toutes les productions sur un seul bit avec un gain notable de compression ; dans le second cas, on perd simplement un bit par rapport à Efficient XML.

Dans un mode de réalisation plus performant utilisant toujours cette approche prédictive, on prévoit de coder le nombre de prédictions correctes successives afin de produire un flux binaire encore plus compact: par exemple on codera 10 prédictions correctes successives sur 4 bits, au lieu de coder dix fois 1 bit de prédiction (10 bits seraient alors nécessaires).

Pour chaque prédiction correcte, il se peut que certaines informations ne soient pas prédites (typiquement, la valeur d'un événement de contenu textuel ou d'un attribut), auquel cas on encode ces informations

(suivant l'encodage classique EXI) et on les stocke dans une mémoire tampon (buffer).

Lorsque l'on rencontre une prédiction incorrecte, on encode le nombre de prédictions correctes courant, puis l'on copie les données bufférisées dans le flux encodé, et ensuite le code de la production décrivant l'événement pour lequel la prédiction a échoué. Le nombre de prédictions correctes est alors remis à zéro, et le processus reprend.

On décrit maintenant le procédé de décodage correspondant au procédé d'encodage décrit précédemment en référence aux **figures 8** à **10**.

Diverses variantes ont été présentées lors de la description de l'encodage et peuvent trouver leur pendant au niveau du décodeur. On comprend ici que même si l'on ne décrit pas l'ensemble de ces variantes appliquées au décodage, l'homme du métier saurait les transposer sans difficulté.

Le procédé de décodage prend place au sein d'un décodeur 801 comprend ainsi les mêmes principaux modules internes que le codeur 501.

En référence à la **figure 8**, on retrouve ainsi les gestionnaires de grammaires 805 et de valeurs 807 ainsi que leurs grammaires 806 et dictionnaires 808 respectifs. Toutefois le flux de données se trouve inversé : les données d'entrée sont maintenant le flux binaire 803 codé résultant à l'étape 450, alors que le résultat issu du décodage correspond au document (ou flux de données) XML 802. Ce document XML 802 est généré par un générateur XML tandis que le flux binaire d'entrée 803 est géré par l'analyseur binaire 809 qui alimente les deux gestionnaires 805 et 807.

En référence à la **figure 9**, on décrit maintenant le procédé de décodage correspondant.

A l'étape 900, le décodeur 801 récupère les informations du flux binaire via l'analyseur de flux binaire 809.

On teste ensuite si un item XML est à décoder, à l'étape 905. Si tel est le cas (sortie OUI), on décode l'item XML à l'étape 910. Lorsque tous les items sont décodés, le procédé se termine à l'étape 950.

Ce même analyseur peut alors tester, à l'étape 915, si l'item nécessite le décodage d'une valeur (nom d'un élément, valeur d'un attribut...). Si aucun décodage de valeur n'est à effectuer, on repart à l'étape 905. Ce peut notamment être le cas si l'item à décoder correspond à une information de structure. Dans ce cas, cet item à décoder est traité de façon classique par le gestionnaire de grammaires 805.

Sinon (sortie OUI de l'étape 915), le gestionnaire de valeurs 807 effectue le décodage de la valeur en débutant par l'étape 920.

A cette étape 920, on extraie un code du flux binaire 903. Ce décodage peut utiliser un décodage arithmétique ou de Huffman et peut nécessiter ou pas la mise à jour de statistiques de manière symétrique au procédé d'encodage.

Le gestionnaire de valeurs 807 détermine ensuite si ce code correspond à une référence ou à un caractère à l'étape 925. Dans le cas où le code correspond à un caractère, on ajoute, à l'étape 930, ce caractère à la chaîne décodée en construction.

S'il s'agit d'une référence (sortie NON de l'étape 925), on décode alors les informations de référence (code de référence, indice, longueur et offset) à l'étape 935 de manière symétrique au procédé d'encodage. Cette étape peut nécessiter l'utilisation d'un décodage de Huffman, décodage arithmétique et/ou la mise à jour de statistiques.

Une fois la référence entièrement décodée, on accède à la sous-chaîne référencée dans les dictionnaires, puis on ajoute à l'étape 940 la sous-chaîne référencée à la chaîne décodée en construction.

Suite aux étapes 930 et 940 d'enrichissement de la chaîne décodée en construction, on teste si la chaîne est terminée à l'étape 945. Ce test de fin de chaîne peut se baser, comme décrit pour l'encodage, sur le nombre d'octets lus pour le décodage de la chaîne, le nombre d'octets de la chaîne ou la présence d'un symbole de fin de chaîne (caractère décodé à l'étape 930 ou caractère présent à la fin de la sous-chaîne).

Si la chaîne n'est pas terminée, on décode un nouveau code à l'étape 920. Sinon, on passe au nouvel item à décoder (étape 905).

En commençant depuis le premier item du flux binaire au dernier item, on procède ainsi au décodage de tout le flux binaire de sorte à récupérer l'ensemble du document XML précédemment codé par le procédé de l'invention.

En référence à la **figure 10**, on prévoit la mise en place d'un décodage partiel qui permet d'accéder à une partie seulement du document XML codé sans nécessiter le décodage de tout le début du document.

Cette approche est particulièrement utile lorsqu'une partie des données XML seulement est utile à l'application, ce qui est par exemple le cas lors d'une recherche basée sur le langage XPath (contraction de *XML Path Language*).

Pour rappel, XPath est une spécification définissant une syntaxe pour adresser des parties d'un document XML. Cette syntaxe utilise une notation similaire aux expressions de chemins dans un système de fichiers, par exemple "/list/person/city" pour l'exemple de la **figure 1**. Cette syntaxe sert de base à la formation de requêtes sur des documents en vue de leur transformation, d'un accès rapide à des sous-parties ou d'effectuer des traitements sur des parties du document. XPath a pour principal avantage de simplifier le développement d'applications appelées à naviguer dans des documents XML. L'entité responsable de l'évaluation d'une expression XPath est appelée « Processeur XPath ». Le processeur XPath accepte généralement comme entrées d'une part une ou plusieurs expressions XPath et d'autre part une référence sur des données XML (lues dans un fichier ou reçues via une transmission réseau) sur lesquelles la ou les expressions doivent être évaluées.

Le procédé de décodage partiel débute, à l'étape 1000, par l'obtention du flux binaire.

A l'étape 1005, on teste si l'on doit décoder un item. Ceci est indiqué par le processeur XPath analysant l'expression XPath représentant la partie des données intéressant l'application. Il s'agit notamment des items désignés par le chemin XPath mais également des items qui ont servi à déterminer les items désignés. Si tel est le cas, le gestionnaire de valeurs 807 effectue le décodage de l'item à l'étape 1010. Il s'agit ici de décoder, au travers de l'item,

une information de structure du document XML telle qu'encodée comme décrit précédemment. Notamment, on peut récupérer l'information de "conformité" renseignant d'une prédiction effectuée, et réalisée celle-ci à partir des grammaires courantes pour déterminer cette information de structure.

A l'issue de ce décodage, l'application détermine si une valeur associée à l'item, s'il y en a une, est nécessaire au traitement, c'est-à-dire si elle est utile à la détermination des items désignés par l'expression XPath. Ce test est effectué à l'étape 1015. Si la valeur n'est pas nécessaire, on peut alors stocker cette valeur en mémoire, à l'étape 1020. Dans le cas où il n'y a pas de valeur, aucun stockage n'est effectué et on repasse directement à l'étape 1005 pour le décodage d'un item suivant.

Le stockage d'une valeur dépend de différents paramètres :

- paramètres d'encodage : si l'encodage des codes est dynamique, on doit nécessairement récupérer l'ensemble des informations statistiques permettant le décodage des valeurs futures. Si l'encodage est statique, aucun traitement particulier n'est nécessaire
- type de flux de données : si le flux est persistent (fichier), on peut ne conserver qu'un pointeur vers les données ; si le flux n'est pas persistent, on peut stocker les données dans une mémoire tampon.

Ce stockage conserve ainsi les valeurs rencontrées sous une forme compressée. Le stockage 1020 inclut aussi une étape d'insertion si nécessaire de la valeur stockée dans le dictionnaire afin de tenir à jour ces derniers.

Dans le cas où la valeur est nécessaire (sortie OUI de l'étape 1015), le gestionnaire de valeurs 807 procède au décodage total de cette valeur.

Pour ce faire, il débute par décoder le premier code de la valeur à l'étape 1025.

Si ce code correspond à un symbole caractère (étape 1030), on ajoute le caractère à la chaîne décodée en cours de reconstruction.

Si ce code correspond à une référence, on décode l'information de référence à l'étape 1040. Cette information permet de récupérer notamment l'entrée du dictionnaire référencée. On teste alors, à l'étape 1045, si cette entrée du dictionnaire est déjà décompressée ou pas. Si c'est le cas, à l'étape

1055, on extrait la sous-chaîne référencée et on l'ajoute à la chaîne décodée. Si l'entrée n'est pas décompressée (sortie NON de l'étape 1045), on décode, à l'étape 1050, l'entrée référencée avant d'effectuer l'étape d'ajout de la sous-chaîne référencée (étape 1055).

Le décodage de l'étape 1050 dépend du format de stockage utilisé. Si l'entrée a été conservée sous sa forme compressée, on effectue le même processus que le procédé des étapes 1025 à 1060. Si l'entrée a été partiellement décompressée (mise sous forme de symboles et non de codes par exemple) on peut faire de même. Une variante de cet algorithme consiste à intégrer les étapes 1050 et 1055 pour ne décoder que la partie pertinente à la référence. Dans ce cas, l'encodeur 501 effectuera de préférence une recherche de référence des entrées les plus anciennes aux entrées les plus récentes pour obtenir le maximum de références à la même entrée du dictionnaire et ainsi minimiser le nombre d'entrées décompressées.

Suite aux étapes 1035 et 1055, on vérifie, à l'étape 1060, si le décodage complet de la chaîne est terminé. Si ce n'est pas le cas, on effectue le décodage du code suivant à l'étape 1025. Sinon, on passe au test de l'étape 1005 qui détermine si le flux est terminé ou si l'application a récupéré l'ensemble des données nécessaires.

Une fois tout le flux souhaité décodée, on met fin au traitement à l'étape 1065.

Dans un mode de réalisation en variante utilisant un flux binaire codé persistant, par exemple sous la forme d'un fichier stocké en local, on peut prévoir que les entrées des dictionnaires ne contiennent pas les flux binaires correspondant aux valeurs encodées, mais uniquement un pointeur direct vers ces flux binaires à l'intérieur du fichier stocké. On limite ainsi la taille du dictionnaire.

Dans ce cas, la décompression de l'étape 1050 consiste à aller chercher tout d'abord le flux binaire dans le fichier stocké, puis à décompresser ce flux.

En référence à la **figure 11**, il est maintenant décrit à titre d'exemple une configuration matérielle particulière d'un dispositif de traitement d'information apte à une mise en œuvre du procédé selon l'invention.

Un dispositif de traitement d'information mettant en œuvre l'invention est par exemple un micro-ordinateur 1100, une station de travail, un assistant personnel, ou un téléphone mobile connecté à différents périphériques. Selon encore un autre mode de réalisation de l'invention, le dispositif de traitement d'information se présente sous la forme d'un appareil photographique muni d'une interface de communication pour autoriser une connexion à un réseau.

Les périphériques reliés au dispositif de traitement d'information comprennent par exemple une caméra numérique 1240, ou un scanner ou tout autre moyen d'acquisition ou de stockage d'images, reliée à une carte d'entrée/sortie (non représentée) et fournissant au dispositif de traitement d'information des données multimédia, éventuellement sous forme de documents XML.

Le dispositif 1100 comporte un bus de communication 1110 auquel sont reliés :

- Une unité centrale de traitement CPU 1120 se présentant par exemple sous la forme d'un microprocesseur ;

- Une mémoire morte 1130 dans laquelle peuvent être contenus les programmes dont l'exécution permet la mise en œuvre du procédé selon l'invention ;

- Une mémoire vive 1140 qui, après la mise sous tension du dispositif 1100, contient le code exécutable des programmes de l'invention ainsi que des registres adaptés à enregistrer des variables et paramètres nécessaires à la mise en œuvre de l'invention, notamment les tables et grammaires de la **figure 1** ;

- Un écran 1150 permettant de visualiser des données et/ou de servir d'interface graphique avec l'utilisateur qui peut ainsi interagir avec les programmes de l'invention, à l'aide d'un clavier 1160 ou de tout autre moyen tel qu'un dispositif de pointage, comme par exemple une souris 1170 ou un crayon optique ;

- Un disque dur 1180 ou une mémoire de stockage, telle qu'une mémoire de type compact flash, pouvant comporter les programmes de l'invention ainsi que des données utilisées ou produites lors de la mise en œuvre de l'invention ;

- Un lecteur de disquette 1190 optionnel, ou un autre lecteur de support de données amovible, adapté à recevoir une disquette 1210 et à y lire / écrire des données traitées ou à traiter conformément à l'invention ; et

- Une interface de communication 1200 reliée au réseau de télécommunications 1220, l'interface 1200 étant apte à transmettre et à recevoir des données.

Dans le cas de données audio, le dispositif 1100 est équipé de préférence d'une carte d'entrée/sortie (non représentée) qui est reliée à un microphone 1230.

Le bus de communication 1110 autorise une communication et une interopérabilité entre les différents éléments inclus dans le dispositif 1100 ou reliés à celui-ci. La représentation du bus 1110 n'est pas limitative et, notamment, l'unité centrale 1120 est susceptible de communiquer des instructions à tout élément du dispositif 1100 directement ou par l'intermédiaire d'un autre élément du dispositif 1100.

Les disquettes 1210 peuvent être remplacées par tout support d'information tel que, par exemple, un disque compact (CD-ROM) réinscriptible ou non, un disque ZIP ou une carte mémoire. D'une manière générale, un moyen de stockage d'information, lisible par un micro-ordinateur ou par un microprocesseur, intégré ou non au dispositif de traitement d'information, éventuellement amovible, est adapté à mémoriser un ou plusieurs programmes dont l'exécution permet la mise en œuvre du procédé selon l'invention.

Le code exécutable permettant au dispositif de traitement d'information la mise en œuvre de l'invention peut être indifféremment stocké en mémoire morte 1130, sur le disque dur 1180 ou sur un support numérique amovible tel que par exemple une disquette 1210 comme décrite précédemment. Selon une variante, le code exécutable des programmes est reçu par l'intermédiaire du réseau de télécommunications 1220, via l'interface

1200, pour être stocké dans un des moyens de stockage du dispositif 1100 (tel que le disque dur 1180 par exemple) avant d'être exécuté.

L'unité centrale 1120 commande et dirige l'exécution des instructions ou portions de code logiciel du ou des programmes de l'invention, les instructions ou portions de code logiciel étant stockées dans l'un des moyens de stockage précités. Lors de la mise sous tension du dispositif 1100, le ou les programmes qui sont stockés dans une mémoire non volatile, par exemple le disque dur 1180 ou la mémoire morte 1130, sont transférés dans la mémoire vive 1140 qui contient alors le code exécutable du ou des programmes de l'invention, ainsi que des registres pour mémoriser les variables et paramètres nécessaires à la mise en œuvre de l'invention.

On notera également que le dispositif mettant en œuvre l'invention ou incorporant celle-ci est réalisable aussi sous la forme d'un appareil programmé. Par exemple, un tel dispositif peut alors contenir le code du ou des programmes informatiques sous une forme figée dans un circuit intégré à application spécifique (ASIC).

Le dispositif décrit ici et, particulièrement, l'unité centrale 1120, sont susceptibles de mettre en oeuvre tout ou partie des traitements décrits en lien avec les **figures 4** à **10**, pour mettre en œuvre le procédé objet de la présente invention et constituer le dispositif objet de la présente invention.

Ainsi, selon l'invention on améliore le codage et le décodage de documents structurés tels que le XML tout en conservant les avantages du format XML.

En particulier, dans le cadre de *Fast Infoset*, l'invention s'applique uniquement à la compression de valeurs, en définissant un encodage spécifique. Dans ce cas, on conserve la conformité du flux ainsi compressé avec la norme ISO *Fast Infoset*.

En variante, on peut envisager de compresser les valeurs, les noms locaux, préfixes et URI, auquel cas on ne conserve pas la conformité avec la norme ISO *Fast Infoset*, avec cependant une mise à jour peu coûteuse des implémentations *Fast Infoset* standard pour accepter ce nouveau flux.

En outre, il est possible de définir, dans le préambule d'un flux *Fast Infoset*, des vocabulaires et/ou de faire référence à des vocabulaires externes, par exemple des vocabulaires définis dans un schéma XML. Ainsi, il pourra être envisagé de tenir compte de ces différents vocabulaires pour compresser un document XML selon la présente invention.

Dans le cas d'*EXI*, l'encodage des chaînes de caractères peut être redéfini conformément au procédé de l'invention via la fonctionnalité des "*pluggable codec*", permettant de conserver une conformité du flux compressé avec la recommandation *EXI*.

Enfin, Fast Infoset et EXI peuvent s'appuyer sur des dictionnaires pré-remplis auxquels il peut être fait référence selon l'invention.

Comme déjà précisé, l'invention permet ainsi de diminuer la taille des dictionnaires utilisés soit parce que les valeurs intégralement codées à l'aide de références n'entraînent pas de nouvelle insertion dans le dictionnaire, soit parce qu'une nouvelle insertion dans ce dictionnaire pour une nouvelle valeur à coder fait référence à une autre entrée et, de ce fait, utilise un nombre de bits d'information réduit dans la nouvelle entrée insérée.

On utilisera cette dernière solution de préférence dans les cas où il subsiste une partie de la valeur à coder qui ne peut être référencée efficacement (par exemple à l'aide d'au moins trois caractères communs) par une autre entrée du dictionnaire.

Dans la première solution de codage, c'est-à-dire sans création d'une nouvelle entrée dans le dictionnaire lorsqu'une chaîne peut être référencée dans son intégralité, le mécanisme de signalisation pour les noms et les valeurs est étendu afin de permettre un décodage efficace du flux binaire généré lors du codage.

Dans le cas du standard EXI par exemple, le mécanisme de signalisation prend classiquement la forme suivante :

- pour un préfixe représentant un espace de nommage ou la valeur de cet espace de nommage, l'encodeur génère le code 0 s'il s'agit d'une nouvelle valeur ou génère le code ID+1 s'il s'agit d'une valeur déjà indexée, où

ID représente l'entrée dans le dictionnaire correspondant au nom du préfixe ou à sa valeur ;

- concernant la partie locale des noms d'éléments ou d'attributs, un octet de valeur 0 permet d'indiquer que ce nom est indexé ;
- enfin, concernant la valeur d'un nœud de type texte ou la valeur d'un attribut, un octet égal à 0 indique que la valeur existe dans le dictionnaire local de valeurs tandis qu'un octet égal à 1 est utilisé pour indiquer qu'elle se trouve dans le dictionnaire global.

Dans la cas de la présente invention où le codage de valeurs ou de noms n'entraîne pas la création d'une nouvelle entrée/d'un nouvel index dans les dictionnaires, le mécanisme de signalisation présenté ci-dessus est modifié afin de prendre en compte non seulement les cas classiques (« *valeur ou nom indexé* », « *nouveau nom ou nouvelle valeur* ») mais également d'autres cas comme par exemple « *valeur ou nom pouvant être codé entièrement à partir d'une seule référence* » (par exemple « Live » lorsque « Liverpool » est indexé) et « *valeur ou nom pouvant être codé entièrement à partir de plusieurs références* » (par exemple « Will Smith » lorsque « Williams » et « Smith » sont déjà indexés auquel cas la valeur est encodée en référence à ces deux valeurs).

On a vu ci-dessus qu'en cas de « *valeur ou nom que ne peut être que partiellement codé à partir d'une ou plusieurs références* » (par exemple « Mary Lisa » lorsque seule « Mary » est indexée et que « Lisa » doit être codée), on crée une nouvelle entrée dans le dictionnaire. On obtient ainsi une gestion moins coûteuse de ce cas (par rapport à l'absence de nouvelle entrée dans le dictionnaire) eu égard aux coûts de signalisation et de représentation de toutes les parties de la valeur à coder..

En détail, pour un préfixe représentant un espace de nommage ou la valeur de cet espace de nommage, l'encodeur génère le code 0 s'il s'agit d'une nouvelle valeur ; 1 s'il s'agit d'une « *valeur ou nom pouvant être codé entièrement à partir d'une seule référence* » ; 2 s'il s'agit d'une « *valeur ou nom pouvant être codé entièrement à partir de plusieurs références* » et ID+3 sinon.

L'ensemble de ces codes est encodé sur un nombre de bits égal à *2 + $log_2(m+1)$* où *m* est le nombre d'entrées dans la table.

Concernant la partie locale des noms d'éléments ou d'attributs, un mot de 2 bits est formé de telle sorte que : le code *00* indique que ce nom est indexé, l'index est alors encodé classiquement ; le code *01* indique que ce nom correspond à une « *valeur ou nom pouvant être codé entièrement à partir d'une seule référence* », la référence étant ensuite encodée ; le code *10* indique que ce nom correspond à une « *valeur ou nom pouvant être codé entièrement à partir de plusieurs références* », le nombre de références étant ensuite encodé puis la liste de ces références ; et le code *11* indique un nouveau nom dont la longueur est ensuite encodée suivie de sa valeur.

Enfin, concernant les valeurs de nœuds de type texte ou les valeurs d'attributs, le même mécanisme est mis en œuvre, cette fois sur un mot de 3 bits : le code *000* indique que la valeur existe dans le dictionnaire de valeurs local tandis que le code *001* indique qu'il se trouve dans le dictionnaire global ; le code *010* indique une « *valeur ou nom pouvant être codé entièrement à partir d'une seule référence* », cette référence étant alors encodée ; le code *011* indique une « *valeur ou nom pouvant être codé entièrement à partir de plusieurs références* » dont le nombre est d'abord encodé avant l'encodage de chacune de ces références ; et le code *100* indique une nouvelle valeur à indexer suivie de sa longueur puis de sa valeur.

Ces différents codes de signalisation permettent au décodeur d'identifier la façon de reconstruire une chaîne de caractères codées.

Les exemples qui précèdent ne sont que des modes de réalisation de l'invention qui ne s'y limite pas.

REVENDICATIONS

1. Procédé de codage d'un document structuré (502) en un flux binaire (503, 803), le document structuré comprenant des valeurs à coder, le procédé comprenant une étape de codage (445), à partir d'au moins un dictionnaire d'indexation (508) comprenant des entrées, d'au moins une valeur à coder, **caractérisé en ce qu'**il comprend les étapes suivantes:

- l'identification (425, 630) d'au moins une valeur d'entrée du dictionnaire liée à la valeur à coder par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur;
- le codage (740, 750, 755, 760, 765) de ladite valeur à coder par référence, selon ladite correspondance, à au moins l'entrée identifiée du dictionnaire.

2. Procédé de codage selon la revendication précédente, dans lequel lesdites valeurs comprennent des chaînes de caractères et ladite identification (630) comprend la recherche de la chaîne ou d'une sous-chaîne de caractères de la valeur à coder dans une pluralité de valeurs d'entrée du dictionnaire (508).

3. Procédé de codage selon la revendication précédente, dans lequel, lorsque ladite identification (630) comprend la recherche d'une sous-chaîne, on itère (660), jusqu'à ce qu'un critère soit atteint (650) et tant que la recherche est positive:

- l'ajout d'au moins un caractère à ladite sous-chaîne de recherche;
- ladite recherche de la sous-chaîne ainsi augmentée dans ladite pluralité de valeurs d'entrée du dictionnaire.

4. Procédé de codage selon l'une des revendications 2 et 3, dans lequel ledit dictionnaire comprend une entrée correspondant à une sous-chaîne déjà codée de ladite valeur à coder de telle sorte qu'une sous-chaîne ultérieure de ladite valeur à coder est codée en référence à ladite sous-chaîne déjà codée.

5. Procédé de codage selon l'une des revendications précédentes, dans lequel ledit au moins un dictionnaire comprend une pluralité d'entrées correspondant à des valeurs prédéfinies avant codage.

6. Procédé de codage selon l'une des revendications précédentes, comprenant au moins :

- une étape préalable de codage d'une valeur antérieure précédent ladite valeur à coder dans le document,
- une étape préalable d'ajout, dans ledit au moins un dictionnaire, d'une entrée correspondant à la valeur antérieure, et

la référence lors dudit codage (740, 750, 755, 760, 765) par référence portant sur ladite entrée ajoutée.

7. Procédé de codage selon l'une des revendications 1 à 6, dans lequel ledit codage par référence comprend le codage d'une information d'identification dudit dictionnaire d'indexation et de ladite entrée identifiée dans le dictionnaire, d'une information de positionnement dans ladite valeur d'entrée identifiée et d'une information de longueur.

8. Procédé de codage selon l'une des revendications 1 à 6, dans lequel ledit codage par référence comprend le codage d'une donnée de codage correspondant à une entrée d'au moins un dictionnaire de références, une entrée dudit dictionnaire de références renseignant une information d'identification d'une entrée d'un dictionnaire d'indexation, une information de positionnement et une information de longueur.

9. Procédé de codage selon l'une des revendications 7 et 8, dans lequel ladite information d'identification comprend un identifiant d'un dictionnaire parmi une pluralité de dictionnaires d'indexation, et un identifiant (730) de ladite entrée identifiée à l'intérieur dudit dictionnaire.

10. Procédé de codage selon l'une des revendications 7 à 9, dans lequel l'information de positionnement comprend un premier code lorsque ladite partie ou sous-partie de la valeur à coder correspond au début de la valeur d'entrée identifiée, d'un deuxième code lorsque ladite partie ou sous-partie de la valeur à coder correspond à la fin de la valeur d'entrée identifiée, et une valeur

d'offset de la position de ladite partie ou sous-partie de la valeur à coder dans la valeur d'entrée identifiée, dans les autres cas.

11. Procédé de codage selon la revendication précédente, dans lequel on détermine des positions préférentielles dans la valeur d'entrée identifiée, et ladite information de positionnement est établie, au moins dans lesdits autres cas, relativement à au moins une desdites positions préférentielles déterminées.

12. Procédé de codage selon l'une des revendications 7 à 11, dans lequel l'information de longueur comprend un code parmi un ensemble fini de codes de longueur correspondant respectivement à un ensemble fini de longueurs, et un troisième code combiné à une valeur pour les longueurs non comprises dans ledit ensemble.

13. Procédé de codage selon la revendication précédente lorsqu'elle dépend de l'une des revendications 10 et 11, dans lequel lesdites informations de positionnement et de longueur sont encodées conjointement par un encodage adaptatif desdits premier code, deuxième code, troisième code et codes de longueur.

14. Procédé de codage selon l'une quelconque des revendications précédentes, dans lequel ledit document structuré (502) comprend des données de structure à coder, le procédé comprenant les étapes suivantes de codage d'une donnée de structure :

- la prédiction d'au moins une partie des informations composant ladite donnée de structure, à partir d'une grammaire de codage ;
- si la prédiction est positive, l'encodage d'une information dite « de conformité » représentative de ce résultat positif, et l'encodage des autres informations de ladite donnée de structure non comprise dans ladite partie.

15. Procédé de codage selon la revendication précédente, dans lequel on effectue une pluralité de prédictions pour des données de structure successives, ledit procédé comprenant :

- une étape de détermination s'il existe une succession de prédictions correctes pour des données de structure, et

- si le résultat de ladite étape de détermination est positif, une étape de codage du nombre de prédictions correctes successives.

16. Procédé de codage selon l'une quelconque des revendications précédentes, comprenant une étape, préalable à ladite identification, de sélection d'au moins un dictionnaire parmi une pluralité de dictionnaires, ladite sélection étant fonction du type de valeur à coder.

17. Procédé de décodage d'un flux codé binaire (803) d'un document structuré (802) comprenant des valeurs décodées, le procédé comprenant une étape de décodage (920, 935, 1025, 1040), à partir d'au moins un dictionnaire de décodage (808) comprenant des entrées, de données codées correspondant à une valeur décodée, **caractérisé en ce que** ledit décodage comprend les étapes suivantes :

- l'identification (925, 1030), dans lesdites données codées, et le décodage (935, 1040) d'au moins une information de référence liant la valeur d'une entrée du dictionnaire de décodage à la valeur décodée par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur ;

- la récupération de ladite partie ou sous-partie de la valeur de l'entrée à partir de ladite référence décodée de sorte à former (940, 1055), au moins en partie, la valeur décodée.

18. Procédé de décodage selon la revendication précédente, dans lequel ledit décodage des données codées correspondant à la valeur décodée comprend une succession de décodages partiels de parties desdites données codées pour former des sous-parties décodées de la valeur décodée et la concaténation desdites sous-parties de sorte à former ladite valeur décodée, chaque décodage partiel comprenant l'un parmi :

- lesdits identification (1030) et décodage (1040) d'une information de référence suivis de ladite récupération ; et

- un décodage direct (1025) d'au moins un caractère.

19. Procédé de décodage selon la revendication précédente, dans lequel on identifie et on stocke (1020) dans ledit dictionnaire, préalablement audit décodage des données codées correspondant à la valeur décodée, une

pluralité de données codées correspondant à une pluralité d'autres valeurs codées dudit document,

et ladite récupération comprend le décodage des données codées associées à l'entrée référencée.

20. Procédé de décodage selon l'une des revendications 17 à 19, comprenant une étape de détermination, dans ledit flux codé binaire, desdites données codées correspondant à la valeur décodée à l'aide d'au moins une indication renseignant la fin de ladite valeur codée.

21. Procédé d'accès à une partie d'un document structuré (802) comprenant le décodage dudit document selon le procédé de la revendication 17, dans lequel on identifie (1015) et on stocke (1020) ladite pluralité de données codées pour les valeurs codées situées, dans ledit document, avant ladite partie à accéder,

et on décode ladite partie à accéder, ledit décodage de la partie à accéder comprenant, pour au moins une valeur codée de ladite partie à accéder, lesdits identification (1030) et décodage (1040) d'une information de référence suivis de ladite récupération.

22. Système (501) de codage d'un document structuré (502) comprenant des valeurs à coder, le système comprenant des moyens de codage (509), à partir d'au moins un dictionnaire d'indexation, (508) comprenant des entrées, d'au moins une valeur à coder, **caractérisé en ce qu'**il comprend un moyen d'identification d'au moins une valeur d'entrée du dictionnaire liée à la valeur à coder par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur,

et les moyens de codage sont aptes à coder ladite valeur à coder par référence, selon ladite correspondance, à au moins ladite entrée identifiée du dictionnaire.

23. Système (801) de décodage d'un flux codé binaire (803) d'un document structuré (802) comprenant des valeurs décodées, le dispositif comprenant des moyens de décodage (804), à partir d'au moins un dictionnaire de décodage (808) comprenant des entrées, de données codées correspondant à une valeur décodée, caractérisé en ce qu'il comprend :

- un moyen d'identification, dans lesdites données codées, et de décodage d'au moins une information de référence liant la valeur d'une entrée du dictionnaire de décodage à la valeur décodée par correspondance entre au moins une partie de l'une de ces deux valeurs et une sous-partie de l'autre valeur ;

- un moyen de récupération de ladite partie ou sous-partie de la valeur de l'entrée à partir de ladite référence décodée de sorte à former, au moins en partie, la valeur décodée.

24. Moyen de stockage d'informations, éventuellement totalement ou partiellement amovible, lisible par un système informatique, comprenant des instructions pour un programme informatique adapté à mettre en œuvre le procédé conforme à l'une quelconque des revendications 1 à 21, lorsque le programme est chargé et exécuté par le système informatique.

25. Produit programme d'ordinateur lisible par un microprocesseur, comprenant des portions de code logiciel adaptées à mettre en œuvre le procédé selon l'une quelconque des revendications 1 à 21, lorsqu'il est chargé et exécuté par le microprocesseur.

```
<list>
<person>
        <firstName>Mary</firstName>
        <lastName>Smith</lastName>
        <city>London</city>
</person>
<person>
        <firstName>John</firstName>
        <lastName>Williams</lastName>
        <city>Liverpool</city>
</person>
<person>
        <firstName>Mary Lisa</firstName>
        <lastName>Londasle</lastName>
        <city>Liverpool</city>
</person>
</list>
```

Figure 1

Character and Attribute Values	
0	« Mary »
1	« Smith »
2	« London »
3	« John »
4	« Williams »
5	« Liverpool »
6	« Mary Lisa »
7	« Londasle »

Figure 2

Character and Attribute Values	
0	« Mary »
1	« Smith »
2	« London »
3	« John »
4	« Williams »
5	« Liverpool »
6	Ref(0), « Lisa »
7	Ref(2,0,4), « asle »

Figure 3

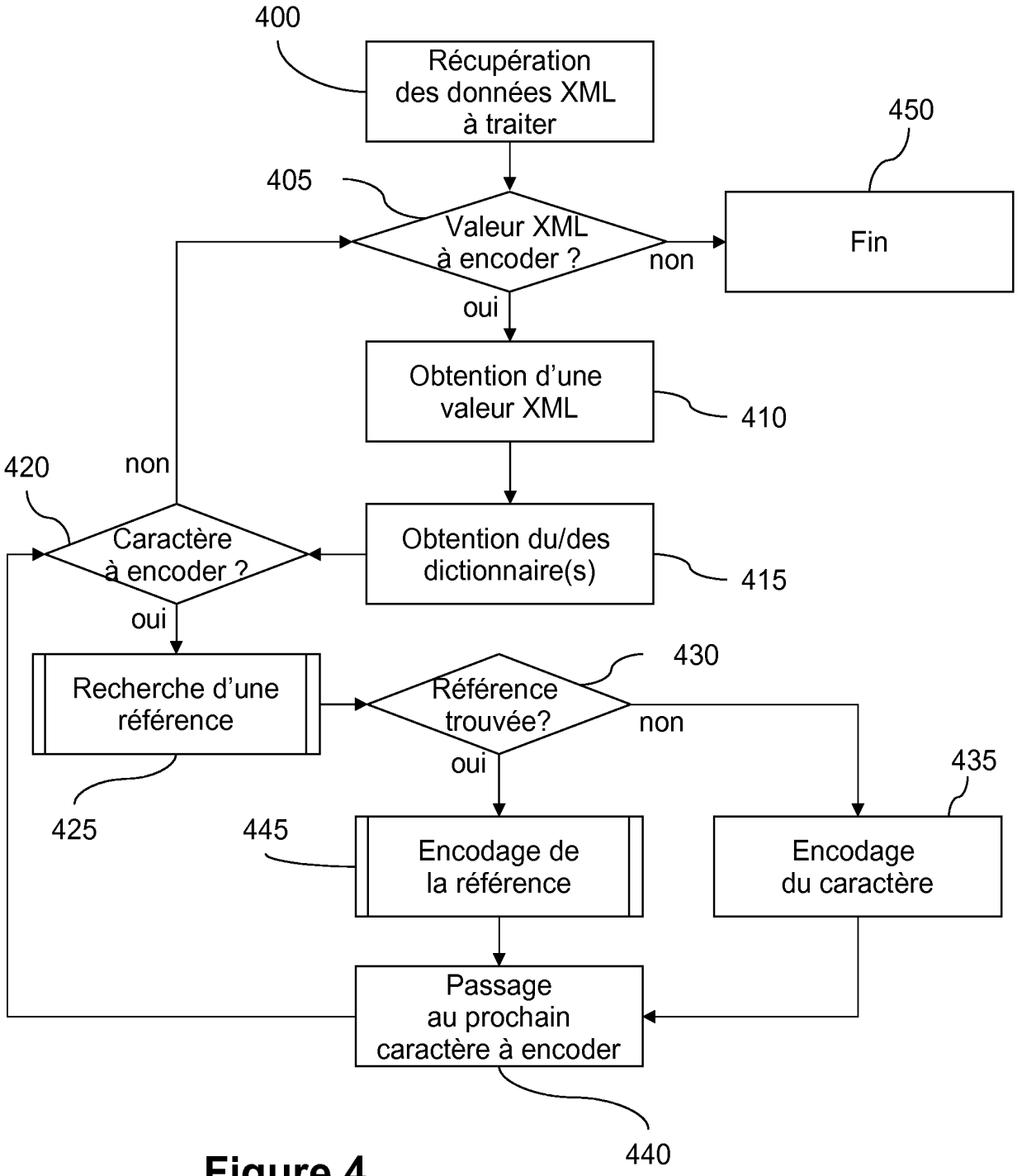

400
Récupération des données XML à traiter
405
Valeur XML à encoder ?
non
450
Fin
oui
Obtention d'une valeur XML
410
420
non
Caractère à encoder ?
Obtention du/des dictionnaire(s)
415
oui
Recherche d'une référence
425
430
Référence trouvée?
non
oui
435
445
Encodage de la référence
Encodage du caractère
Passage au prochain caractère à encoder
440

Figure 4

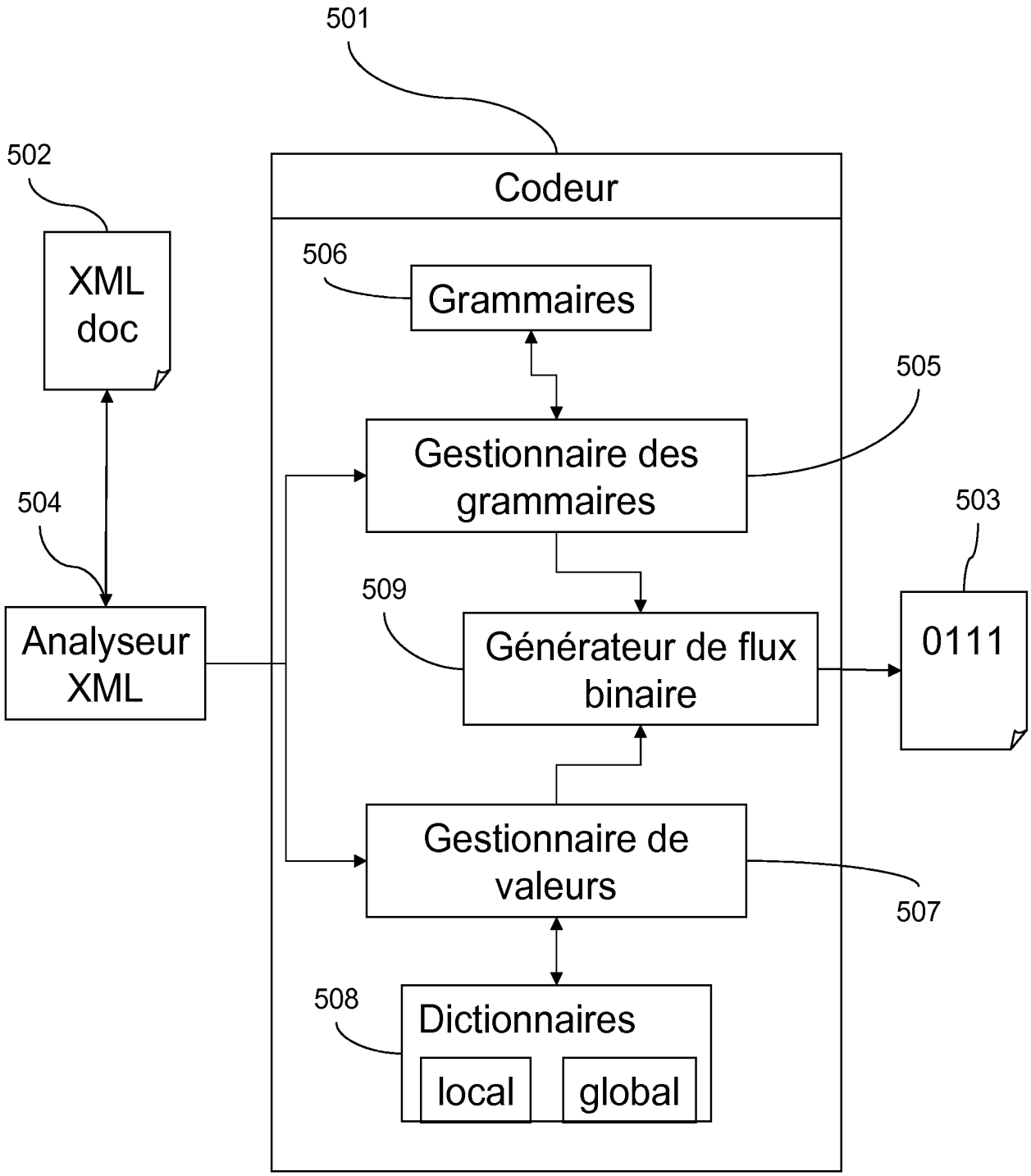

501
Codeur
502
XML doc
506
Grammaires
505
Gestionnaire des grammaires
504
503
509
Analyseur XML
Générateur de flux binaire
0111
Gestionnaire de valeurs
507
508
Dictionnaires
local
global

Figure 5

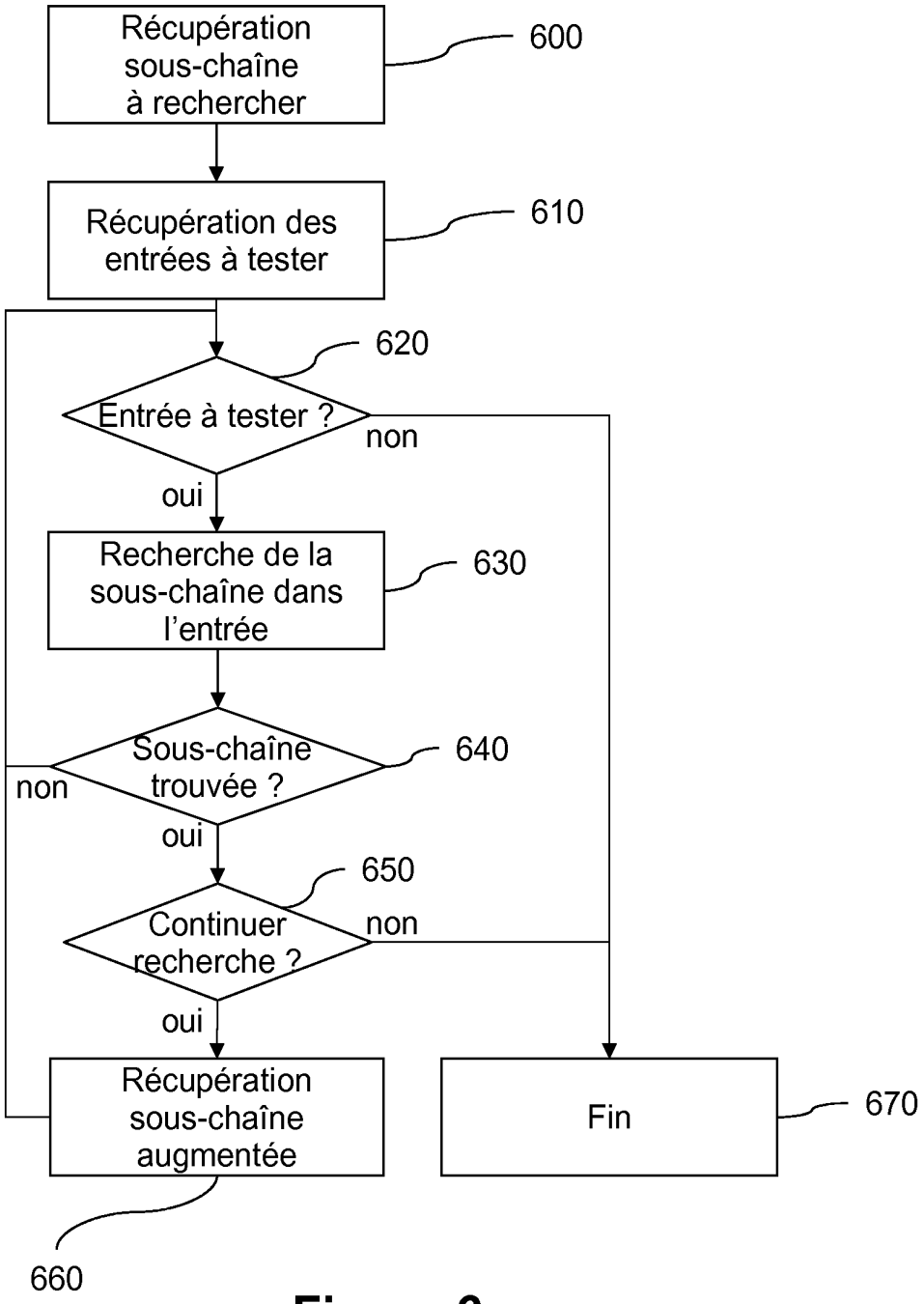

Récupération sous-chaîne à rechercher
600
Récupération des entrées à tester
610
620
Entrée à tester ?
non
oui
Recherche de la sous-chaîne dans l'entrée
630
Sous-chaîne trouvée ?
640
non
oui
650
Continuer recherche ?
non
oui
Récupération sous-chaîne augmentée
Fin
670
660

Figure 6

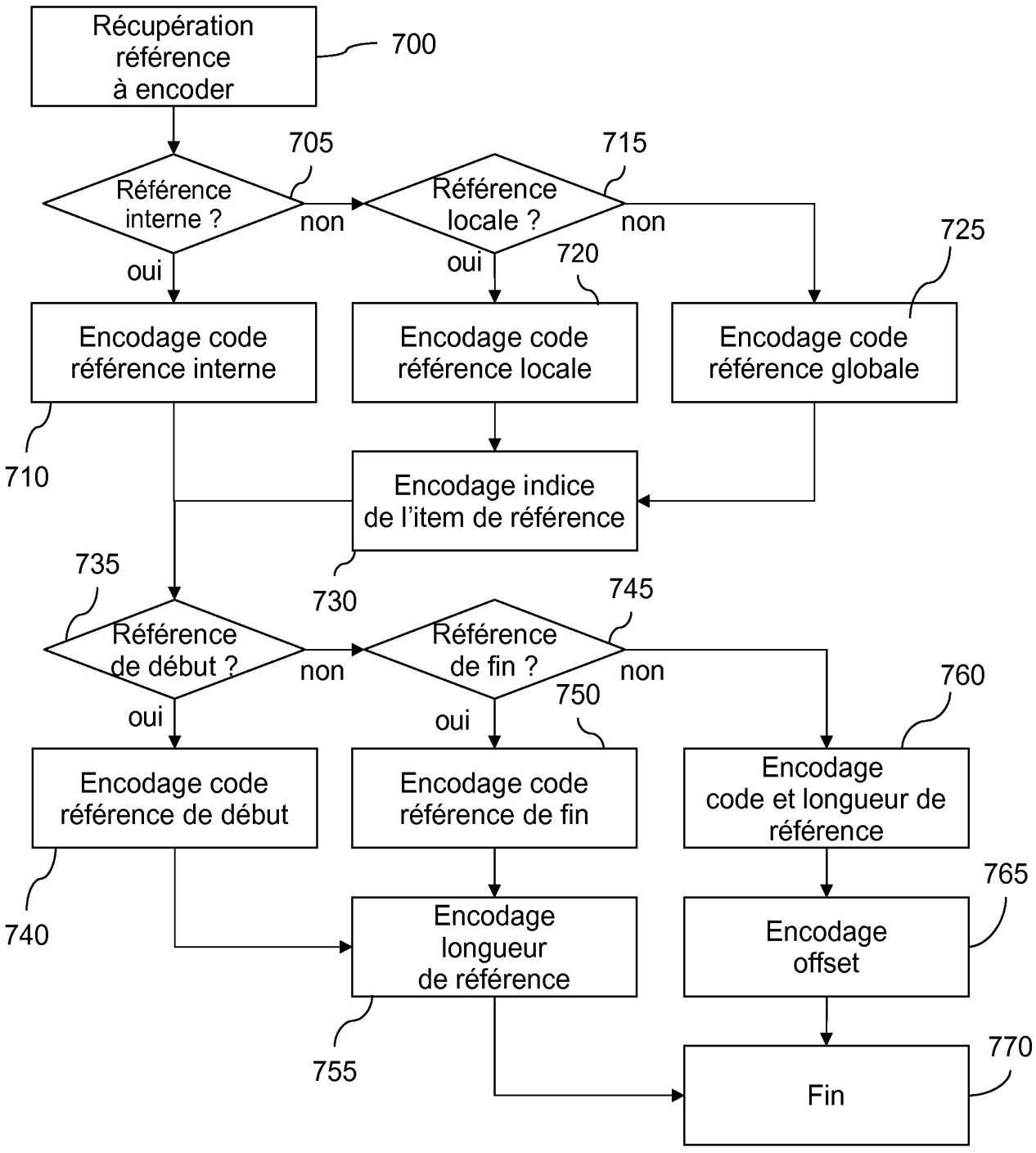

Récupération référence à encoder
700
Référence interne ?
705
non
oui
Référence locale ?
715
non
oui
720
725
Encodage code référence interne
Encodage code référence locale
Encodage code référence globale
710
Encodage indice de l'item de référence
730
735
Référence de début ?
non
oui
Référence de fin ?
745
non
oui
750
760
Encodage code référence de début
Encodage code référence de fin
Encodage code et longueur de référence
740
Encodage longueur de référence
765
Encodage offset
755
770
Fin

Figure 7

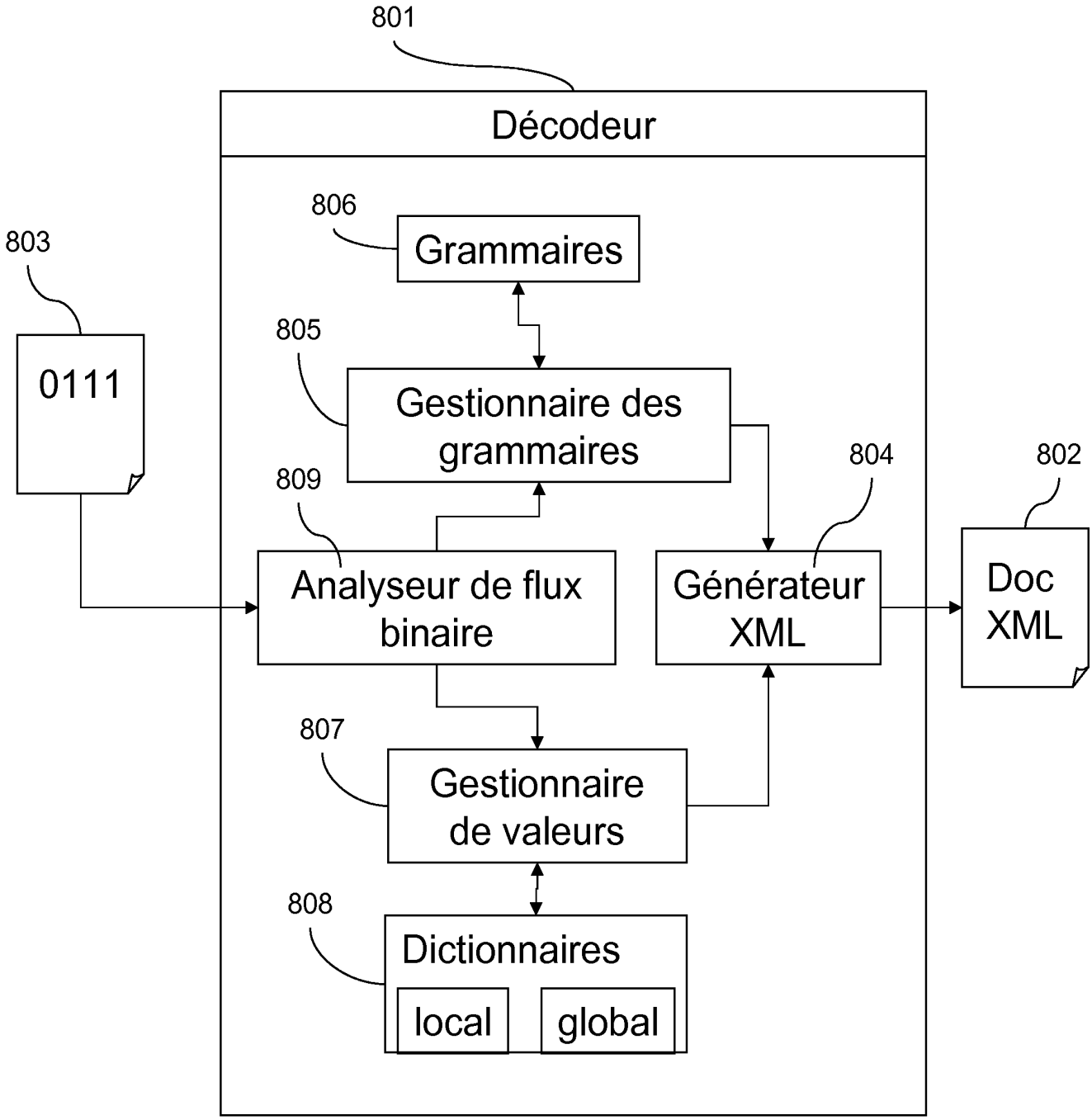

801
Décodeur
806
Grammaires
803
0111
805
Gestionnaire des grammaires
804
802
809
Analyseur de flux binaire
Générateur XML
Doc XML
807
Gestionnaire de valeurs
808
Dictionnaires
local
global

Figure 8

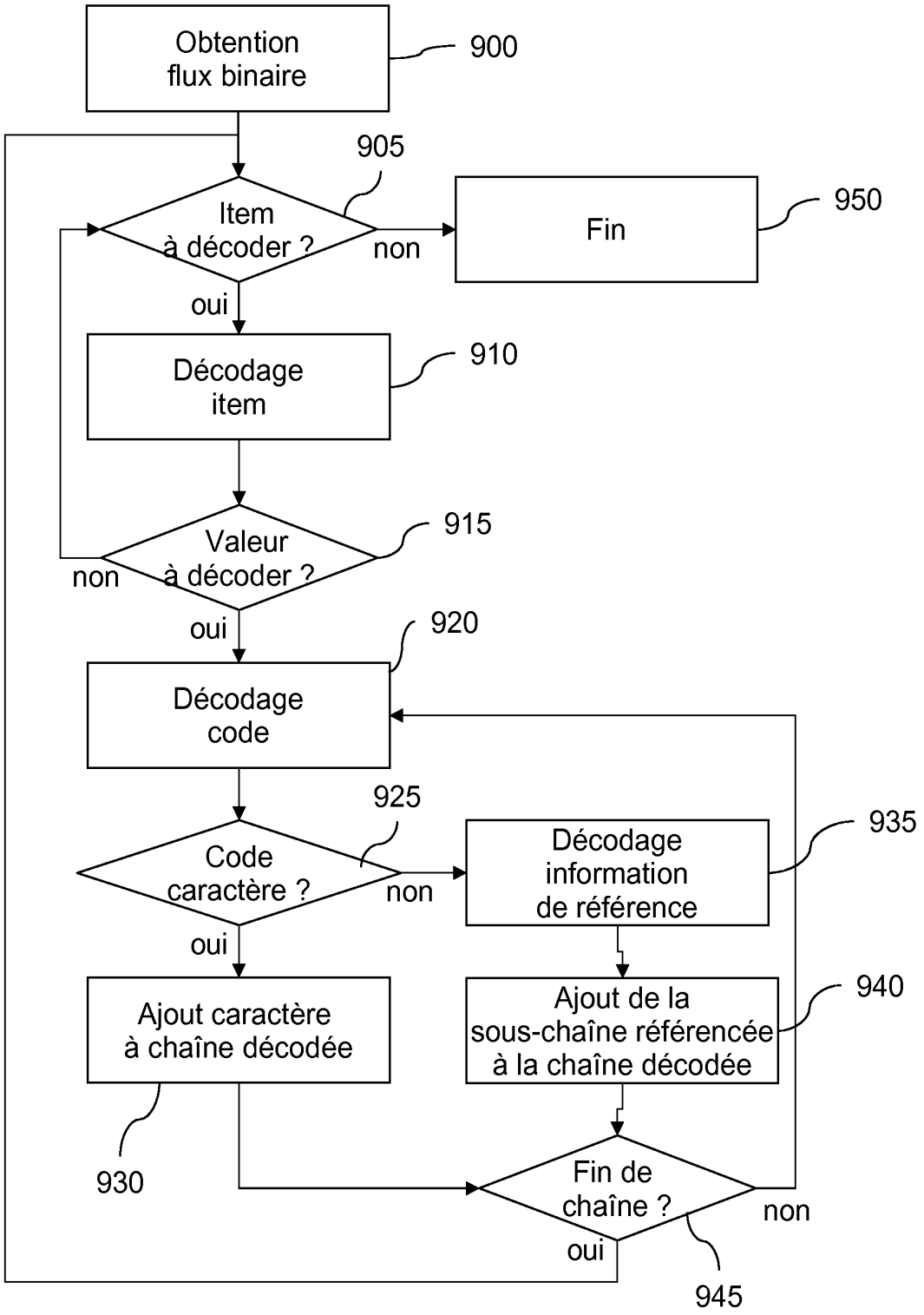

Obtention flux binaire
900
905
Item à décoder ?
non
Fin
950
oui
Décodage item
910
Valeur à décoder ?
915
non
oui
920
Décodage code
925
Code caractère ?
non
Décodage information de référence
935
oui
Ajout caractère à chaîne décodée
Ajout de la sous-chaîne référencée à la chaîne décodée
940
930
Fin de chaîne ?
non
oui
945

Figure 9

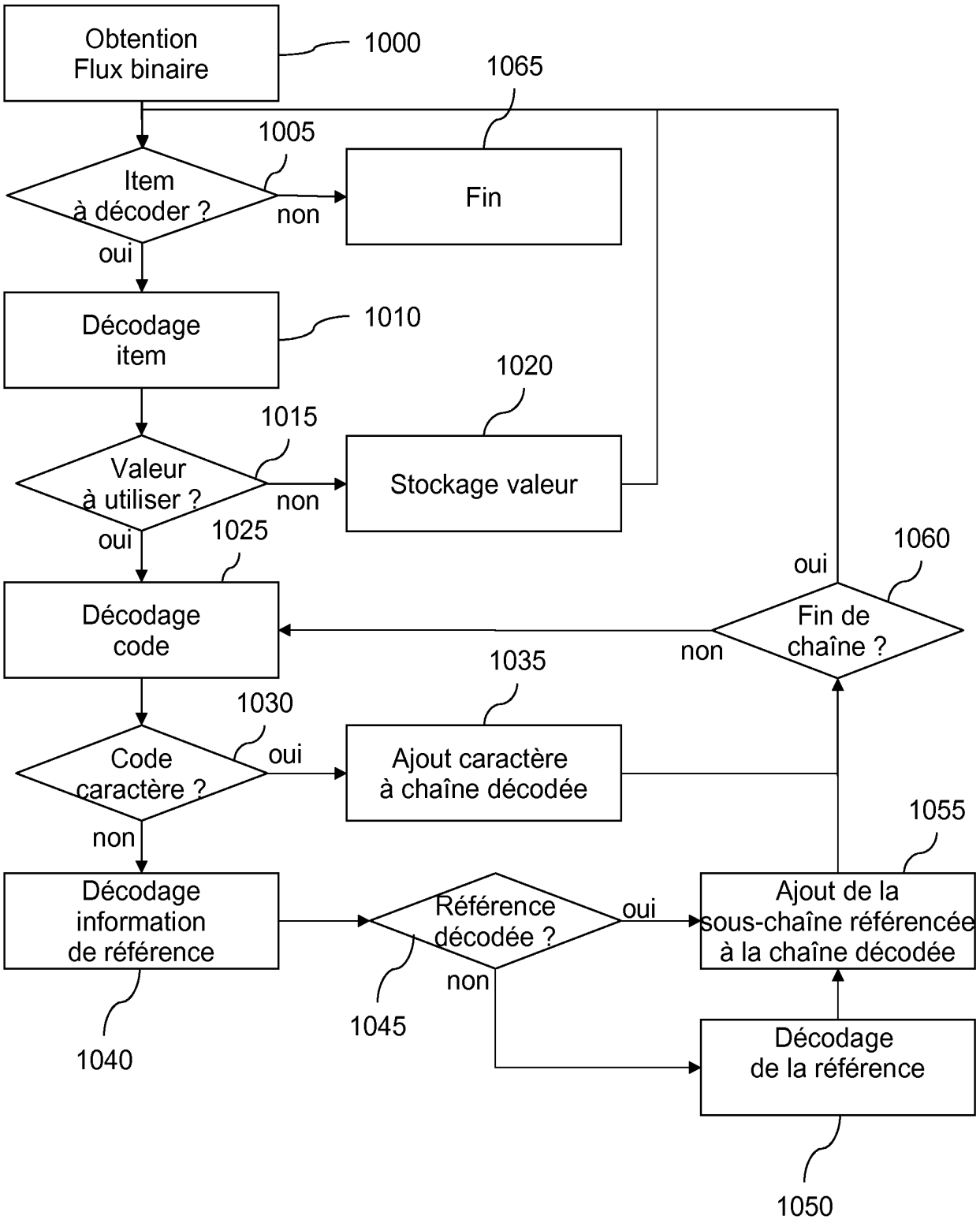

Obtention
Flux binaire
1000
1065
1005
Item
à décoder ?
non
Fin
oui
Décodage
item
1010
1020
1015
Valeur
à utiliser ?
non
Stockage valeur
oui
1025
1060
oui
Décodage
code
Fin de
chaîne ?
non
1035
1030
Code
caractère ?
oui
Ajout caractère
à chaîne décodée
non
1055
Décodage
information
de référence
Référence
décodée ?
oui
Ajout de la
sous-chaîne référencée
à la chaîne décodée
non
1045
1040
Décodage
de la référence
1050

Figure 10

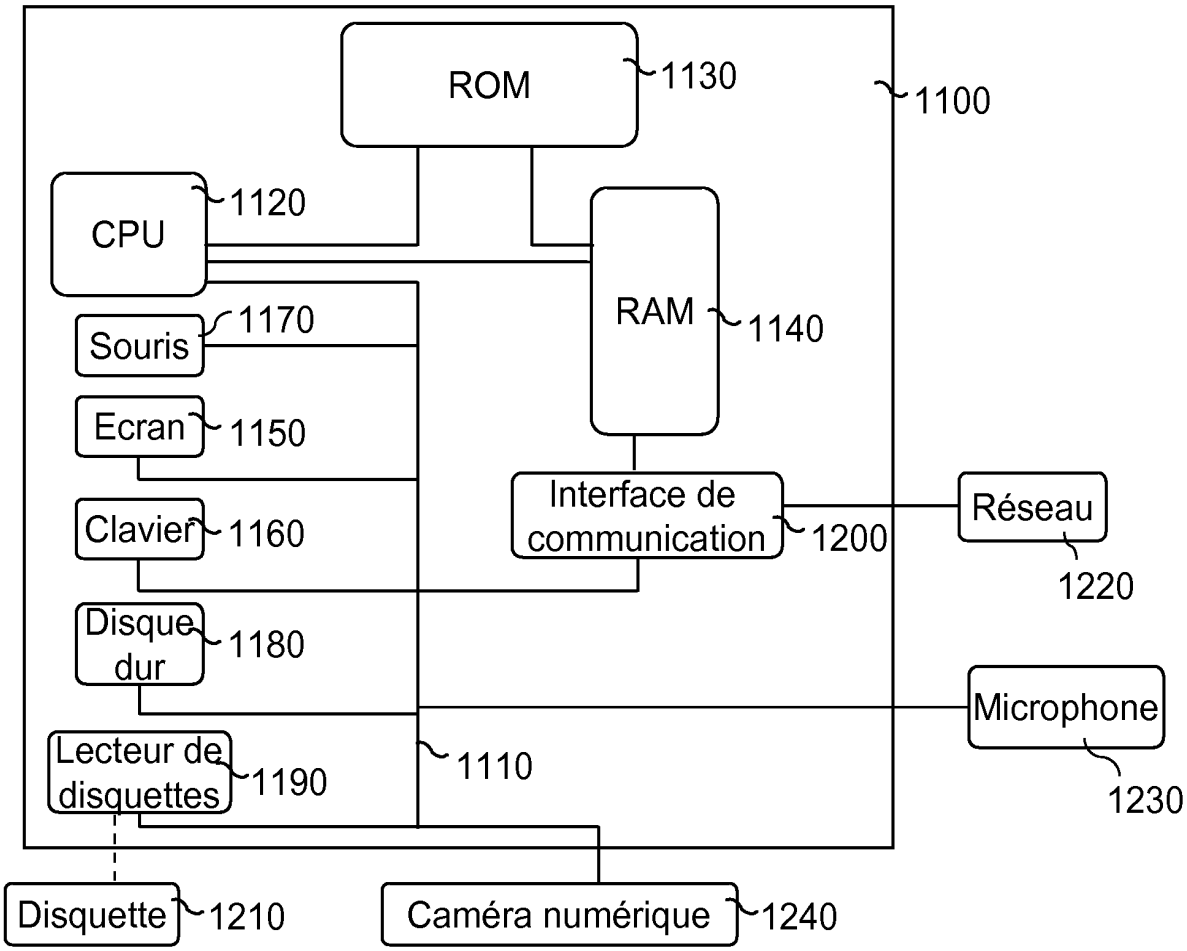

ROM
1130
1100
CPU
1120
RAM
1140
Souris
1170
Ecran
1150
Clavier
1160
Interface de communication
1200
Réseau
1220
Disque dur
1180
Microphone
1230
Lecteur de disquettes
1190
1110
Disquette
1210
Caméra numérique
1240

Figure 11

RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 710907
FR 0854788

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	ADIEGO J ET AL: "Lempel-Ziv compression of highly structured documents" JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE ANDTECHNOLOGY, WILEY & SONS, NEW YORK, NY, US, vol. 58, no. 4, 25 janvier 2007 (2007-01-25), pages 461-478, XP002455183 ISSN: 1532-2882 * le document en entier *	1-25	G06F17/22
X	ADIEGO J ET AL: "Lempel-Ziv compression of structured text" DATA COMPRESSION CONFERENCE, 2004. PROCEEDINGS. DCC 2004 SNOWBIRD, UT, USA MARCH 23-25, 2004, PISCATAWAY, NJ, USA,IEEE, 23 mars 2004 (2004-03-23), pages 112-121, XP010692181 ISBN: 978-0-7695-2082-7 * le document en entier *	1-25	
			DOMAINES TECHNIQUES RECHERCHÉS (IPC)
			H03M

Date d'achèvement de la recherche	Examinateur
16 décembre 2008	Van Staveren, Martin

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)