Groupe de travail Réseau
Request for Comments : 4287
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

M. Nottingham, éditeur
R. Sayre, éditeur
janvier 2005

Format de publication simultanée Atom

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet rur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document spécifie Atom, un format de publication de contenus et de métadonnées de la Toile fondé sur XML.

Table des Matières

1. Introduction

Atom est un format de document fondé sur XML qui décrit des listes d'informations liées appelées des "flux". Les flux sont composés d'un certain nombre d'éléments, appelés des "entrées", dont chacune à un ensemble extensible de métadonnées rattaché. Par exemple, chaque entrée a un titre.

Le principal cas d'utilisation que vise Atom est la publication simultanée de contenus de la Toile comme des blogues et des titres de nouvelles sur des sites de la Toile ainsi que directement aux agents d'utilisateur.

1.1 Exemples

Un bref document de flux Atom d'une seule entrée :

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Exemple de flux</title>
  <link href="http://exemple.org/"/>
  <updated>2003-12-13T18:30:02Z</updated>
  <author>
    <name>John Doe</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>

  <entry>
    <title>Robots Run Amok alimentés par Atom</title>
    <link href="http://exemple.org/2003/12/13/atom03"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2003-12-13T18:30:02Z</updated>
    <summary>Du texte.</summary>
  </entry>

</feed>
```

Document plus développé de flux Atom d'une seule entrée :

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">plongée dans le balisage</title>
  <subtitle type="html">
    Un &lt;em&gt;lot&lt;/em&gt; de l'effort a abouti à le rendre sans effort
  </subtitle>
  <updated>2005-07-31T12:29:29Z</updated>
  <id>tag:exemple.org,2003:3</id>
  <link rel="alternate" type="text/html"
   hreflang="en" href="http://exemple.org/"/>
  <link rel="self" type="application/atom+xml"
   href="http://exemple.org/feed.atom"/>
  <rights>Copyright (c) 2003, Mark Pilgrim</rights>
  <generator uri="http://www.exemple.com/" version="1.0">
    exemple de boîte à outils
  </generator>
  <entry>
    <title>Instantané du projet Atom 07</title>
    <link rel="alternate" type="text/html"
     href="http://exemple.org/2005/04/02/atom"/>
    <link rel="enclosure" type="audio/mpeg" length="1337"
     href="http://exemple.org/audio/ph34r_my_podcast.mp3"/>
    <id>tag:exemple.org,2003:3.2397</id>
```

```
   <updated>2005-07-31T12:29:29Z</updated>
   <published>2003-12-13T08:29:29-04:00</published>
   <author>
    <name>Mark Pilgrim</name>
    <uri>http://exemple.org/</uri>
    <email>f8dy@exemple.com</email>
   </author>
   <contributor>
    <name>Sam Ruby</name>
   </contributor>
   <contributor>
    <name>Joe Gregorio</name>
   </contributor>
   <content type="xhtml" xml:lang="en"
    xml:base="http://diveintomark.org/">
    <div xmlns="http://www.w3.org/1999/xhtml">
     <p><i>[Update: Le projet Atom est terminé.]</i></p>
    </div>
   </content>
  </entry>
</feed>
```

1.2 Espace de noms et version

L'URI de l'espace de noms XML [W3C.REC-xml-names-19990114] pour le format de données XML décrit dans la présente spécification est :

http://www.w3.org/2005/Atom

Pour abréger, ce format de données peut être appelé "Atom 1.0". La présente spécification utilise "Atom" en interne.

1.3. Conventions de notation

La présente spécification décrit la conformité avec deux termes : documents de flux Atom et documents d'entrée Atom. Elle pose de plus des exigences sur les processeurs Atom.

La présente spécification utilise le préfixe d'espace de noms "atom:" pour l'URI d'espace de noms identifié au paragraphe 1.2. Noter que le choix du préfixe d'espace de noms est arbitraire et n'a pas de signification sémantique.

Atom est spécifié en utilisant des termes de l'Infoset XML [W3C.REC-xml-infoset-20040204]. Cependant, la présente spécification utilise un abrégé pour deux termes courants : la locution "élément d'information" est omise quand on désigne des éléments "informations d'élément" et "informations d'attribut". Donc, quand la présente spécification utilise le terme "élément", elle se réfère à un élément "Informations d'élément" dans les termes de Infoset. De même, quand elle utilise le terme "attribut", elle se réfère à un élément "informations d'atttribut".

Certains paragraphes de la présente spécification sont illustrés avec des fragments d'un schéma compact non normatif de RELAX NG [RELAX-NG]. Cependant, le texte de cette spécification fournit la définition de la conformité. Un schéma complet est donné à l'Appendice B.

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

2. Documents Atom

La présente spécification décrit deux sortes de documents Atom : les documents de flux Atom et les documents d'entrée Atom.

Un document de flux Atom est une représentation d'un flux Atom, incluant des métadonnées sur le flux, et certaines ou toutes les entrées qui y sont associées. Sa racine est l'élément atom:feed.

Un document d'entrée Atom représente exactement une entrée Atom, en dehors du contexte d'un flux Atom. Sa racine est l'élément atom:entry.

```
espace de noms atom = "http://www.w3.org/2005/Atom"
début = atomFeed | atomEntry
```

Les deux sortes de documents Atom sont spécifiées dans les termes de l'ensemble d'informations XML, abrégé en XML 1.0 [W3C.REC-xml-20040204] et identifié avec le type de support "application/atom+xml". Les documents Atom DOIVENT être en XML bien formé. La présente spécification ne définit pas de DTD pour les documents Atom, et donc, n'exige pas qu'ils soient valides (au sens utilisé par XML).

Atom permet l'utilisation des IRI [RFC3987]. Chaque URI [RFC3986] est aussi un IRI, de sorte qu'un URI peut être utilisé partout où un IRI est désigné. Deux points particuliers sont à considérer : (1) quand un IRI qui n'est pas aussi un URI est donné pour déréférencement, il DOIT être transposé en un URI en utilisant les étapes du paragraphe 3.1 de la [RFC3987] et (2) quand un IRI sert de valeur d'un atom:id, il NE DOIT PAS être transposé ainsi, afin que la comparaison fonctionne comme décrit au paragraphe 4.2.6.1.

Tout élément défini par la présente spécification PEUT avoir un attribut xml:base [W3C.REC-xmlbase-20010627]. Quand xml:base est utilisé dans un document Atom, il sert à la fonction décrite au paragraphe 5.1.1 de la [RFC3986], établissant l'URI (ou IRI) de base pour résoudre toutes les références qui s'y rapportent trouvées dans la portée effective de l'attribut xml:base.

Tout élément défini par la présente spécification PEUT avoir un attribut xml:lang, dont le contenu indique le langage naturel pour l'élément et ses descendants. Le contexte de langage n'est significatif que pour les éléments et attributs déclarés être "sensible au langage" par la présente spécification. Les exigences relatives au contenu et à l'interprétation de xml:lang sont spécifiées dans XML 1.0 [W3C.REC-xml-20040204], paragraphe 2.12.

```
atomCommonAttributes =
  attribute xml:base { atomUri }?,
  attribute xml:lang { atomLanguageTag }?,
  undefinedAttribute*
```

Atom est un format extensible. Voir à la Section 6 de ce document une description complète de la façon dont les documents Atom peut être étendus.

Les processeurs Atom PEUVENT garder l'état provenant des documents de flux Atom et le combiner avec d'autres documents de flux Atom, afin de faciliter une vue d'ensemble du contenu d'un flux. La manière dont les documents de flux Atom sont combinés afin de reconstruire un flux (par exemple, mise à jour des entrées et métadonnées, traiter les entrées manquantes) sort du domaine d'application de la présente spécification.

3. Constructions Atom courantes

Beaucoup d'éléments de Atom partagent quelques structures communes. Cette Section définit ces structures et leurs exigences pour une référence pratique par les définitions d'éléments appropriées.

Quand un élément est identifié comme étant d'une sorte de construction particulière, il hérite des exigences correspondantes provenant de cette définition de construction dans cette section.

Noter qu'il NE DOIT PAS y avoir d'espace blanche dans une construction Date ni dans aucun IRI. Certaines mises en œuvre émettant du XML insèrent de façon erronée des espaces blanches autour des valeurs par défaut, et de telles mises en œuvre vont émettre des documents Atom invalides.

3.1 Constructions Text

Une construction Text contient du texte lisible par l'hommme, généralement en petites quantités. Le contenu des

constructions Text est sensible au langage.

```
atomPlainTextConstruct =
  atomCommonAttributes,
  attribute type { "text" | "html" }?,
  text

atomXHTMLTextConstruct =
  atomCommonAttributes,
  attribute type { "xhtml" },
  xhtmlDiv

atomTextConstruct = atomPlainTextConstruct | atomXHTMLTextConstruct
```

3.1.1 Attribut "type"

Les constructions Text PEUVENT avoir un attribut "type". Quand il est présent, sa valeur DOIT être une de "text", "html", ou "xhtml". Si l'attribut "type" n'est pas fourni, les processeurs Atom DOIVENT se comporter comme si il était présent avec la valeur de "text". À la différence de l'élément atom:content défini au paragraphe 4.1.3, les types de supports MIME [RFC4288] NE DOIVENT PAS être utilisés comme valeurs pour l'attribut "type" sur des constructions Text.

3.1.1.1 Texte

Exemple de atom:title avec contenu text :

```
...
<title type="text">
  Less: &lt;
</title>
...
```

Si la valeur est "text", le contenu de la construction Text NE DOIT PAS contenir d'éléments fils. Un tel texte est destiné à être présenté aux humains sous une forme lisible. Donc, les processeurs Atom PEUVENT introduire des espaces blanches (incluant des coupures de ligne) et afficher le texte en utilisant des techniques typographiques comme la justification et les fontes proportionnelles.

3.1.1.2 HTML

Exemple de atom:title avec contenu HTML :

```
...
<title type="html">
  Less: &lt;em> &amp;lt; &lt;/em>
</title>
...
```

Si la valeur de "type" est "html", le contenu de la construction Text NE DOIT PAS contenir d'éléments fils et DEVRAIT être convenable pour un traitement HTML [HTML]. Tout balisage en son sein DOIT être échappé ; par exemple, "<br>" pour "<br>". Le balisage HTML en son sein DEVRAIT être tel qu'il puisse apparaître de façon valide directement au sein d'un élément HTML <DIV>, après la suppression de l'échappement. Les processeurs Atom qui affichent de tels contenus PEUVENT utiliser ce balisage pour aider à l'affichage.

3.1.1.3 XHTML

Exemple de atom:title avec contenu XHTML :

```
...
<title type="xhtml" xmlns:xhtml="http://www.w3.org/1999/xhtml">
```

```
  <xhtml:div>
    Less: <xhtml:em> &lt; </xhtml:em>
  </xhtml:div>
</title>
...
```

Si la valeur de "type" est "xhtml", le contenu de la construction Text DOIT être un seul élément div XHTML [XHTML] et DEVRAIT être convenable pour le traitement comme XHTML. L'élément div XHTML lui-même NE DOIT PAS être considéré comme faisant partie du contenu. Les processeurs Atom qui affichent le contenu PEUVENT utiliser le balisage pour aider à son affichage. Les versions échappées de caractères comme "&" et ">" représentent ces caractères, pas le balisage.

Exemples de contenu XHTML valide :

```
...
<summary type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
    Ceci est du contenu <b>XHTML</b>.
  </div>
</summary>
...

<summary type="xhtml">
  <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">
    Ceci est du contenu <xhtml:b>XHTML</xhtml:b>.
  </xhtml:div>
</summary>
...
```

L'exemple suivant suppose que l'espace de noms XHTML a été limité par le préfixe "xh" plus tôt dans le document:

```
...
<summary type="xhtml">
  <xh:div>
    Ceci est du contenu <xh:b>XHTML</xh:b>.
  </xh:div>
</summary>
...
```

3.2 Constructions Person

Une construction Person est un élément qui décrit une personne, une entreprise , ou entité similaire (ci-après appelée "person"').

```
atomPersonConstruct =
  atomCommonAttributes,
  (element atom:name { text }
   & element atom:uri { atomUri }?
   & element atom:email { atomEmailAddress }?
   & extensionElement*)
```

La présente spécification n'alloue pas de signification à l'ordre d'apparition des éléments fils dans une construction Person. Les constructions Person permettent des éléments de métadonnées d'extension (voir au paragraphe 6.4).

3.2.1 Élément "atom:name"

Le contenu de l'élément "atom:name" porte un nom lisible par l'homme pour la personne. Le contenu de atom:name est sensible au langage. Les constructions Person DOIVENT contenir exactement un élément "atom:name".

3.2.2 Élément "atom:uri"

Le contenu de l'élément "atom:uri" porte un IRI associé à la personne. Les constructions Person PEUVENT contenir un élément atom:uri, mais NE DOIVENT PAS en contenir plus d'un. Le contenu de atom:uri dans une construction Person DOIT être une référence d'IRI [RFC3987].

3.2.3 Élément "atom:email"

Le contenu de l'élément "atom:email" porte une adresse de messagerie électronique associée à la personne. Les constructions Person PEUVENT contenir un élément atom:email, mais NE DOIVENT PAS en contenir plus d'un. Son contenu DOIT être conforme à la production "addr-spec" de la [RFC2822].

3.3 Constructions Date

Une construction Date est un élément dont le contenu DOIT se conformer à la production "date-time" de la [RFC3339]. De plus, un caractère majuscule "T" DOIT être utilisé pour séparer la date et l'heure, et un caractère majuscule "Z" DOIT être présent en l'absence d'un décalage numérique de zone horaire.

```
atomDateConstruct =
  atomCommonAttributes,
  xsd:dateTime
```

Ces valeurs de date se trouvent être compatibles avec les spécifications suivantes : [ISO.8601.1988], [W3C.NOTE-datetime-19980827], et [W3C.REC-xmlschema-2-20041028].

Exemples de constructions Date :

```
<updated>2003-12-13T18:30:02Z</updated>
<updated>2003-12-13T18:30:02.25Z</updated>
<updated>2003-12-13T18:30:02+01:00</updated>
<updated>2003-12-13T18:30:02.25+01:00</updated>
```

Les valeurs de Date DEVRAIENT être aussi précises que possible. Par exemple, il serait généralement inapproprié pour un système de publication d'appliquer le même horodatage à plusieurs entrées qui ont été publiées durant le cours d'un seul jour.

4. Définitions d'éléments Atom

4.1 Éléments conteneurs

4.1.1 Élément "atom:feed"

L'élement "atom:feed" est l'élément de document (c'est-à-dire, de niveau supérieur) d'un document de flux Atom, agissant comme un conteneur pour les métadonnées et données associées au flux. Ses éléments fils consistent en éléments de métadonnées suivis par zéro, un ou plusieurs éléments fils atom:entry.

```
atomFeed =
  element atom:feed {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContributor*
     & atomGenerator?
     & atomIcon?
     & atomId
     & atomLink*
     & atomLogo?
     & atomRights?
     & atomSubtitle?
```

```
      & atomTitle
      & atomUpdated
      & extensionElement*),
     atomEntry*
  }
```

La présente spécification n'accorde aucune signification à l'ordre des éléments atom:entry dans le flux.

Les éléments fils suivants sont définis par cette spécification (noter que la présence de certains de ces éléments est exigée) :

- Les éléments atom:feed DOIVENT contenir un ou plusieurs éléments atom:author, sauf si tous les éléments fils atom:entry de l'élément atom:feed contiennent au moins un élément atom:author.
- Les éléments atom:feed PEUVENT contenir un nombre quelconque d'éléments atom:category.
- Les éléments atom:feed PEUVENT contenir un nombre quelconque d'éléments atom:contributor.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:generator.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:icon.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:logo.
- Les éléments atom:feed DOIVENT contenir exactement un élément atom:id.
- Les éléments atom:feed DEVRAIENT contenir un élément atom:link avec une valeur d'attribut de relation de "self". C'est l'URI préféré pour restituer les documents de flux Atom qui représentent ce flux Atom.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:link avec une valeur d'attribut de relation de "alternate" qui ait la même combinaison de valeurs d'atttribut de type et hreflang.
- Les éléments atom:feed PEUVENT contenir des éléments atom:link supplémentaires au-delà de ceux décrits ci-dessus.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:rights.
- Les éléments atom:feed NE DOIVENT PAS contenir plus d'un élément atom:subtitle.
- Les éléments atom:feed DOIVENT contenir exactement un élément atom:title.
- Les éléments atom:feed DOIVENT contenir exactement un élément atom:updated.

Si plusieurs éléments atom:entry avec la même valeur de atom:id apparaissent dans un document de flux Atom, ils représentent la même entrée. Leurs horodatages atom:updated DEVRAIENT être différents. Si un document de flux Atom contient plusieurs entrées avec le même atom:id, les processeurs Atom PEUVENT choisir de tous les afficher ou de n'en afficher qu'un sous ensemble. Un comportement normal serait de n'afficher que l'entrée qui a le dernier horodatage atom:updated.

4.1.1.1 Fourniture de contenu textuel

L'expérience enseigne que les flux qui contiennent du contenu de texte sont en général plus utiles que ceux qui n'en ont pas. Certaines applications (un exemple est celui des indexations en pleine page) exigent une quantité minimum de texte ou (X)HTML pour fonctionner de façon fiable et prévisible. Les producteurs de flux devraient être conscients de ces problèmes. Il est conseillé que chaque élément atom:entry contienne un élément atom:title non vide, un élément atom:content non vide quand cet élément est présent, et un élément atom:summary non vide quand l'entrée ne contient pas d'élément atom:content. Cependant, l'absence de atom:summary n'est pas une erreur, et les processeurs Atom NE DOIVENT PAS échouer à fonctionner correctement par suite d'une telle absence.

4.1.2 Élément "atom:entry"

L'élément "atom:entry" représente une entrée individuelle, agissant comme un conteneur pour les métadonnées et données associées à l'entrée. Cet élément peut apparaître comme fils de l'élément atom:feed, ou comme l'élément de document (c'est-à-dire, de niveau supérieur) d'un document d'entrée Atom autonome.

```
atomEntry =
  element atom:entry {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContent?
     & atomContributor*
     & atomId
     & atomLink*
     & atomPublished?
```

```
      & atomRights?
      & atomSource?
      & atomSummary?
      & atomTitle
      & atomUpdated
      & extensionElement*)
  }
```

La présente spécification n'accorde aucune signification à l'ordre d'apparition des éléments fils de atom:entry.

Les éléments fils suivants sont définis par la présente spécification (noter que la présence de certains de ces éléments est exigée) :

- Les éléments atom:entry DOIVENT contenir un ou plusieurs éléments atom:author, sauf si la atom:entry contient un élément atom:source qui contient un élément atom:author ou si, dans un document de flux Atom, l'élément atom:feed contient lui-même un élément atom:author.
- Les éléments atom:entry PEUVENT contenir un nombre quelconque d'éléments atom:category.
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:content.
- Les éléments atom:entry PEUVENT contenir un nombre quelconque d'éléments atom:contributor.
- Les éléments atom:entry DOIVENT contenir exactement un élément atom:id.
- Les éléments atom:entry qui ne contiennent pas d'élément fils atom:content DOIVENT contenir au moins un élément atom:link avec une valeur d'attribut "rel" de "alternate".
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:link avec une valeur d'attribut "rel" de "alternate" qui ait la même combinaison de valeurs d'attributs "type" et "hreflang".
- Les éléments atom:entry PEUVENT contenir des éléments atom:link supplémentaires au delà de ceux décrits ci-dessus.
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:published.
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:rights.
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:source.
- Les éléments atom:entry DOIVENT contenir un élément atom:summary dans l'un des cas suivants :
  - la atom:entry contient un atom:content qui a un attribut "src" (et est donc vide) ;
  - la atom.entry contient un contenu codé en Base64 ; c'est-à-dire, l'attribut "type" du atom:content est un type de support MIME [RFC4288], mais n'est pas un type de support XML [RFC3023], ne commence pas par "text/", et ne se finit pas par "/xml" ou "+xml".
- Les éléments atom:entry NE DOIVENT PAS contenir plus d'un élément atom:summary.
- Les éléments atom:entry DOIVENT contenir exactement un élément atom:title.
- Les éléments atom:entry DOIVENT contenir exactement un élément atom:updated.

4.1.3 Élément "atom:content"

L'élément "atom:content" soit contient, soit relie au contenu de l'entrée. Le contenu de atom:content est sensible au langage.

```
atomInlineTextContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { "text" | "html" }?,
    (text)*
  }

atomInlineXHTMLContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { "xhtml" },
    xhtmlDiv
  }

atomInlineOtherContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    (text|anyElement)*
```

```
  }

atomOutOfLineContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    attribute src { atomUri },
    empty
  }

atomContent = atomInlineTextContent
 | atomInlineXHTMLContent
 | atomInlineOtherContent
 | atomOutOfLineContent
```

4.1.3.1 Attribut "type"

Sur l'élément atom:content, la valeur de l'attribut "type" PEUT être "text", "html", ou "xhtml". De plus, il DOIT se conformer à la syntaxe d'un type de support MIME, mais NE DOIT PAS être un type composite (voir au paragraphe 4.2.6 de la [RFC4288]). Si ni l'attribut de type ni l'attribut src ne sont fournis, les processeurs Atom DOIVENT se comporter comme si l'attribut de type était présent avec une valeur de "text".

4.1.3.2 Attribut "src"

atom:content PEUT avoir un attribut "src", dont la valeur DOIT être une référence d'IRI [RFC3987]. Si l'attribut "src" est présent, atom:content DOIT être vide. Les processeurs Atom PEUVENT utiliser l'IRI pour restituer le contenu et PEUVENT choisir d'ignorer le contenu distant ou de le présenter d'une manière différente du contenu local.

Si l'attribut "src" est présent, l'attribut "type" DEVRAIT être fourni et DOIT être un type de support MIME [RFC4288], plutôt que "text", "html", ou "xhtml". La valeur est pour avis ; c'est-à-dire que quand l'URI correspondant (transposé d'un IRI, si nécessaire) est déréférencé, si le serveur qui fournit ce contenu fournit aussi un type de support, le type de support fourni par le serveur est d'autorité.

4.1.3.3 Modèle de traitement

Les documents Atom DOIVENT se conformer aux règles suivantes. Les processeurs Atom DOIVENT interpréter atom:content conformément à la première règle applicable.

1. Si la valeur de "type" est "text", le contenu de atom:content NE DOIT PAS contenir d'éléments fils. Un tel texte est destiné à être présenté aux humain de façon lisible. Donc, les processeurs Atom PEUVENT introduire des espaces blanches (y compris des coupures de ligne) et afficher le texte en utilisant des techniques typographiques comme la justification et des fontes proportionnelles.

2. Si la valeur de "type" est "html", le contenu de atom:content NE DOIT PAS contenir d'éléments fils et DEVRAIT être convenable pour un traitement comme HTML [HTML]. Le balisage HTML DOIT être échappé ; par exemple, "<br>" comme "<br>". Le balisage HTML DEVRAIT être tel qu'il puisse apparaître valide directement dans un élément HTML <DIV>. Les processeurs Atom qui affichent le contenu PEUVENT utiliser le balisage pour aider à son affichage.

3. Si la valeur de "type" est "xhtml", le contenu de atom:content DOIT être un seul élément div XHTML [XHTML] et DEVRAIT être convenable pour un traitement comme XHTML. L'élément div XHTML lui-même NE DOIT PAS être considéré comme faisant partie du contenu. Les processeurs Atom qui affichent le contenu PEUVENT utiliser le balisage pour aider à son affichage. Les versions échappées de caractères comme "&" et ">" représentent ces caractères, et non le balisage.

4. Si la valeur de "type" est un type de support XML [RFC3023] ou se termine par "+xml" ou "/xml" (insensible à la casse) le contenu du atom:content PEUT inclure des éléments fils et DEVRAIT être convenable pour un traitement comme le type de support indiqué. Si l'attribut "src" n'est pas fourni, cela va normalement signifier que l'élément "atom:content" va contenir un seul élément fils qui va servir d'élément racine du document XML du type indiqué.

5. Si la valeur de "type" commence par "text/" (insensible à la casse) le contenu du atom:content NE DOIT PAS contenir d'éléments fils.

6. Pour toutes les autres valeurs de "type", le contenu du atom:content DOIT être un codage Base64 valide, comme décrit dans la [RFC3548], section 3. Quand il est décodé, il DEVRAIT être convenable pour le traitement du type de support indiqué. Dans ce cas, les caractères en codage Base64 PEUVENT être précédés et suivis dans l'élément atom:content par une espace blanche, et les lignes séparées par un seul caractère de nouvelle ligne (U+000A).

4.1.3.4 Exemples

XHTML en ligne :

```
...
<content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
    Ceci est un contenu <b>XHTML</b>.
  </div>
</content>
...
<content type="xhtml">
  <xhtml:div xmlns:xhtml="http://www.w3.org/1999/xhtml">
    Ceci est un contenu <xhtml:b>XHTML</xhtml:b>.
  </xhtml:div>
</content>
...
```

L'exemple suivant suppose que l'espace de noms XHTML a été limité au préfixe "xh" précédemment dans le document :

```
...
<content type="xhtml">
  <xh:div>
    Ceci est un contenu <xh:b>XHTML</xh:b>.
  </xh:div>
</content>
...
```

4.2 Éléments de métadonnées

4.2.1 Élément "atom:author"

L'élément "atom:author" est une construction Person qui indique l'auteur de l'entrée ou flux.

```
atomAuthor = élément atom:author { atomPersonConstruct }
```

Si l'élément atom:entry ne contient pas d'élément atom:author, alors l'élément atom:author de l'élement atom:source contenu est considéré comme s'appliquant. Dans un document de flux Atom, les éléments atom:author de l'élément contenant atom:feed sont supposés s'appliquer à l'entrée si il n'y a pas d'élément atom:author dans les localisations décrites ci-dessus.

4.2.2 Élément "atom:category"

L'élément "atom:category" porte des informations sur une catégorie associée à une entrée ou flux. La présente spécification n'accorde pas de signification au contenu (si il en est) de cet élément.

```
atomCategory =
  élément atom:category {
    atomCommonAttributes,
    attribute term { text },
```

```
    attribute scheme { atomUri }?,
    attribute label { text }?,
    undefinedContent
  }
```

4.2.2.1 Attribut "term"

L'attribut "term" est une chaîne qui identifie la catégorie à laquelle appartient l'entrée ou flux. Les éléments de catégorie DOIVENT avoir un attribut "term".

4.2.2.2 Attribut "scheme"

L'attribut "scheme" est un IRI qui identifie un schéma de catégorisation. Les éléments Category PEUVENT avoir un attribut "scheme".

4.2.2.3 Attribut "label"

L'attribut "label" fournit une étiquette lisible par l'humain pour affichage dans les applications d'utilisateur final. Le contenu de l'attribut "label" est sensible au langage. Les entités comme "&" et "<" représentent leurs caractères correspondants ("&" et "<", respectivement) pas le balisage. Les éléments Category PEUVENT avoir un attribut "label".

4.2.3 Élément "atom:contributor"

L'attribut "atom:contributor" est une construction Person qui indique une personne ou autre entité qui a contribué à l'entrée ou flux.

```
atomContributor = élément atom:contributor { atomPersonConstruct }
```

4.2.4 Élément "atom:generator"

Le contenu de l'élément "atom:generator" identifie l'agent utilisé pour générer un flux, pour les besoins de débogage et autres.

```
atomGenerator = élément atom:generator {
  atomCommonAttributes,
  attribute uri { atomUri }?,
  attribute version { text }?,
  text
}
```

Le contenu de cet élément, quand il est présent, DOIT être une chaîne qui est un nom lisible par l'homme pour l'agent générateur. Des entités comme "&" et "<" représentent leurs caractères correspondants ("&" et "<" respectivement) pas le balisage.

L'élément atom:generator PEUT avoir un attribut "uri" dont la valeur DOIT être une référence d'IRI [RFC3987]. Quand il est déréférencé, l'URI résultant (transposé d'un IRI, si nécessaire) DEVRAIT produire une représentation pertinente pour cet agent.

L'élément atom:generator PEUT avoir un attribut "version" qui indique la version de l'agent générateur.

4.2.5 Élément "atom:icon"

Le contenu de l'élément "atom:icon" est une référence d'IRI [RFC3987] qui identifie une image donnant une identification iconique visuelle pour un flux.

```
atomIcon = élément atom:icon {
  atomCommonAttributes,
```

```
  (atomUri)
}
```

L'image DEVRAIT avoir un ratio d'aspect de un (horizontal) à un (vertical) et DEVRAIT convenir pour une présentation en petite taille.

4.2.6 Élément "atom:id"

L'élément "atom:id" porte un identifiant permanent, universellement unique pour une entrée ou flux.

```
atomId = élément atom:id {
  atomCommonAttributes,
  (atomUri)
}
```

Son contenu DOIT être un IRI, comme défini par la [RFC3987]. Noter que la définition de "IRI" exclut les références relatives. Bien que l'IRI puisse utiliser un schéma de déréférence, les processeurs Atom NE DOIVENT PAS supposer qu'il peut être déréférencé.

Quand un document Atom est relocalisé, migré, groupé, republié, exporté, ou importé, le contenu de son élément atom:id NE DOIT PAS changer. Dit autrement, un élément atom:id relève de toutes les instanciations d'une entrée ou d'un flux Atom particulier ; les révisions conservent le même contenu dans leurs éléments atom:id. Il est suggéré que l'élément atom:id soit mémorisé avec la ressource associée.

Le contenu d'un élément atom:id DOIT être créé d'une façon qui assure son unicité.

À cause du risque de confusion entre les IRI qui pourraient être équivalents si ils étaient transposés en URI et déréférencés, la stratégie de normalisation suivante DEVRAIT être appliquée lors de la génération des éléments atom:id :

- Fournir le schéma en caractères minuscules.
- Fournir l'hôte, si il en est, en caractères minuscules.
- N'effectuer de codage en pourcentage que si c'est essentiel.
- Utiliser les caractères majuscules A à F avec le codage en pourcentage.
- Empêcher les segments points d'apparaître dans les chemins.
- Pour les schémas qui définissent une autorité par défaut, utiliser une autorité vide si la valeur par défaut est désirée.
- Pour les schémas qui définissent un chemin vide comme équivalent à un chemin de "/", utiliser "/".
- Pour les schémas qui définissent un accès, utiliser un accès vide si l'accès par défaut est désiré.
- Préserver les identifiants et interrogations de fragment vide.
- S'assurer que tous les composants de l'IRI ont leurs caractères normalisés de façon appropriée, par exemple, en utilisant NFC ou NFKC.

4.2.6.1 Comparaison de atom:id

Les instances d'éléments atom:id peuvent être comparées pour déterminer si une entrée ou flux est la même qu'une vue précédemment. Les processeurs DOIVENT comparer les éléments atom:id caractère par caractère (non sensible à la casse). Les opérations de comparaison DOIT se fonder seulement sur les chaînes de caractères de l'IRI et NE DOIVENT PAS s'appuyer sur le déréférencement des IRI ou des URI qui en sont transposés.

Par suite, deux IRI qui se résolvent en la même ressource mais ne sont pas identiques caractère par caractère vont être considérés comme différents pour les besoins de la comparaison d'identifiants.

Par exemple, voici quatre identifiants distincts, en dépit du fait qu'il ne diffèrent que par la casse :

```
http://www.exemple.org/thing
http://www.exemple.org/Thing
http://www.exemple.org/thing
HTTP://www.exemple.org/thing
```

De même, voici trois identifiants distincts, parce que l'échappement en pourcentage d'IRI est significatif pour les besoins de la comparaison :

```
http://www.exemple.com/~bob
```

http://www.exemple.com/%7ebob
http://www.exemple.com/%7Ebob

4.2.7 Élément "atom:link"

L'élément "atom:link" définit une référence d'une entrée ou d'un flux à une ressource de la Toile. La présente spécification ne donne aucune signification au contenu (si il en est un) de cet élément.

```
atomLink =
  élément atom:link {
    atomCommonAttributes,
    attribute href { atomUri },
    attribute rel { atomNCName | atomUri }?,
    attribute type { atomMediaType }?,
    attribute hreflang { atomLanguageTag }?,
    attribute title { text }?,
    attribute length { text }?,
    undefinedContent
  }
```

4.2.7.1 Attribut "href"

L'attribut "href" contient l'IRI de la liaison. Les éléments atom:link DOIVENT avoir un attribut href, dont la valeur DOIT être une référence d'IRI [RFC3987].

4.2.7.2 Attribut "rel"

Les éléments atom:link PEUVENT avoir un attribut "rel" qui indique le type de relation de la liaison. Si l'attribut "rel" n'est pas présent, l'élément link DOIT être interprété comme si le type de relation de la liaison était "alternate".

La valeur de "rel" DOIT être une chaîne qui est non vide et correspond à la production "isegment-nz-nc" ou "IRI" dans la [RFC3987]. Noter que l'utilisation d'une référence relative autre qu'un nom simple n'est pas permise. Si un nom est donné, les mises en œuvre DOIVENT considérer le type de relation de la liaison comme équivalent au même nom enregistré par l'IANA au registre des relations de liaison (Section 7) et donc à l'IRI qui serait obtenu en ajoutant la valeur de l'attribut rel à la chaîne "http://www.iana.org/assignments/relation/". La valeur de "rel" décrit la signification de la liaison, mais n'impose pas d'exigence de comportement aux processeurs Atom.

Le présent document définit cinq valeurs initiales pour le registre des relations de liaisons :

1. La valeur "alternate" signifie que l'IRI dans la valeur de l'attribut href identifie une autre version de la ressource décrite par l'élément contenant.

2. La valeur "related" signifie que l'IRI dans la valeur de l'attribut href identifie une ressource en relation avec la ressource décrite par l'élément contenant. Par exemple, le flux pour un site qui discute des performances du moteur de recherche à "http://search.exemple.com" pourrait contenir, comme fils de atom:feed :

   <link rel="related" href="http://search.exemple.com/"/>

   Une liaison identique pourrait apparaître comme un fils de toute atom:entry dont le contenu contient une discussion de ce même moteur de recherche.

3. La valeur "self" signifie que l'IRI dans la valeur de l'attribut href identifie une ressource équivalente à l'élément contenant.

4. La valeur "enclosure" signifie que l'IRI dans la valeur de l'attribut href identifie une ressource en relation qui est potentiellement de grande taille et pourrait exiger un traitement spécial. Pour les éléments atom:link avec rel="enclosure", l'attribut length DEVRAIT être fourni.

5. La valeur "via" signifie que l'IRI dans la valeur de l'attribut href identifie une ressource qui est la source des

informations fournies dans l'élément contenant.

4.2.7.3 Attribut "type"

Sur l'élément de liaison, la valeur de l'attribut "type" est un type de suppport conseillé : c'est une indication sur le type de la représentation qu'on s'attend à voir retourner quand la valeur de l'attribut href est déréférencée. Noter que l'attribut de type n'outrepasse pas le type de support réel retourné avec la représentation. Les éléments de liaison PEUVENT avoir un attribut de type, dont la valeur DOIT se conformer à la syntaxe d'un type de support MIME [RFC4288].

4.2.7.4 Attribut "hreflang"

Le contenu de l'attribut "hreflang" décrit le langage de la ressource sur laquelle pointe l'attribut href. Utilisé avec rel="alternate", il implique une version traduite de l'entrée. Les éléments de liaison PEUVENT avoir un attribut hreflang, dont la valeur DOIT être une étiquette de langue [RFC3066].

4.2.7.5 Attribut "title"

L'attribut "title" *(titre)* porte des informations lisibles par l'homme sur la liaison. Le contenu de l'attribut "title" est sensible au langage. Les entitiés comme "&" et "<" représentent leurs caractères correspondants ("&" et "<", respectivement) et pas le balisage. Les éléments de liaison PEUVENT avoir un attribut title.

4.2.7.6 Attribut "length"

L'attribut "length" indique une longueur conseillée du contenu lié en octets ; c'est un conseil sur la longueur de contenu de la représentation retournée quand l'IRI dans l'attribut "href" est transposé en un URI et déréférencé. Noter que l'attribut de longueur n'outrepasse pas la longueur de contenu réelle de la représentation telle que rapportée par le protocole sous-jacent. Les éléments de liaison PEUVENT avoir un attribut "length".

4.2.8 Élément "atom:logo"

Le contenu de l'élément "atom:logo" est une référence d'IRI [RFC3987] qui identifie une image fournissant une identification visuelle pour un flux.

```
atomLogo = element atom:logo {
  atomCommonAttributes,
  (atomUri)
}
```

L'image DEVRAIT avoir un ratio d'aspect de 2 (horizontal) à 1 (vertical).

4.2.9 Élément "atom:published"

L'élément "atom:published" est une construction Date qui indique un instant associé à un événement antéreur dans la vie de l'entrée.

```
atomPublished = element atom:published { atomDateConstruct }
```

Normalement, atom:published va être associé à la création initiale ou à la première disponibilité de la ressource.

4.2.10 Élément "atom:rights"

L'élément "atom:rights" est une construction Text qui porte des informations sur les droits détenus dans et sur une entrée ou flux.

```
atomRights = element atom:rights { atomTextConstruct }
```

L'élément atom:rights NE DEVRAIT PAS être utilisé pour porter des informations de licence lisibles par la machine.

Si un élément atom:entry ne contient pas d'élément atom:rights, l'élément atom:rights de l'élément atom:feed conteneur, si il est présent, est considéré comme s'appliquant à l'entrée.

4.2.11 Élément "atom:source"

Si une atom:entry est copiée d'un flux à un autre flux, alors les métadonnées de l'atom:feed source (tous les éléments fils de atom:feed autres que les éléments atom:entry) PEUVENT être préservés dans l'entrée copiée en ajoutant un élément atom:source fils, si il n'est pas déjà présent dans l'entrée, et en incluant certains ou tous les éléments Metadata du flux de source comme fils de l'élément atom:source. Ces métadonnées DEVRAIENT être préservées si le atom:feed source contient un des éléments fils atom:author, atom:contributor, atom:rights, ou atom:category et que ces éléments fils ne sont pas présents dans l'atom:entry de source.

```
atomSource =
   element atom:source {
      atomCommonAttributes,
      (atomAuthor*
       & atomCategory*
       & atomContributor*
       & atomGenerator?
       & atomIcon?
       & atomId?
       & atomLink*
       & atomLogo?
       & atomRights?
       & atomSubtitle?
       & atomTitle?
       & atomUpdated?
       & extensionElement*)
   }
```

L'élément atom:source est destiné à permettre l'agrégation des entrées provenant de différents flux tout en conservant les informations sur le flux de source d'une entrée. Pour cette raison, les processeurs Atom qui effectuent une telle agrégation DEVRAIENT inclure au moins les éléments Metadata exigés au niveau du flux (atom:id, atom:title, et atom:updated) dans l'élément atom:source.

4.2.12 Élément "atom:subtitle"

L'élément "atom:subtitle" est une construction Text qui porte une description ou sous titre lisible par l'homme pour un flux.

```
atomSubtitle = element atom:subtitle { atomTextConstruct }
```

4.2.13 Élément "atom:summary"

L'élément "atom:summary" est une construction Text qui porte un bref résumé, sommaire, ou extrait d'une entrée.

```
atomSummary = element atom:summary { atomTextConstruct }
```

Il n'est pas conseillé que l'élément atom:summary duplique atom:title ou atom:content parce que les processeurs Atom pourraient supposer que c'est un résumé utile alors qu'il ne l'est pas.

4.2.14 Élément "atom:title"

L'élément "atom:title" est une construction Text qui porte un titre lisible par l'homme pour une entrée ou un flux.

```
atomTitle = element atom:title { atomTextConstruct }
```

4.2.15 Élément "atom:updated"

L'élément "atom:updated" est une construction Date qui indique le plus récent instant où une entrée ou flux a été modifié d'une façon que l'éditeur considère comme significative. Donc, toutes les modifications ne résultent pas nécessairement en un changement de la valeur de atom:updated.

```
atomUpdated = element atom:updated { atomDateConstruct }
```

Les éditeurs PEUVENT changer la valeur de cet élément au fil du temps.

5. Sécurisation des documents Atom

Parce que Atom est un format fondé sur XML, les mécanismes existants de sécurité XML peuvent être utilisés pour sécuriser son contenu.

Les producteurs de flux et/ou entrées, et les intermédiaires qui agrègent les flux et/ou entrées, peuvent avoir de très bonnes raisons pour signer et/ou chiffrer un contenu par ailleurs non protégé. Par exemple, un commerçant pourrait signer numériquement un message qui contient un coupon de rabais pour ses produits. Une banque qui utilise Atom pour livrer des états de comptes d'utilisateur va très probablement vouloir signer et chiffrer ces messages pour protéger les informations financières de ses clients et pour assurer leur authenticité au client. Les intermédiaires peuvent vouloir chiffrer des flux agrégés afin qu'un observateur passif ne puisse pas dire à quels sujets s'intéresse le receveur. Bien sûr, de nombreux autres exemples existent aussi.

Les exigences d'algorithme de cette section relèvent du processeur Atom. Elles exigent qu'un receveur soit au minimum capable de traiter les messages qui utilisent les algorithmes de chiffrement spécifés. Ces exigences ne limitent pas les algorithmes que l'envoyeur peut choisir.

5.1 Signatures numériques

La racine d'un document Atom (c'est-à-dire, atom:feed dans un document de flux Atom, atom:entry dans un document d'entrée Atom) ou tout élément atom:entry PEUT avoir une signature enveloppée, comme décrit par XML-Signature et le traitement syntaxique [W3C.REC-xmldsig-core-20020212].

Les processeurs Atom NE DOIVENT PAS rejeter un document Atom contenant une telle signature parce qu'ils ne sont pas capables de la vérifier ; ils DOIVENT continuer le traitement et PEUVENT informer l'utilisateur de leur échec à valider la signature.

En d'autres termes, la présence d'un élément avec l'URI d'espace de noms "http://www.w3.org/2000/09/xmldsig#" et un nom local de "Signature" comme fils de l'élément de document NE DOIT PAS causer l'échec d'un processeur Atom simplement à cause de sa présence.

Les autres éléments dans un document Atom NE DOIVENT PAS être signés sauf si leur définition spécifie explicitement une telle capacité.

Le paragraphe 6.5.1 de [W3C.REC-xmldsig-core-20020212] exige la prise en charge du XML canonique [W3C.REC-xml-c14n-20010315]. Cependant, de nombreuses mises en œuvre ne l'utilisent pas parce que les documents XML signés enclos dans d'autres documents XML ont leur signature coupée. Donc, les processeurs Atom qui vérifient les documents Atom signés DOIVENT être capables de canoniser avec la méthode de canonisation XML exclusive identifiée par l'URI "http://www.w3.org/2001/10/xml-exc-c14n#", comme spécifiée dans la canonisation XML exclusive [W3C.REC-xml-exc-c14n-20020718].

Les intermédiaires comme les agrégateurs peuvent avoir besoin d'ajouter un élément atom:source à une entrée qui ne contient pas son propre élément atom:source. Si une telle entrée est signée, l'ajout va casser la signature. Donc, un éditeur d'entrées signées individuellement devrait réfléchir pour ajouter un élément atom:source à ces entrées avant de les signer. Les mises en œuvre devraient aussi être conscientes des problèmes concernant l'utilisation de balisage dans l'espace de noms "xml:" car il interagit avec la canonisation.

Le paragraphe 4.4.2 de [W3C.REC-xmldsig-core-20020212] exige la prise en charge des signatures DSA et recommande la prise en charge des signatures RSA. Cependant, à cause de la plus grande popularité de RSA sur le marché que de DSA, les

processeurs Atom qui vérifient les documents Atom signés DOIVENT être capables de vérifier les signatures RSA, mais n'ont pas besoin d'être capables de vérifier les signatures DSA. Du fait des problèmes de sécurité qui peuvent survenir si le matériel de chiffrement pour l'authentification par code d'authentification de message (MAC) n'est pas traité correctement, les documents Atom NE DEVRAIENT PAS utiliser des MAC pour les signatures.

5.2 Chiffrement

La racine d'un document Atom (c'est-à-dire, atom:feed dans un document de flux Atom, atom:entry dans un document d'entrée Atom) PEUT être chiffrée, en utilisant les mécanismes décrits par la syntaxe et le traitement du chiffrement XML [W3C.REC-xmlenc-core-20021210].

Le paragraphe 5.1 de [W3C.REC-xmlenc-core-20021210] exige la prise en charge de TripleDES, AES-128, et AES-256. Les processeurs Atom qui déchiffrent les documents Atom DOIVENT être capables de déchiffrer avec AES-128 en mode de chaînage de bloc de chiffrement (CBC).

Le chiffrement fondé sur [W3C.REC-xmlenc-core-20021210] n'assure pas l'intégrité du document original. Il y a des attaques cryptographiques connues où quelqu'un qui ne peut pas déchiffrer un message peut quand même changer des bits de façon qu'une partie ou tout le message déchiffré fasse sens mais ait une signification différente. Donc, les processeurs Atom qui déchiffrent les documents Atom DEVRAIENT vérifier l'intégrité du document déchiffré en vérifiant le hachage dans la signature (si il en est une) dans le document, ou en vérifiant un hachage du document au sein du document (si il en est).

5.3 Signature et chiffrement

Quand un document Atom est à la fois signé et chiffré, c'est généralement une bonne idée de signer d'abord le document, puis de chiffrer le document signé. Cela protège l'intégrité du document de base tout en chiffrant toutes les informations, incluant l'identité de l'entité qui a signé le document. Noter que, si des MAC sont utilisés pour l'authentification, l'ordre DOIT être que le document soit signé et ensuite chiffré, et non le contraire.

6. Extension de Atom

6.1 Extensions provenant de vocabulaires non Atom

La présente spécification décrit le vocabulaire de balisage XML de Atom. Le balisage provenant d'autres vocabulaires ("balisage étranger") peut être utilisé dans un document Atom. Noter que l'élément atom:content est destiné à prendre en charge l'inclusion de balisage étranger arbitraire.

6.2 Extensions au vocabulaire Atom

L'espace de noms Atom est réservé pour de futures révisions compatibles de Atom. De futures versions de la présente spécification pourraient ajouter de nouveaux éléments et attributs au vocabulaire de balisage Atom. Le logiciel écrit pour se conformer à la présente version de la spécification ne sera pas capable de traiter correctement un tel balisage et, en fait, ne va pas être capable de le distinguer d'une erreur de balisage. Pour les besoins de cette discussion, le balisage non reconnu provenant du vocabulaire Atom va être considéré comme du "balisage étranger".

6.3 Traitement de balisage étranger

Les processeurs Atom qui rencontrent du balisage étranger dans une localisation qui est légale selon la présente spécification NE DOIVENT PAS arrêter le traitement ou signaler une erreur. Il se pourrait que le processeur Atom soit capable de traiter correctement le balisage étranger et le fasse. Autrement, ce balisage est appelé un " balisage étranger inconnu".

Quand du balisage étranger inconnu est rencontré comme fils d'une atom:entry, atom:feed, ou d'une construction Person, les processeurs Atom PEUVENT court-circuiter le balisage et tout contenu de texte et NE DOIVENT PAS changer leur comportement par suite de la présence du balisage.

Quand du balisage étranger inconnu est rencontré dans une construction Text ou un élément atom:content, le logiciel DEVRAIT ignorer le balisage et traiter tout contenu de texte d'éléments étrangers comme si le balisage enveloppant était

absent.

6.4 Éléments d'extension

Atom permet du balisage étranger partout dans un document Atom, sauf quand il est explicitement interdit. Les éléments fils de atom:entry, atom:feed, atom:source, et des constructions Person sont considérés comme des éléments de métadonnées et sont décrits ci-dessous. Les éléments fils des constructions Person sont considérés comme s'appliquant à la construction. Le rôle des autres balisages étrangers est non défini dans la présente spécification.

6.4.1 Éléments d'extension simple

Un élément Simple Extension NE DOIT PAS avoir d'attributs ou éléments fils. L'élément PEUT contenir des données de caractères ou être vide. Les éléments Simple Extension ne sont pas sensibles au langage.

```
simpleExtensionElement =
  element * - atom:* {
    text
  }
```

L'élément peut être interprété comme une simple propriété (ou paire nom/valeur) de l'élément parent qui l'enclot. La paire consistant en l'URI d'espace de noms de l'élément et le nom local de l'élément peut être interprétée comme le nom de la propriété. Le contenu de données de caractères de l'élément peut être interprété comme la valeur de la propriété. Si l'élément est vide, alors la valeur de la propriété peut être interprétée comme une chaîne vide.

6.4.2 Éléments d'extension structurés

L'élément racine d'un élément Extension structurée DOIT avoir au moins un attribut ou élément fils. Il PEUT avoir des attributs, il PEUT contenir un contenu XML bien formé (incluant des données de caractères) ou il PEUT être vide. Les éléments Extension structurée sont sensibles au langage.

```
structuredExtensionElement =
  element * - atom:* {
    (attribute * { text }+,
      (text|anyElement)*)
   | (attribute * { text }*,
     (text?, anyElement+, (text|anyElement)*))
  }
```

La structure d'un élément Extension structurée, incluant l'ordre de ses éléments fils, pourrait être significatif.

La présente spécification ne donne pas d'interprétation d'un élément Extension structurée. La syntaxe du XML contenu dans l'élément (et une interpretation de comment l'élément se rapporte à son élément contenant) est définie par la spécification de l'extension Atom.

7. Considérations relatives à l'IANA

Un document Atom, quand il est documenté comme XML 1.0, peut être identifié par le type de support suivant :

Nom de type de support MIME : application
Nom de sous type MIME : atom+xml
Paramètres obligatoires : aucun.
Paramètres facultatifs :
"charset" : ce paramètre a une sémantique identique au paramètre charset du type de support "application/xml" comme spécifié dans la [RFC3023].
Considérations de codage : Identiques à celles de "application/xml" comme décrit au paragraphe 3.2 de la [RFC3023].
Considérations de sécurité : Comme définies dans la présente spécification. De plus, comme ce type de support utilise la convention "+xml", il partage les mêmes considérations de sécurité que décrites à la Section 10 de la [RFC3023].

Considérations d'interopérabilité : il n'y a pas de problème d'interopérabilité connu.
Spécification publiée : la présente spécification.
Applications qui utilisent de type de supports : aucune application connue n'utilise actuellement ce type de supports.

Informations supplémentaires :
Numéros magiques : comme spécifié pour "application/xml" au paragraphe 3.2 de la [RFC3023].
Extension de fichier : .atom
Identifiants de fragment : comme spécifié pour "application/xml" à la Secion 5 de la [RFC3023].
URI de base : comme spécifié à la Section 6 de la [RFC3023].
Code de type de fichier Macintosh : TEXT
Adresse personnelle et de messagerie à contacter pour plus d'informations : Mark Nottingham <mnot@pobox.com>
Usage prévu : COMMUN
Auteur/contrôleur des changements : IESG

7.1 Registre des relations de liaisons

Ce registre est tenu par l'IANA et contient initialement cinq valeurs : "alternate", "related", "self", "enclosure", et "via". Les nouvelles allocations sont sujettes à approbation de l'IESG, comme précisé dans la [RFC2434]. Les demandes devraient être faites par message électronique à l'IANA, qui transmettra la demande à l'IESG, pour approbation. La demande devrait utiliser le gabarit suivant :

- Valeur d'attribut : (une valeur pour l'attribut "rel" qui se conforme à la règle de syntaxe donnée au paragraphe 4.2.7.2)
- Description :
- Caractéristiques d'affichage attendues :
- Considérations de sécurité :

8. Considérations sur la sécurité

8.1 Contenu HTML et XHTML

Les constructions Text et atom:content permettent la livraison de HTML et XHTML. De nombreux éléments dans ces langages sont considérés comme "non sûrs" en ce qu'ils ouvrent les clients à un ou plusieurs types d'attaques. Les mises en œuvre de logiciels qui traitent Atom devraient considérer avec attention leur traitemenr de chaque type d'élément lors du traitement de (X)HTML entrant dans les documents Atom. Voir les sections de sécurité de [RFC2854] et [HTML] pour des lignes directrices.

Les processeurs Atom devraient porter une attention particulière à la sécurité des éléments IMG, SCRIPT, EMBED, OBJECT, FRAME, FRAMESET, IFRAME, META, et LINK, mais d'autres éléments pourraient aussi avoir des propriétés de sécurité négatives.

(X)HTML peut contenir soit directement soit indirectement un contenu de référence exécutable.

8.2 URI

Les processeurs Atom traitent les URI. Voir la Section 7 de la [RFC3986].

8.3 IRI

Les processeurs Atom traitent les IRI. Voir la Section 8 de la [RFC3987].

8.4 Usurpation d'identité

Les processeurs Atom devraient être conscients du potentiel d'attaques en usurpation d'identité où l'attaquant publie une atom:entry avec la valeur de atom:id d'une entrée provenant d'un autre flux, peut-être avec un élément atom:source falsifié qui duplique le atom:id de l'autre flux. Par exemple, un processeur Atom pourrait supprimer l'affichage d'entrées dupliquées en affichant seulement une entrée sur l'ensemble des entrées avec des valeurs de atom:id identiques. Dans cette situation, le processeur Atom pourrait aussi prendre des mesures pour déterminer si les entrées sont originaires du même éditeur avant de les considérer comme des dupliquées.

8.5 Chiffrement et signature

Les documents Atom peut être chiffrés et signés en utilisant [W3C.REC-xmlenc-core-20021210] et [W3C.REC-xmldsig-core-20020212], respectivement, et sont sujets aux considérations de sécurité impliquées par leur utilisation.

Les signatures numériques assurent l'authentification, l'intégrité de message, et la non répudiation avec preuve de l'origine. Le chiffrement assure la confidentialité des données.

9. Références

9.1 Références normatives

[HTML] Raggett, D., Hors, A., and I. Jacobs, "HTML 4.01 Specification", W3C REC-html401-19991224, décembre 1999, <http://www.w3.org/TR/1999/REC-html401-19991224>.

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

[RFC2822] P. Resnick, "Format de message Internet", avril 2001. *(Remplace la* RFC0822, STD 11, *Remplacée par* RFC5322)

[RFC2854] D. Connelly et L. Masinter, "Type de support 'text/html'", juin 2000. *(Information)*

[RFC3023] M. Murata, S. St.Laurent et D. Kohn, "Types de supports XML", janvier 2001. (*Obsolète, voir* RFC7303)

[RFC3066] H. Alvestrand, "Étiquettes pour l'identification des langues", BCP 47, janvier 2001. *(Obsolète, voir la* RFC4646.)

[RFC3339] G. Klyne, C. Newman, "La date et l'heure sur l'Internet : horodatages", juillet 2002. *(P.S.)*

[RFC3548] S. Josefsson, "Codages de données Base16, Base32, et Base64", juillet 2003. *(Obsolète, voir* 4648) *(Info)*

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

[RFC3987] M. Duerst et M. Suignard, "Identifiant de ressource internationalisé (IRI)", janvier 2005.

[RFC4288] N. Freed et J. Klensin, "Spécifications du type de support et procédures d'enregistrement", BCP 13, décembre 2005.

[W3C.REC-xml-20040204] Yergeau, F., Paoli, J., Sperberg-McQueen, C., Bray, T., and E. Maler, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC REC-xml-20040204, février 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[W3C.REC-xml-c14n-20010315] Boyer, J., "Canonical XML Version 1.0", W3C REC REC-xml-c14n-20010315, mars 2001, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>.

[W3C.REC-xml-exc-c14n-20020718] Eastlake, D., Boyer, J., et J. Reagle, "Exclusive XML Canonicalization Version 1.0", W3C REC REC-xml-exc-c14n-20020718, July 2002, <http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718>.

[W3C.REC-xml-infoset-20040204] Cowan, J. and R. Tobin, "XML Information Set (Second Edition)", W3C REC REC-xml-infoset-20040204, février 2004, <http://www.w3.org/TR/2004/REC-xml-infoset-20040204>.

[W3C.REC-xml-names-19990114] Hollander, D., Bray, T., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, janvier 1999, <http://www.w3.org/TR/1999/REC-xml-names-19990114>.

[W3C.REC-xmlbase-20010627] Marsh, J., "XML Base", W3C REC REC-xmlbase-20010627, juin 2001, <http://www.w3.org/TR/2001/REC-xmlbase-20010627>.

[W3C.REC-xmldsig-core-20020212] Solo, D., Reagle, J., and D. Eastlake, "XML-Signature Syntax and Processing", W3C REC REC-xmldsig-core-20020212, février 2002, <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212>.

[W3C.REC-xmlenc-core-20021210] Reagle, J. and D. Eastlake, "XML Encryption Syntax and Processing", W3C REC REC-xmlenc-core-20021210, décembre 2002, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>.

[XHTML] Altheim, M., Boumphrey, F., McCarron, S., Dooley, S., Schnitzenbaumer, S., and T. Wugofski, "Modularization of XHTML[TM]", W3C REC REC-xhtml-modularization-20010410, avril 2001, <http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410>.

9.2 Références pour information

[ISO.8601] Organisation Internationale de normalisation, "Éléments de données et formats d'échange - Échanges d'informations - Représentation de la date et de l'heure", norme ISO 8601, juin 1988.

[RELAX-NG] Clark, J., "RELAX NG Compact Syntax", décembre 2001, <http://www.oasis-open.org/committees/relax-ng/compact-20021121.html>.

[RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre 1998. *(Rendue obsolète par la* RFC5226)

[W3C.NOTE-datetime-19980827] Wolf, M. and C. Wicksteed, "Date and Time Formats", W3C NOTE-datetime-19980827, août 1998, <http://www.w3.org/TR/1998/NOTE-datetime-19980827>.

[W3C.REC-xmlschema-2-20041028] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", W3C REC REC-xmlschema-2-20041028, octobre 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

Appendice A. Contributeurs

Les personnes suivantes ont contribué aux versions préliminaires de ce document : Tim Bray, Mark Pilgrim, et Sam Ruby. Norman Walsh a fourni le schéma Relax NG. Le contenu et les concepts ont été produits par la communauté Atom et le groupe de travail Atompub.

Le groupe de travail Atompub a des douzaines de contributeurs très actifs qui ont proposé des idées et la formulation de ce document, incluant : Danny Ayers, James Aylett, Roger Benningfield, Arve Bersvendsen, Tim Bray, Dan Brickley, Thomas Broyer, Robin Cover, Bill de Hora, Martin Duerst, Roy Fielding, Joe Gregorio, Bjoern Hoehrmann, Paul Hoffman, Anne van Kesteren, Brett Lindsley, Dare Obasanjo, David Orchard, Aristotle Pagaltzis, John Panzer, Graham Parks, Dave Pawson, Mark Pilgrim, David Powell, Julian Reschke, Phil Ringnalda, Antone Roundy, Sam Ruby, Eric Scheid, Brent Simmons, Henri Sivonen, Ray Slakinski, James Snell, Henry Story, Asbjorn Ulsberg, Walter Underwood, Norman Walsh, Dave Winer, et Bob Wyman.

Appendice B. Schéma compact RELAX NG

Cet Appendice est pour information.

Le schéma Relax NG exclut explicitement les éléments de l'espace de noms Atom qui ne sont pas définis dans cette révision de la spécification. Les exigences pour les processeurs Atom qui rencontrent un tel balisage sont données aux paragraphes 6.2 et 6.3.

```
# -*- rnc -*-
# Grammaire de la syntaxe compacte de RELAX NG pour la spécification de format Atom version 11

namespace atom = "http://www.w3.org/2005/Atom"
```

```
namespace xhtml = "http://www.w3.org/1999/xhtml"
namespace s = "http://www.ascc.net/xml/schematron"
namespace local = ""

start = atomFeed | atomEntry

# Attributs communs

atomCommonAttributes =
  attribute xml:base { atomUri }?,
  attribute xml:lang { atomLanguageTag }?,
  undefinedAttribute*

# Constructions Text

atomPlainTextConstruct =
  atomCommonAttributes,
  attribute type { "text" | "html" }?,
  text

atomXHTMLTextConstruct =
  atomCommonAttributes,
  attribute type { "xhtml" },
  xhtmlDiv

atomTextConstruct = atomPlainTextConstruct | atomXHTMLTextConstruct

# Constructions Person

atomPersonConstruct =
  atomCommonAttributes,
  (element atom:name { text }
   & element atom:uri { atomUri }?
   & element atom:email { atomEmailAddress }?
   & extensionElement*)

# Constructions Date

atomDateConstruct =
  atomCommonAttributes,
  xsd:dateTime

# atom:feed

atomFeed =
  [
    s:rule [
      context = "atom:feed"
      s:assert [
        test = "atom:author ou non(atom:entry[non(atom:author)])"
          "Un atom:feed doit avoir un atom:author sauf si tous ses atom:entry fils ont un atom:author."
      ]
    ]
  ]
  element atom:feed {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContributor*
     & atomGenerator?
     & atomIcon?
```

```
      & atomId
      & atomLink*
      & atomLogo?
      & atomRights?
      & atomSubtitle?
      & atomTitle
      & atomUpdated
      & extensionElement*),
     atomEntry*
   }
```

 # atom:entry

```
atomEntry =
  [
    s:rule [
      context = "atom:entry"
      s:assert [
        test = "atom:link[@rel='alternate'] "
        ~ "ou atom:link[not(@rel)] "
        ~ "ou atom:content"
        "Une atom:entry doit avoir au moins un élément atom:link avec un attribut rel de 'alternate'ou un atom:content."
      ]
    ]
    s:rule [
      context = "atom:entry"
      s:assert [
        test = "atom:author ou "
        ~ "../atom:author ou atom:source/atom:author"
        "Une atom:entry doit avoir un atom:author si son flux n'en a pas."
      ]
    ]
  ]
  element atom:entry {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContent?
     & atomContributor*
     & atomId
     & atomLink*
     & atomPublished?
     & atomRights?
     & atomSource?
     & atomSummary?
     & atomTitle
     & atomUpdated
     & extensionElement*)
  }
```

 # atom:content

```
atomInlineTextContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { "text" | "html" }?,
    (text)*
  }

atomInlineXHTMLContent =
  element atom:content {
```

```
    atomCommonAttributes,
    attribute type { "xhtml" },
    xhtmlDiv
  }

atomInlineOtherContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    (text|anyElement)*
  }

atomOutOfLineContent =
  element atom:content {
    atomCommonAttributes,
    attribute type { atomMediaType }?,
    attribute src { atomUri },
    empty
  }

atomContent = atomInlineTextContent
| atomInlineXHTMLContent
| atomInlineOtherContent
| atomOutOfLineContent
```

 # atom:author

```
atomAuthor = element atom:author { atomPersonConstruct }
```

 # atom:category

```
atomCategory =
  element atom:category {
    atomCommonAttributes,
    attribute term { text },
    attribute scheme { atomUri }?,
    attribute label { text }?,
    undefinedContent
  }
```

 # atom:contributor

```
atomContributor = element atom:contributor { atomPersonConstruct }
```

 # atom:generator

```
atomGenerator = element atom:generator {
  atomCommonAttributes,
  attribute uri { atomUri }?,
  attribute version { text }?,
  text
}
```

 # atom:icon

```
atomIcon = element atom:icon {
  atomCommonAttributes,
  (atomUri)
}
```

 # atom:id

```
atomId = element atom:id {
  atomCommonAttributes,
  (atomUri)
}
```

 # atom:logo

```
atomLogo = element atom:logo {
  atomCommonAttributes,
  (atomUri)
}
```

 # atom:link

```
atomLink =
  element atom:link {
    atomCommonAttributes,
    attribute href { atomUri },
    attribute rel { atomNCName | atomUri }?,
    attribute type { atomMediaType }?,
    attribute hreflang { atomLanguageTag }?,
    attribute title { text }?,
    attribute length { text }?,
    undefinedContent
  }
```

 # atom:published

```
atomPublished = element atom:published { atomDateConstruct }
```

 # atom:rights

```
atomRights = element atom:rights { atomTextConstruct }
```

 # atom:source

```
atomSource =
  element atom:source {
    atomCommonAttributes,
    (atomAuthor*
     & atomCategory*
     & atomContributor*
     & atomGenerator?
     & atomIcon?
     & atomId?
     & atomLink*
     & atomLogo?
     & atomRights?
     & atomSubtitle?
     & atomTitle?
     & atomUpdated?
     & extensionElement*)
  }
```

 # atom:subtitle

```
atomSubtitle = element atom:subtitle { atomTextConstruct }
```

 # atom:summary

```
atomSummary = element atom:summary { atomTextConstruct }

# atom:title

atomTitle = element atom:title { atomTextConstruct }

# atom:updated

atomUpdated = element atom:updated { atomDateConstruct }

# Types simples de niveau inférieur

atomNCName = xsd:string { minLength = "1" pattern = "[^:]*" }

# Quel que soit un type de supports, il contient au moins une barre oblique
atomMediaType = xsd:string { pattern = ".+/.+" }

# Comme défini dans la RFC 3066
atomLanguageTag = xsd:string {
  pattern = "[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*"
}

# Sans contrainte ; la façon dont un IRI tient dans un URI xsd:any n'est pas entièrement claire donc on n'essaye pas ici de
contraindre atomUri = text
# Quelle que soit une adresse de messagerie élecronique, elle contient au moins une @
atomEmailAddress = xsd:string { pattern = ".+@.+" }

# Extension simple

simpleExtensionElement =
  element * - atom:* {
    text
  }

# Extension structurée

structuredExtensionElement =
  element * - atom:* {
    (attribute * { text }+,
      (text|anyElement)*)
  | (attribute * { text }*,
     (text?, anyElement+, (text|anyElement)*))
  }

# Autre extensibilité

extensionElement = simpleExtensionElement | structuredExtensionElement

undefinedAttribute = attribute * - (xml:base | xml:lang | local:*) { text }

undefinedContent = (text|anyForeignElement)*

anyElement =
  element * {
    (attribute * { text }
     | text
     | anyElement)*
  }

anyForeignElement =
  element * - atom:* {
```

```
    (attribute * { text }
     | text
     | anyElement)*
  }

# XHTML

anyXHTML = element xhtml:* {
  (attribute * { text }
   | text
   | anyXHTML)*
}

xhtmlDiv = element xhtml:div {
  (attribute * { text }
   | text
   | anyXHTML)*
}

# EOF
```

Adresse des auteurs

Mark Nottingham
mél : mnot@pobox.com
URI : http://www.mnot.net/

Robert Sayre
mél : rfsayre@boswijck.com
URI : http://boswijck.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est fourni par l'activité de soutien administratif de l'IETF (IASA).