ESPPADOM
Echanges financeurs – prestataires pour les services à domicile auprès des personnes en perte d'autonomie

Programme soutenu par
la Caisse nationale de solidarité pour l'autonomie

EVOLUTION DU SCHEMA XSD

Résumé : Ce document détaille les évolutions du schéma xsd de description de l'ensemble des messages Esppadom à partir de la version 1.1.
Les évolutions antérieures à cette première « version officielle » sont détaillées dans des guides qui rendent compte des décisions prises lors des ateliers du premier trimestre 2016 et sont donc, comme l'ont été les ateliers, spécifiques à chaque message

Statut	Document de travail
Version	1.3
Date	21/11/2016
Auteurs	EDESS – Philippe AMELINE, François ROUGERIE

Evolutions

V 1.2.1	12/07/2016	Version initiale, suite aux ateliers du 17/06/2016 et du 07/07/2016
V1.2.3	21/11/2016	Mise à jour suite à l'atelier du 17/11/16

Table des matières

1 CONTEXTE

Les schémas xsd d'Esppadom décrivent l'arborescence des balises qui portent les informations contenues dans chaque message (ORDER, DELIVERY et INVOICE), ainsi que des « objets » qu'ils partagent (comme la description des personnes et de leurs contacts). Les schémas définissent également la cardinalité de ces balises ainsi que les informations de formatage des données qu'elles hébergent.

Modifier le schéma xsd est une opération délicate, puisqu'elle peut obliger à modifier les systèmes déjà fonctionnels. Il est donc fondamental de ne le faire qu'à bon escient, et, si nécessaire, le plus tôt possible afin de limiter le nombre de sites impactés.

Ce document détaille les modifications apportées au sein de chaque nouvelle version du standard, généralement à l'issue d'une réunion technique. Il a pour vocation de servir de manuel d'instruction aux équipes ayant déjà mis en œuvre la version précédente du standard, mais aussi, dans un esprit de « mémoire des organisations », de laisser une trace de ces évolutions et des raisons qui les ont motivées.

2 VERSION 1.2

La version 1.2 a été mise en œuvre le 11/07/2016, suite aux décisions prises lors des ateliers techniques du 17/06/2016 et du 07/07/2016.

2.1 OBJETS PARTAGÉS

2.1.1 Ajout d'un identifiant et d'un type de profil aux contacts

Afin, par exemple, d'identifier l'agent ayant réalisé l'acte, il a été décidé d'ajouter un identifiant au bloc de description des contacts.

De la même façon, il a été décidé d'ajouter une balise afin de pouvoir comparer le profil de cet intervenant à celui qui était éventuellement spécifié dans le message Order dans le bloc :

CrossIndustryOrder / CIOHSupplyChainTradeTransaction / IncludedCIOLSupplyChainTradeLineItem / SpecifiedCIOLSupplyChainTradeDelivery / DeliveryCIDeliveryInstructions / HandlingCode

Le bloc de type CITradeContactType devient donc, avec l'ajout d'une balise optionnelle **ID** et d'une balise optionnelle **HandlingCode** :

```
<xsd:complexType name="CITradeContactType">
   <xsd:sequence>
      <xsd:element name="ID" type=”IDQualifiedType" minOccurs="0"/>
      <xsd:element name="PersonName" type="TextType" minOccurs="0"/>
      <xsd:element name="TypeCode" type="CodeQualifiedType" minOccurs="0"/>
      <xsd:element name="TelephoneCIUniversalCommunication" type="CIUniversalCommunicationNumberType" minOccurs="0"/>
      <xsd:element name="MobileTelephoneCIUniversalCommunication" type="CIUniversalCommunicationNumberType"
minOccurs="0"/>
      <xsd:element name="EmailURICIUniversalCommunication" type="CIUniversalCommunicationUnqualifiedURIType" minOccurs="0"/>
      <xsd:element name="PostalAddress" type="CITradeAddressType" minOccurs="0"/>
      <xsd:element name="HandlingCode" type="CodeQualifiedType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Un éditeur avait proposé d'entamer la création d'une liste de codes à partir des valeurs « Agent à domicile », « Employé à domicile » et « Auxiliaire de vie à domicile ». La liste finale des profils, comme celle des actes devra probablement être organisée en arborescence, et l'atelier du 07/07/2106 a dégagé un consensus sur le danger qu'il y aurait à amorcer une liste trop arbitrairement – d'autant plus que la liste des profils utilisée pour qualifier l'intervenant au sein du message Delivery sera nécessairement commune à celle utilisée pour préciser le profil requis au sein du message Order.
Dans un premier temps, la liste utilisée pour alimenter ces deux balises devra donc être établie localement.

2.2 ORDER

2.2.1 Optionnalité du document de relation contractuelle

Le bloc d'informations qui décrit le donneur d'ordre et le prestataire permet également, par l'intermédiaire de la balise ContractReferencedCIReferencedDocument de préciser leurs relations contractuelles. Ce bloc était obligatoire, alors que son usage au sein d'Esppadom sera très probablement anecdotique. Il a été décidé de le rendre optionnel.

Le type CIOHSupplyChainTradeAgreementType devient ainsi :

```
<xsd:complexType name="CIOHSupplyChainTradeAgreementType">
   <xsd:sequence>
      <xsd:element name="SellerCITradeParty" type="CITradePartyType"/>
      <xsd:element name="BuyerCITradeParty" type="CITradePartyType"/>
      <xsd:element name="ContractReferencedCIReferencedDocument" type="CIReferencedDocumentType" minOccurs="0"/>
```

```
   </xsd:sequence>
</xsd:complexType>
```

2.3 DELIVERY

Il a été exprimé, lors de l'atelier du 17/06/2016, l'intérêt de pouvoir utiliser Delivery comme support du mécanisme de pointage, ce qui était rendu difficile par le statut obligatoire des dates de début et de fin au sein du bloc qui permet de préciser l'horodatage :

```
<xsd:complexType name="CIDTimeStampPeriodType">
   <xsd:sequence>
      <xsd:element name="StartDateTime" type="CIDDateTimeStampType"/>
      <xsd:element name="EndDateTime" type="CIDDateTimeStampType"/>
   </xsd:sequence>
</xsd:complexType>
```

Il est clair que, lors de l'horodatage d'arrivée du prestataire, il n'est pas possible de remplir les informations d'horodatage de départ. Par ailleurs, si le système d'horodatage permet d'indiquer le statut « arrivée » vs « départ » du prestataire, il sera alors logique, pour véhiculer chacun des événements d'utiliser l'une ou l'autre des balises.
Il a donc été décidé de rendre les deux balises optionnelles, et de faire évoluer le bloc vers :

```
<xsd:complexType name="CIDTimeStampPeriodType">
   <xsd:sequence>
      <xsd:element name="StartDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
      <xsd:element name="EndDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Dans ce cas d'usage de « Delivery à fin de pointage/horodatage », il est évident que le bloc qui précise la période retenue (et reste obligatoire) n'a plus de sens. On pourra le signifier explicitement en utilisant le code HOR pour renseigner la balise TypeCode qui porte habituellement le « motif de correction ».

CrossIndustryDespatchAdvice / CIDDHSupplyChainTradeTransaction / ApplicableCIDDHSupplyChainTradeDelivery / ActualDespatchCISupplyChainEvent / TypeCode

2.4 AJOUT D'UN NUMÉRO DE VERSION

L'évolution des schémas xsd modifie les contraintes et il est important, que ce soit pendant la période de basculement d'une version à l'autre, ou lors du traitement d'un message historisé, de savoir quelle version de schéma utiliser afin de vérifier la conformité d'un fichier à analyser.

Il a ainsi été décidé, lors de l'atelier du 17/06/2016, d'intégrer un numéro de version aux messages Esppadom.

Plusieurs options se présentaient :

- ajouter le numéro de version au sein de l'enveloppe (balises order, delivery et invoice créées au sein d'Esppadom afin de grouper les envois de messages au sein d'un même fichier XML),
- ajouter le numéro de version comme attribut de la véritable balise racine des messages UN/CEFACT (CrossIndustryOrder, CrossIndustryDespatchAdvice et CrossIndustryInvoice),
- enfin l'ajouter comme balise au sein de la balise « DocumentContext » de ces mêmes messages.

Ce dilemme nous a permis de constater que l'enveloppe et les messages qu'elle contient sont deux entités distinctes et qu'il était utile de créer une information de version pour chacune d'entre elles (on peut, par exemple, imaginer qu'une enveloppe récente véhicule des messages plus anciens).

On constate par ailleurs que les messages UN/CEFACT n'utilisent pas les attributs pour véhiculer des informations autonomes. Il a donc été décidé, pour des raisons d'homogénéité de structure, d'ajouter une balise au sein du bloc DocumentContext des messages.

2.4.1 Version d'enveloppe

L'enveloppe était « mono-balise », il était naturel d'ajouter la version au sein d'un attribut :

```
<xsd:element name="order">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="CrossIndustryOrder" type="CrossIndustryOrderType" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="versionID" type="xsd:token" use="required"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="delivery">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element maxOccurs="unbounded" name="CrossIndustryDespatchAdvice" type="CrossIndustryDespatchAdviceType"/>
      </xsd:sequence>
      <xsd:attribute name="versionID" type="xsd:token" use="required"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="invoice">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="CrossIndustryInvoice" type="CrossIndustryInvoiceType" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="versionID" type="xsd:token" use="required"/>
   </xsd:complexType>
</xsd:element>
```

2.4.2 Version des messages

La version du message doit désormais être spécifiée par une balise **VersionID** au sein du bloc DocumentContext :

Au sein du message Order :

```
<xsd:complexType name="CIOHExchangedDocumentContextType">
   <xsd:sequence>
      <xsd:element name="VersionID" type="xsd:token"/>
      <xsd:element name="SpecifiedTransactionID" type="IDUnqualifiedType"/>
      <xsd:element name="OrderContextParameter" type="CIOHDocumentContextParameterType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

Au sein du message Delivery :

```
<xsd:complexType name="CIDDHExchangedDocumentContextType">
   <xsd:sequence>
      <xsd:element name="VersionID" type="xsd:token"/>
      <xsd:element name="SpecifiedTransactionID" type="IDUnqualifiedType"/>
      <xsd:element name="DeliveryContextParameter" type="CIDeliveryContextParameterType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Au sein du message Invoice :

```
<xsd:complexType name="CIIHExchangedDocumentContextType">
   <xsd:sequence>
      <xsd:element name="VersionID" type="xsd:token"/>
      <xsd:element name="SpecifiedTransactionID" type="IDUnqualifiedType"/>
   </xsd:sequence>
</xsd:complexType>
```

3 VERSION 1.3

La version 1.3 a été mise en œuvre le 18/11/2016, suite aux décisions prises lors de l'atelier technique du 17/11/2016.

3.1 ORDER

3.1.1 Ajout d'une information libre de temporalité de prestation

Le bloc d'informations qui décrit la temporalité des prestations ou des actes ne contenait qu'un bloc de données structurées au format RRULE. Compte tenu du fait que la temporalité est généralement exprimé en français, cette limitation aurait supposé que ce verbatim soit systématiquement exprimé en formalisme RRULE (avec les risques d'erreurs induits). Par ailleurs, le format RRULE, qui est conçu pour transcrire des informations d'agenda, ne gère pas les temporalités floues (comme « le matin », « le soir », etc).

Il a donc été décidé d'ajouter un élément d'information afin de pouvoir décrire la temporalité en langage naturel.

Le type CIOHSupplyChainTradeAgreementType devient ainsi :

```
<xsd:complexType name="CIDeliveryInstructionsType">
   <xsd:sequence>
      <xsd:element name="HandlingCode" type="UNECE_UDT_9p0:CodeQualifiedType" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Description" type="UNECE_UDT_9p0:TextType" minOccurs="0"  maxOccurs="1"/>
      <xsd:element name="RRule" type="UNECE_UDT_9p0:TextType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
```

Le nom de balise « Description » est celui qui est utilisé dans le message CrossIndustryOrder dont Order est dérivé.

Il a, par ailleurs, été évoqué d'étendre RRULE afin d'exprimer de façon structurée les informations de temporalité du domaine de l'aide à domicile ; ces extensions seront détaillées dans le guide d'implémentation et n'impactent pas la structure du message.

3.2 DELIVERY

3.2.1 Déplacement du fichier binaire d'horodatage

L'horodatage, qui a beaucoup évolué lors des récentes versions, était géré par les blocs d'information suivants :

```
<xsd:complexType name="CIReferencedDocumentSignatureType">
   <xsd:sequence>
      <xsd:element name="AttachedSpecifiedBinaryFile" type="UNECE_QDT_8p0:SpecifiedBinaryFileType" minOccurs="0"/>
      <xsd:element name="EffectiveCISpecifiedPeriod" type="UNECE_RAM_8p0:CIDTimeStampPeriodType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CIDTimeStampPeriodType">
   <xsd:sequence>
      <xsd:element name="StartDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
      <xsd:element name="EndDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CIDDateTimeStampType">
   <xsd:sequence>
      <xsd:element name="CertifiedDateTime" type="UNECE_QDT_8p0:DateMandatoryDateTimeType"/>
      <xsd:element name="CustomerIdentificationMethod" type="UNECE_UDT_9p0:CodeQualifiedType"/>
      <xsd:element name="SupplierIdentificationMethod" type="UNECE_UDT_9p0:CodeQualifiedType"/>
   </xsd:sequence>
```

```
</xsd:complexType>
```

On remarque donc qu'il peut exister un pointage d'arrivée et un pointage de début, chacun d'entre eux pouvant donner lieu à une information de mode d'identification du prestataire et de mode d'identification du bénéficiaire, soit potentiellement quatre éléments d'information, alors que le fichier binaire, qui les matérialise (balise AttachedSpecifiedBinaryFile) est resté unique.

Il a été décidé qu'il ne devrait exister qu'un seul fichier binaire pour un bloc [pointage, identification du bénéficiaire, identification du bénéficiaire], et que la balise AttachedSpecifiedBinaryFile devait être déplacée du type CIReferencedDocumentSignatureType vers le type CIDDateTimeStampType, au sein duquel elle reste unique. Les blocs impactés deviennent donc :

```
<xsd:complexType name="CIReferencedDocumentSignatureType">
   <xsd:sequence>
      <xsd:element name="EffectiveCISpecifiedPeriod" type="UNECE_RAM_8p0:CIDTimeStampPeriodType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CIDTimeStampPeriodType">
   <xsd:sequence>
      <xsd:element name="StartDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
      <xsd:element name="EndDateTime" type="CIDDateTimeStampType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CIDDateTimeStampType">
   <xsd:sequence>
      <xsd:element name="CertifiedDateTime" type="UNECE_QDT_8p0:DateMandatoryDateTimeType"/>
      <xsd:element name="CustomerIdentificationMethod" type="UNECE_UDT_9p0:CodeQualifiedType"/>
      <xsd:element name="SupplierIdentificationMethod" type="UNECE_UDT_9p0:CodeQualifiedType"/>
      <xsd:element name="AttachedSpecifiedBinaryFile" type="UNECE_QDT_8p0:SpecifiedBinaryFileType" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```