



# PHAST

Promoteur d'interopérabilité



# Mode d'emploi

Identification du document	
Référence	20240502_ParserIO_Doc_v2.16.docx
Date de création	22/12/2010
Date de dernière mise à jour	02/05/2024
Nombre de pages	13

Identification du projet	
Nom du projet	Description
ParserIO.Benchmark	Programme de test utilisant le jeu d'essai Barcodestore.xml
ParserIO.Benchmark.Compare	Programme de test de résultats à utiliser avec ParserIO.Benchmark
ParserIO.Console	Client exemple utilisant le Core de type Console
ParserIO.Core	Core avec la classe Functions
ParserIO.Core.UnitTest	Tests unitaires des méthodes de la classe Functions
ParserIO.DAL	Data Access Layer
ParserIO.DAO	Data Access Object
ParserIO.Database	Ce projet sert à initialiser une base MS SQL Server
ParserIO.ORM	Object Relational Mapping
ParserIO.Tools	Une collection d'outils
ParserIO.Tools.Console	Client Console du projet ParserIO.Tools
ParserIO.WinFormsApp	Client exemple utilisant le Core de type Windows Forms
ParserIO_COMClient	Client exemple en C++ utilisant l'interface COM

Historique du document	
Date	Action
22/12/2010	Publication de la première version par Davide Usai (Phast).
24/01/2011	Ajout du diagramme des classes et aligné sur la version 0.20 du Parser par Davide Usai (Phast).
02/03/2011	Aligné sur la version 0.22 du Parser par Davide Usai (Phast).
17/03/2011	Aligné sur la version 0.24 du Parser par Davide Usai (Phast).
28/04/2011	Aligné sur la version 0.26 du ParserIO par Davide Usai (Phast).
07/11/2011	Aligné sur la version 0.28 du ParserIO par Davide Usai (Phast). Ajout du jeu d'essai, du projet Benchmark et du projet DeltaXML.
10/02/2012	Aligné sur la version 0.30 du ParserIO par Davide Usai (Phast).
27/09/2013	Aligné sur la version 1.0.0.0 du ParserIO par Davide Usai (Phast).
26/02/2014	Aligné sur la version 1.0.0.1 du ParserIO par Davide Usai (Phast).
12/05/2014	Aligné sur la version 1.0.0.3 du ParserIO par Davide Usai (Phast).
18/12/2014	Aligné sur la version 1.0.0.4 du ParserIO par Davide Usai (Phast).

24/06/2015	Aligné sur la version 1.0.0.5 du ParserIO par Davide Usai (Phast).
13/11/2015	Aligné sur la version 1.0.0.6 du ParserIO par Davide Usai (Phast).
10/12/2015	Ajouté un chapitre concernant les standards par Davide Usai (Phast).
05/04/2016	Développement de fonctionnalités pour le support du standard HIBC 2.5.
28/06/2016	Publication de ParserIO en service de type WCF et WebAPI, mise à jour majeure des outils par Davide Usai (Phast).
29/09/2016	Gestion de la famille INTERNAL (91-99) de la norme GS1 et développement de l'algorithme en mode récursive par Davide Usai(Phast).
09/08/2017	Aligné sur la version 2.2.0.0 du ParserIO.
12/09/2017	Aligné sur la version 2.2.0.2 du ParserIO.
02/10/2017	Aligné sur la version 2.2.0.4 du ParserIO.
01/03/2018	Version 2.2.0.6.  Gestion de l'AI GS1 241 : Customer Part Number Layout Windows Form avec l'utilisation d'un un Tab Control
11/04/2018	Version 2.2.0.8 List des identifiants Retour de la version de ParserIO
27/06/2018	Version 2.2.1.0 Report des valeurs GTIN-13 ou ACL-13 dans le champ UDI du ParserIO
30/08/2018	Version 2.2.1.1 Correction d'un bug qui ne permettait pas l'affichage de la date normalisée pour les types GS1-Datamatrix. Report dans le champ UDI et GTIN des identifiants EAN contenus dans les codes de type NaS. Désactivation de l'analyse du code NaS 014.
23/11/2018	Version 2.2.2.0 Gestion séparateur longueur variable quand il est codé comme [GS] Retour du GS1 AI 90 INTERNAL Désactivation des SubTypes suivantes : NaS - "006"; // COUSIN BIOSERV Company NaS - "012"; // SEM (Sciences Et Medecine) NaS - "013"; // ABS BOLTON Company Correction dans le xsd des erreurs signalées par Guillaume Lefebvre (HUS) Retour de la désignation pour le SymbologyID selon le standard ISO/IEC CD 15424
21/12/2020	Version 2.6.7656.27330 Désactivation du Subtype NaS 001 Ajout du champ <i>UDI_DI</i> alimenté avec les codes GTIN ou UPN.

	<p>Ajout du champ <i>Issuer</i> alimenté avec la valeur « GS1 » si le champ <i>UDI_DI</i> contient un code GTIN ou la valeur « HIBCC » si le champ <i>UDI_DI</i> contient un code UPN.</p> <p>Création du schéma XSD Barcodestore.0.1.6.xsd</p> <p>Mise à jour du dépôt Jeu d'Essai et Vérification Barcodestore_master_20201217151559.xml</p> <p>Les codes ACL ou CIP ne sont plus transmis dans le champ <i>GTIN</i><sup>1</sup>.</p> <p>Les codes ACL ou CIP capturés à partir de codes à barres GS1-128 ou GS1-DataMatrix sont convertis sur 13 caractères dans les champs ACL/CIP, UDI et Identifiers (suppression du caractère zéro « 0 » à gauche).</p> <p>Les codes GTIN capturés à partir de codes à barres EAN 13 ou NaS Subtype 004 sont convertis sur 14 caractères dans les champs GTIN, UDI, UDI_DI et Identifiers (padding gauche avec caractère zéro « 0 »).</p> <p>Le champ <i>UDI</i> est devenu obsolète, son usage n'est plus recommandé<sup>2</sup>.</p>
13/05/2022	<p>Version 2.8</p> <p>Le type NaS SubType NaS ne retourne plus le champ Reference donc la valeur n'alimente plus la liste Identifiers.</p>
06/07/2023	<p>Version 2.10</p> <p>Implémentation de l'algorithme de calcul du siècle pour les dates dont l'année est sur deux caractères.</p>
16/08/2023	<p>Version 2.12</p> <p>Correction de la méthode qui calcule la date normalisée.</p>
12/01/2024	<p>Version 2.14</p> <p>Prise en charge de la chaîne « \u001d » en tant que séparateur pour les données à longueur variable dans les codes à barres de type GS1-128 et GS1-Datamatrix.</p>
02/05/2024	<p>Version 2.16</p> <p>Prise en charge de la quantité dans un code HIBC lorsque cette donnée est transmise dans le complément de structure Additional Supplemental Data.</p> <p>Correction d'un bug concernant la normalisation de la date dans une structure HIBC Secondary \$\$+.3 et \$\$+.8.3</p>

<sup>1</sup> Cette correction est justifiée par le fait que les codes ACL et CIP ne sont pas considérés comme des GTIN (Global Trade Item Number) mais plutôt comme des NTIN (National Trade Item Number). Cf. GS1 General Specifications Jan 2020

<sup>2</sup> Le nom de ce champ (« UDI ») ne correspond pas à la réalité des identifiants qui y sont reportés.

## Sommaire

1. Présentation du Projet ParserIO .....	5
2. Type du Code .....	5
3. SubType du Code .....	6
4. Les fonctionnalités non couvertes .....	7
5. ParserIO_functions.....	7
6. La console Windows.....	9
7. ParserIO en web-service .....	11
8. Exemple de client du ParserIO en web-service.....	11
9. ParserIO visible en tant qu'objet COM .....	12
10. Le Jeu d'essai.....	12
11. ParserIO Benchmark .....	12
12. ParserIO Benchmark Delta.....	13
13. Droit d'utilisation .....	13
14. Téléchargement .....	13

## 1. PRESENTATION DU PROJET PARSEIO

Les industriels utilisent presque toujours les technologies des codes à barres. Ces codes à barres sont conçus et gérés par et pour les industriels, autour des cas d'utilisation relatifs à la production et à la logistique.

Les codes à barres que l'on peut trouver sur le terrain respectent souvent un standard. Ils peuvent également être propres au fournisseur.

Le code à barres proprement dit est une représentation graphique d'une chaîne de caractères exprimant l'information. Les lecteurs laser présents sur le marché savent interpréter le code à barres et restituer la chaîne de caractères correspondante.

ParserIO est un analyseur syntaxique. Il prend en entrée la chaîne de caractères, l'analyse et retourne les informations contenues sous une forme structurée et exploitable.

ParserIO n'effectue aucun autre traitement. Tout au plus transforme-t-il la date pour la publier dans un format normalisé.

ParserIO dispose de deux qualités :

- La capacité d'interpréter le plus grand nombre de codes-barres,
- La portabilité pour une intégration et une exploitation dans des environnements les plus divers.

## 2. TYPE DU CODE

L'analyse syntaxique du code permet d'identifier son type et sa structure. Le type d'un code est lié à son standard.

Voici la table de correspondance entre les standards supportés et les types :

Standard	Type
<b>European Article Numbering</b>	EAN 13
<b>Health Industry Bar Code</b>	HIBC
<b>Global System One</b>	GS1-128 ou GS1-Datamatrix
<b>Not a Standard</b>	NaS

Les codes à barres, standards ou non, qui ne sont pas reconnus, sont considérés comme des codes du Type NaS, contenant une référence produit du fournisseur identique à la chaîne complète.

### 3. SUBTYPE DU CODE

Une analyse correcte d'un code nécessite parfois d'identifier son SubType. En fait, il s'agit d'identifier la « Variante » du Type. Pour comprendre la signification des valeurs SubType, il est nécessaire d'identifier le standard lui-même.

Voici une liste non exhaustive des valeurs possibles pour un SubType en fonction du standard.

#### EAN 13

SubType	Description
<b>« » (Chaîne vide)</b>	Structure EAN 13 classique. Le code est dans ce cas-là retourné en tant que GTIN.
<b>ACL 13</b>	Variante utilisée pour coder un code ACL à 13 caractères dans une structure EAN 13. Pour toute information complémentaire, le lecteur est invité à utiliser le Cahier n°3 du CIP/ACL disponible sur le site web du CIP/ACL. Pour en savoir plus consulter le cahier téléchargeable ici : *
<b>CIP 13</b>	Variante utilisée pour coder un code CIP à 13 caractères dans une structure EAN 13

\* : <http://www.ucdcip.org/pdf/CIP-ACL%20cahier%20n%C2%B03%20codification%20marquage%20produits%20sante.pdf>

#### HIBC

SubType	Description
<b>Primary</b>	Variante Primary du Standard HIBC
<b>Secondary</b>	Variante Secondary du Standard HIBC
<b>Primary/Secondary</b>	Variante qui concatène un code Primary avec un code Secondary

Un SubType Secondary peut contenir plusieurs sous-variantes.

## GS1-128 et GS1-Datamatrix

L'analyseur ParserIO est capable d'interpréter toute chaîne de caractères dont la structure est de type GS1-128 et GS1-Datamatrix. Si un AI (Application Identifier) n'est pas géré c'est parce qu'il n'a jamais été trouvé sur le terrain. Seules les AIs trouvés sur le terrain ont fait objet d'un développement spécifique.

## NaS

SubType	Description
<b>NaS</b>	Aucune interprétation possible de la chaîne
<b>001</b>	<del>EAN 13 and LPP sans clé de contrôle (Obsolète)</del>
<b>002</b>	ACL 13 and LPP
<b>003</b>	ACL 13 and LPP avec espace
<b>004</b>	EAN 13 and LPP
<b>005</b>	Chris Eyes Company – Référence, Lot, Date d'expiration
<b>006</b>	<del>COUSIN BIOSERV Company – Référence, Lot (Obsolète)</del>
<b>007</b>	<del>BARD France Company – Référence, Lot, Date d'expiration (Obsolète)</del>
<b>008</b>	<del>PHYSIOL France Company – Référence, Lot, Serial (Obsolète)</del>
<b>009</b>	<del>Arthrex Company – Référence (Obsolète)</del>
<b>010</b>	<del>Arthrex Company – Lot (Obsolète)</del>
<b>011</b>	<del>Arthrex Company – Quantité (Obsolète)</del>
<b>012</b>	<del>SEM (Sciences Et Medecine) Company – Référence, Lot</del>
<b>013</b>	<del>ABS BOLTON Company – Référence</del>
<b>014</b>	<del>CHIRURGIE OUEST / EUROSILICONE / SORMED Company – Référence (Obsolète)</del>
<b>015</b>	<del>Symbios Orthopédie – Référence, Date d'expiration (Obsolète)</del>
<b>016</b>	<del>Teleflex / Arrow – Référence, Lot, Date d'expiration (Obsolète)</del>
<b>017</b>	<del>FCL – Lot, Date d'expiration, Ind (pas traité, comme ADD.ID AI 240) (Obsolète)</del>

## 4. LES FONCTIONNALITES NON COUVERTES

Les fonctionnalités effectivement rencontrées sur le terrain mais non développées dans cette version de ParserIO sont listées ci-après.

Aucune.

## 5. PARSEIO.CORE

Le noyau du projet est l'espace de noms ParserIO.Core qui contient la classe Functions et l'interface IFunctions.

La classe est écrite en langage C# en s'appuyant sur Microsoft .NET Framework, le tout est disponible sous forme de projet Microsoft Visual Studio 2015.

La méthode `public InformationSet GetFullInformationSet(string code)` prend en entrée le code et retourne un objet contenant toutes les informations.

## 6. LE PATTERN

L'objectif est d'extraire l'information contenue, ou les informations contenues, dans une chaîne de caractères. La chaîne de caractères peut contenir une seule information, une référence de produit ou une date d'expiration par exemple, ou elle peut en contenir plusieurs voire des dizaines.

L'organisation des données dans la chaîne dépend du standard à laquelle elle se voit conforme. Au départ de l'analyse ParserIO prend en entrée la chaîne mais il ne sait pas de quelle structure il s'agit. C'est donc une reconnaissance préalable de la structure qui permettra ensuite l'extraction des données.

La particularité de l'identification de la structure préalable à l'extraction des données c'est le fait que l'on doit parfois juste « explorer » les données pour que l'identification de la structure puisse se faire. Dans certains cas la présence de certains caractères permet une identification plus robuste et moins complexe.

La structure d'une chaîne peut être identifiée à travers deux valeurs :

- Type
- SubType

Dans l'ordre l'on identifie d'abord le Type, ensuite le SubType.

Une fois que le Type et le SubType sont connus, le ParserIO peut procéder à l'extraction des données contenues dans la chaîne.

Dans l'implémentation actuelle, les méthodes s'appellent avec la désignation de la donnée. Donc une méthode prend toujours en entrée la chaîne.

Concernant l'adhérence du fonctionnement de ParserIO aux différences structures existantes actuellement reconnues, voici un aperçu des documents standards, par organisation en charge de leur maintenance :

### AIM

L'organisation « The Association for Automatic Identification and Data Capture Technologies » s'occupe du standard ITS 0X-2010 Data Carrier/Symbology Identifiers Maintenance Document. Ceci pour le compte de l'organisation ISO. Le standard prend donc le nom de ISO/IEC 15424. C'est le standard qui nous permet, dans le ParserIO, d'implémenter la méthode SymbologyID. La diffusion du standard est interdite. Il est disponible sous paiement auprès de l'ISO.

Le paramétrage du lecteur du symbole code à barres peut modifier le résultat de la lecture. Certains dispositifs ne permettent pas un certain nombre de paramétrages, notamment la génération des identifiants de symbologie définis dans le standard ISO/IEC 15424.

### HIBCC

L'organisation « Heath Industry Business Communications Council » est en charge de l'évolution du standard ANSI/HIBC « THE HEALTH INDUSTRY BAR CODE (HIBC) SUPPLIER LABELING STANDARD ». Pour les codes conformes à cette structure, le ParserIO attribue le Type HIBC. Les différentes SubType possibles pour le Type HIBC sont disponibles dans le standard :



- Primary
- Secondary

Le SubType Primary est défini par intention dans la *Table 1*.

Le SubType Secondary est défini par intention dans la *Table 2*. Par ailleurs, le SubType est défini par extension dans la *Table F1*.

Si un Data Identifier (DI) dans le Secondary est présent et pas encore géré, il sera retourné dans le champ Additional Information. Si plusieurs DI sont présents, ils seront séparés par un point-virgule.

## GS1

L'organisation « Global System One » maintient de nombreux standards. Celui qui intéresse les codes à barres est le « GS1 General Specifications ».

Les structures GS1-128 et GS1-DataMatrix autorisent la concaténation de plusieurs données dans une même chaîne de caractères. Chaque donnée est précédée par une balise appelée Application Identifier (AI) dans le langage GS1. Un AI est toujours numérique et sa longueur est de 2 caractères au moins. A titre d'exemple l'AI 01 précède un code GTIN, l'AI 10 précède un numéro de lot.

Pour les codes à barres conformes à cette structure, ParserIO attribue soit le Type GS1-128 soit le Type GS1-Datamatrix. Ce choix dépend, quand l'information est présente, de la valeur du SymbologyID.

Pour ce type de codes à barres, ParserIO alimente le Subtype en agrégeant les différents AI séparés par des points (et en suivant l'ordre de leur encodage). A titre d'exemple, une structure de Type GS1-128 (ou GS1-Datamatrix) qui contient seulement un code GTIN et un numéro de lot aura « 01.10 » comme valeur de SubType. Le Subtype est traité indifféremment pour les deux types GS1-128 et GS1-DataMatrix. Enfin, les règles d'alimentation du SubType sont propres au ParserIO (non définies dans les standards GS1).

Les structures de type GS1-128 ou GS1-Datamatrix exigent l'encodage de séparateurs de champs pour marquer la séparation entre la fin d'un champ de longueur variable (e.g. numéro de lot, AI 10) et le début d'un autre champ (e.g. numéro de série, AI 21). Le lecteur qui scanne ce type de code à barres doit pouvoir transmettre le séparateur dans la chaîne de caractères pour que les données soient ensuite correctement analysées par ParserIO.

ParserIO reconnaît différents types de séparateurs :

- Caractère non imprimable <GS> (ASCII 29 décimal, 1D hexadécimal)
- Caractères imprimables « @ », « [GS] » et « \u001d »

Les caractères imprimables sont utilisables seulement lorsque ParserIO est utilisé en mode Webservice.

Si un AI n'est pas géré, toute la chaîne résiduelle à sa droite sera retournée dans le champ Additional Information.

## ANSI MHI

L'organisation ANSI, American National Standard Institute, publie différents standards maintenus par le MHI, Material Standard Industry. Un de ces standards est l'ANSI MH10.8.2 dont le titre est « Data Identifier and Application Identifier Standard ». La connaissance de ce standard est à la base des évolutions de la version 2.5 du standard HIBC.

## 7. LA CONSOLE WINDOWS

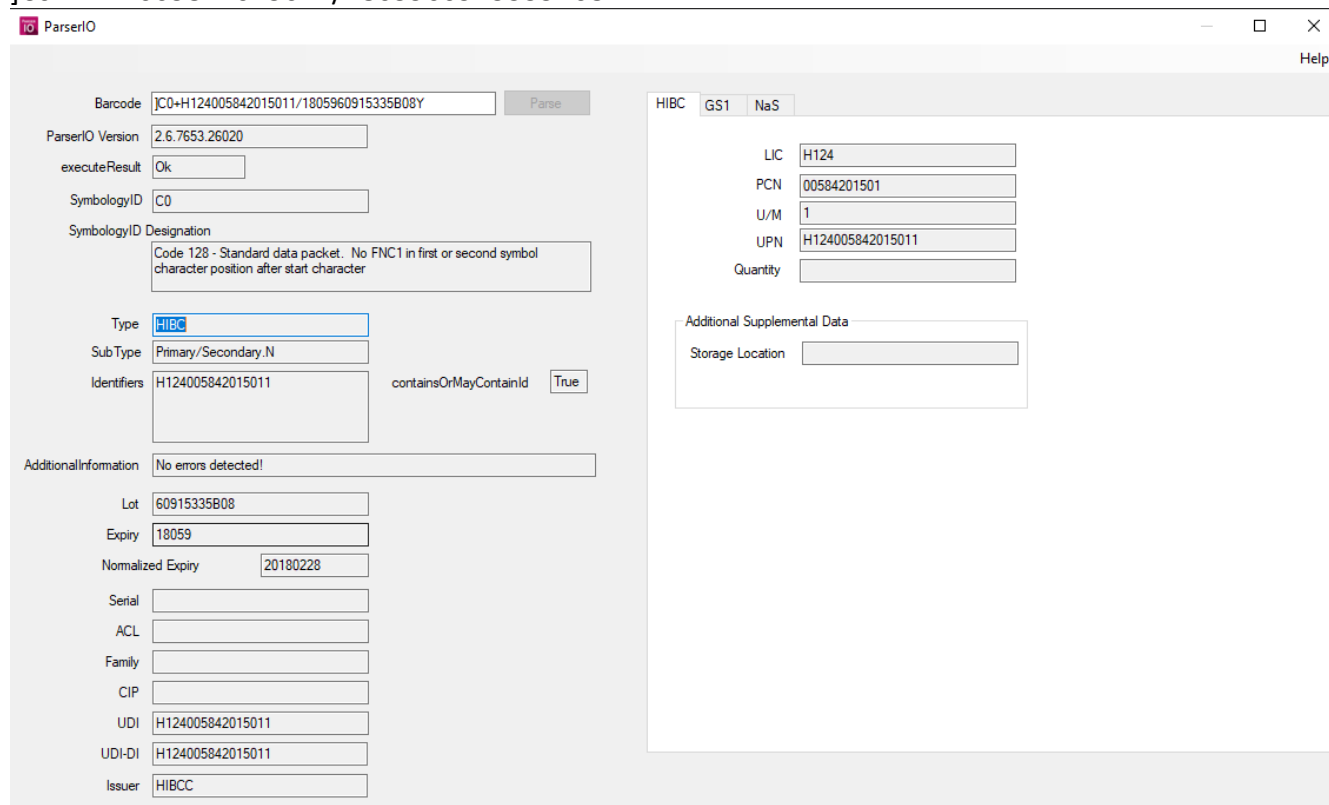
La console Windows est un programme de type Windows Forms Application utilisant Microsoft .NET Framework 4 Client Profile. Il s'agit d'un simple consommateur des méthodes mises à disposition par la classe Functions.

Elle est disponible sous forme de projet Microsoft Visual Studio 2015.

La console est mono code. Cela veut dire qu'elle prend en entrée un seul code à la fois.

Voici une capture d'écran de la console Windows en utilisant le code suivant :

`]C0+H124005842015011/1805960915335B08Y`



The screenshot shows the ParserIO application window. The main area displays the results of parsing the barcode `]C0+H124005842015011/1805960915335B08Y`. The interface includes a 'Barcode' input field, a 'Parse' button, and a 'ParserIO Version' field showing '2.6.7653.26020'. The 'executeResult' is 'Ok'. The 'SymbolologyID' is 'C0'. The 'SymbolologyID Designation' is 'Code 128 - Standard data packet. No FNC1 in first or second symbol character position after start character'. The 'Type' is 'HIBC'. The 'Sub-Type' is 'Primary/Secondary.N'. The 'Identifiers' field contains 'H124005842015011' and the 'containsOrMayContainId' checkbox is checked. The 'AdditionalInformation' field shows 'No errors detected!'. The 'Lot' is '60915335B08', 'Expiry' is '18059', and 'Normalized Expiry' is '20180228'. The 'Serial', 'ACL', 'Family', 'CIP', 'UDI', 'UDI-DI', and 'Issuer' fields are also present. On the right, there is a 'HIBC' tab with fields for 'LIC' (H124), 'PCN' (00584201501), 'U/M' (1), 'UPN' (H124005842015011), and 'Quantity'. Below this is a section for 'Additional Supplemental Data' with a 'Storage Location' field.

## 8. PARSERIO EN SERVICE

ParserIO est également disponible en mode service dans un Webservice SOAP service maintenu par Phast. La description du service est disponible à cette adresse :

<https://services.phast.fr/ParserIO.LegacyWS/parserio.asmx>

## 9. EXEMPLE DE CLIENT DU PARSERIO EN WEB-SERVICE

Phast met à disposition un exemple de client consommateur de ParserIO en Web-Service. Un client pour chaque service.

## 10. PARSEIO VISIBLE EN TANT QU'OBJET COM

L'interface IFunctions est visible en tant qu'objet COM.

Phast met à disposition un exemple de client consommateur de l'objet COM. Le client est écrit avec le langage C++ et est téléchargeable avec l'ensemble de la solution. Il peut faciliter le travail des éditeurs qui souhaiteraient utiliser en local les méthodes de l'interface IFunctions dans un environnement différent de celui utilisé pour le développement. La bibliothèque doit être enregistrée. Afin de toujours utiliser la dernière version de l'interface, le projet ParserIO.Core contient deux commandes dans le « Post build event command line » :

```
"%Windir%\Microsoft.NET\Framework\v4.0.30319\regasm" $(TargetPath) /u  
"%Windir%\Microsoft.NET\Framework\v4.0.30319\regasm" $(TargetPath) /codebase  
/tlb:$(SolutionDir)ParserIO_COMclient\$(TargetName).tlb
```

Dans la version actuelle du projet, cette interface est disponible seulement pour la version 1.0, l'ancienne, de la classe *Functions*. Prochainement elle sera également disponible dans la version par défaut.

## 11. LE JEU D'ESSAI

Phast met à disposition un **jeu d'essai**. Chaque élément du jeu d'essai est constitué d'un code à barres sous forme d'une chaîne de caractères ainsi que du résultat de son analyse. Le jeu d'essai se présente comme un fichier au format XML et son nom est « **Barcodestore\_master\_TimeStamp.xml** », où TimeStamp est la date et l'heure de génération. Il est également validé par un schéma XDS et son nom est « **Barcodestore.X.xsd** » où X est la version. Les deux fichiers sont disponibles en téléchargeant les sources du projet ParserIO.

Les codes à barres contenus dans le jeu d'essai sont issus de plusieurs campagnes de récolte sur le terrain. Les chaînes de caractères sont produites en sortie d'un lecteur monodimensionnel de code à barres.

Un paramétrage différent et/ou un lecteur différent pourrait donner un résultat différent suite à l'interprétation d'un même symbole.

L'analyse des codes à barres a été faite manuellement en respectant les règles d'interprétation données par les standards respectifs.

L'utilisateur/développeur du projet ParserIO pourra se servir du jeu d'essai au moins de deux façons :

1. Tester un analyseur de codes à barres qu'il a développé
2. Tester des modifications qu'il a faites dans le projet ParserIO

## 12. PARSEIO BENCHMARK

Phast met à disposition un **outil de test de non-régression**. L'outil s'appuie sur le jeu d'essai décrit ci-dessus et il peut consommer les méthodes de ParserIO aussi bien par DLL que par service. Le projet appelé ParserIO.Benchmark est disponible dans la solution ParserIO. L'outil prend en entrée le jeu d'essai, récupère la liste des codes à barres et fournit un fichier en sortie avec l'analyse. La structure du jeu d'essai et du fichier en sortie est identique.

## 13. PARSEIO BENCHMARK COMPARE

Une fois l'analyse terminée et le fichier en sortie créé, il s'avère parfois compliqué de faire un différentiel entre le fichier lui-même et le jeu d'essai.

Phast met à disposition un outil d'analyse du différentiel qui prend en entrée le jeu d'essai et le fichier créé par l'outil Benchmark et fournit en sortie un fichier contenant le différentiel.

Ce différentiel identifie précisément le nœud XML de l'objet « code à barres » qui porte la divergence.

Le projet appelé ParserIO.Benchmark.Compare est disponible dans la solution ParserIO.

## 14. DROIT D'UTILISATION

La solution ParserIO est livrée sous licence GNU Lesser General Public License v3.

Pour tout complément d'information sur les droits d'utilisation attachés à cette licence, nous vous conseillons de vous adresser à la page officielle de la licence :

<http://www.gnu.org/copyleft/lesser.html>

## 15. TELECHARGEMENT

Phast délivre ParserIO à travers la forge GitHub :

<https://github.com/phast-fr/ParserIO>