

SOGETI Région

485, avenue de l'Europe

38330 Mont Bonnot St-
Martin



SOGETI

Régions

«API HL7»

Document utilisateur

1.1 Validité du document.....	2
1.2 Avant-propos.....	3
1.3 L'API HL7 de Sogeti.....	4
1.4 Environnement de développement.....	5
1.5 Mise en œuvre	6
1.5.3.1 Patient.....	8
1.5.3.2 PatientLocation.....	11
1.5.3.3 MessageInterface.....	12
1.6 Interfaces tests	14
1.7 Configuration réseau.....	16

1.1 VALIDITÉ DU DOCUMENT

Historique			
Date création	Date application	V.R.	Évolutions
18/03/09		1.0	

Diffusion	
Suivie en mise à jour	Copie non suivie

Référence des documents		
Repère	Référence	Nom
	TIS3	Anthony Crouzet

1.2 AVANT-PROPOS

1.2.1 HL7

Health Level 7 (HL7) est un standard qui définit un format pour les échanges informatisés de données cliniques, financières et administratives entre systèmes d'information hospitaliers (SIH). Ce standard définit entre autre la structure et le rôle des messages pour permettre une communication efficiente des données liées au système de la santé.

1.2.2 MACHINE VIRTUELLE

L'application développée en Java, utilise une machine virtuelle Java (JVM). Les tests ont été effectués avec une JVM version 5, sous le système d'exploitation Windows XP et également sous MAC OS 10.5.

1.3 L'API HL7 DE SOGETI

L'API ¹HL7 est une interface Java d'une série d'objets conçus pour créer, envoyer et recevoir des messages HL7. La construction d'un message HL7 est aussi facile que de créer un autre objet Java.

Au jour d'aujourd'hui, la version de l'API HL7 ne gère que les messages d'admissions et de décharge.

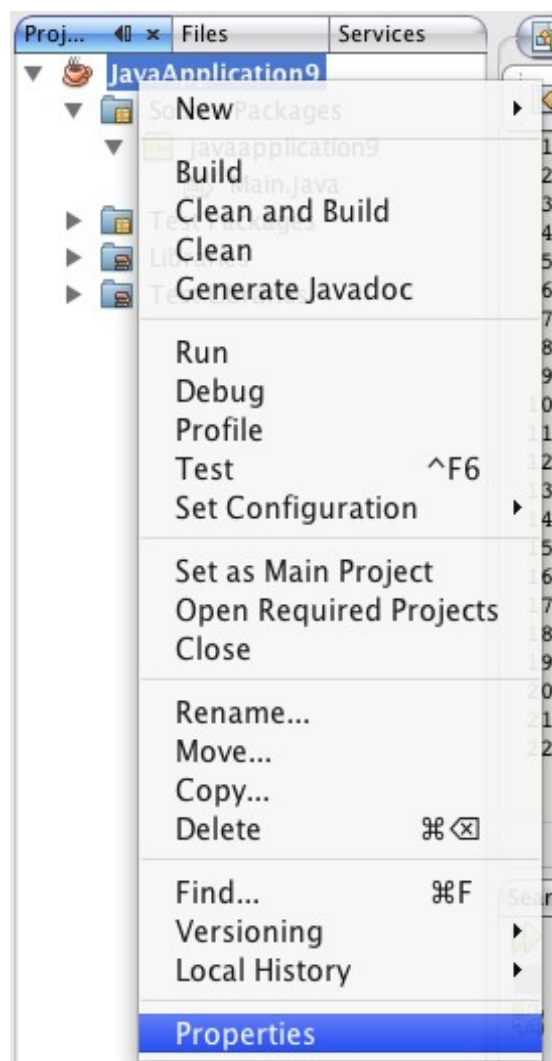
Les messages HL7 lus et générés par l'application font référence à la version 2.5 de HL7.

1 Application Programming Interface : une interface de programmation

1.4 ENVIRONNEMENT DE DÉVELOPPEMENT

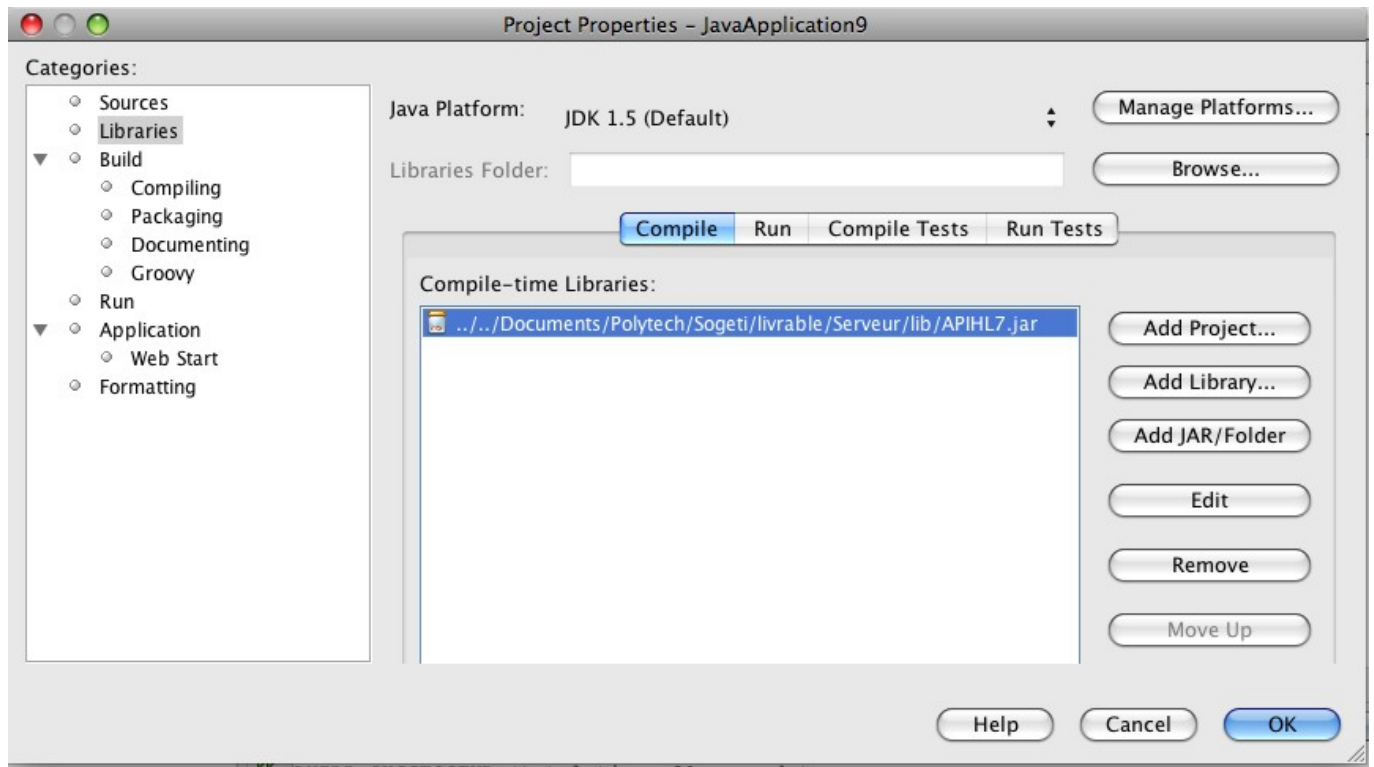
L'API HL7 est une bibliothèque logicielle qui permet de développer des applications utilisant le protocole HL7 sans se soucier de la « tuyauterie ».

Il faut donc l'ajouter à votre projet afin de pouvoir en user. Selon votre environnement de développement (IDE), la manipulation est différente. Dans le cas de NetBeans, effectuer un clic droit sur le projet, puis sélectionner **Properties**.



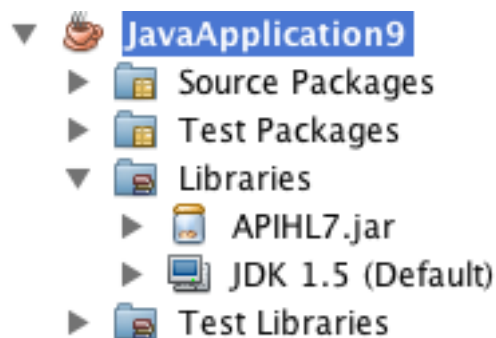
Figure

Ensuite, **Categories > Libraries > Add JAR/Folder ...** et ajouter le fichier APIHL7.jar.



Figure

Après avoir validé à l'aide du bouton **Ok**, la bibliothèque ajoutée est visible dans l'arborescence du projet.



Figure

1.5 MISE EN ŒUVRE

1.5.1 RÉCEPTION DE MESSAGE

L'application qui désire recevoir un message HL7 doit ouvrir un port de connexion. Ce port doit être disponible et consacré uniquement à l'application. Par exemple, le port numéro 4444 peut être utilisé.

Pour ouvrir un port de connexion, il faut avant tout créer un objet `ServeurHL7`, puis faire appel à la méthode `connexion(int port)`.

```
serveur = new ServeurHL7();  
serveur.connection(port);
```

La fonction `connexion` exige un argument :

`port` : un entier indiquant le numéro du port à ouvrir.

Une fois le port ouvert, il faut écouter ce dernier jusqu'à ce qu'une application envoie un message HL7 via ce port. Pour cela, la méthode `ecoute()` doit être invoquée.

```
serveur.ecoute();
```

La méthode est bloquée jusqu'à ce que la connexion soit établie.

Une fois la connexion établie entre l'API et l'application éditrice, la méthode `protocole()` peut être invoquée pour traiter le message reçu. Le message HL7 est analysé et converti en objets Java :

- Patient
- Message

De plus, la méthode renvoie à la source, un message HL7, accusé de réception.

```
String messageHL7 = serveur.  
protocole();
```

La méthode retourne le message HL7 reçu sous forme de chaîne de caractères.

La fin de la connexion est terminée par l'appel à la méthode `fermer()`.

1.5.2 ENVOI D'UN MESSAGE HL7

Avant tout envoi, le message HL7 doit être généré à partir d'objets Java. Ainsi commencer par créer un patient.

```
patient = new Patient(id, surname,
classe);
```

Seulement l'identification, la classe et le nom de famille du patient sont obligatoires. Les autres paramètres peuvent être renseignés par des *setters*.

Ensuite, une connexion doit être établie.

```
Client client = new
Client();
client.connexion (host,
port);
```

La fonction `connexion` exige deux arguments :

- `host`: l'adresse IP de la machine à contacter (`localhost` peut être utilisé en local) ;
- `port` : un entier indiquant le numéro du port à contacter.

Une fois la connexion établie un message peut être envoyé. Pour cela, il y a une méthode par type de message :

Pour un message d'admission d'un patient hospitalisé :

```
client.admit(patient);
```

Pour un message de mutation d'un patient :

```
client.transPat(patient);
```

Pour un message de sortie hôpital d'un patient (décharge) :

```
client.endPat(patient);
```

Le message d'accusé de réception s'obtient par la méthode `getMsg()` .

```
MessageInterface messageAck =
client.getMsg();
```

1.5.3 ACCESSEURS

Une fois le message HL7 reçu et analysé, deux objets Java sont créés et accessibles. Ces objets sont `Patient` et `Message`.

```
Patient getPatient();  
Message getMessage();
```

1.5.3.1 Patient

Les valeurs associées au patient sont :

- l'identification du patient ;
- la classe du patient ;
- le nom de famille du patient ;
- le prénom du patient ;
- la date de décès du patient ;
- l'avis de décès du patient ;
- la localisation assignée du patient ;
- la localisation précédente du patient ;
- la date de sortie du patient ;
- la date de naissance du patient ;
- le sexe du patient ;

Seulement l'identification, la classe et le nom de famille du patient sont obligatoires.

L'identification du patient est de type entier et accessible par la méthode `Integre getID()`.

La classe du patient est de type caractère et accessible par la méthode `char getCharPatClass()`. Mais, il est également accessible sous la forme d'une chaîne de caractère qui est plus explicite qu'un caractère, via la méthode `String getPatClass()`.

E	Emergency
I	Inpatient
O	Outpatient
P	Preadmit
R	Recurring patient
B	Obstetrics
C	Commercial Account
N	Not Applicable

U Unknown

Le nom de famille du patient est de type chaîne de caractères et accessible par la méthode `String getFamilyName()`.

Le prénom du patient est de type chaîne de caractère et accessible par la méthode `String getFirstName()`.

La date de décès du patient est de type `Date` et accessible par la méthode `Date getDeath()`. Si le patient n'est pas décédé ou la date est inconnue, alors la méthode retournera une valeur nulle.

L'avis de décès est accessible sous différents types :

- `boolean isDeath()` qui retourne un booléen.
- `String getValueDeath()` qui retourne 3 possibilités :
 - N : pour non décédé
 - Y : pour décédé
 - : caractère vide pour information non disponible

La localisation assignée au patient est de type **PatientLocation** et accessible par la méthode `PatientLocation getAssignedPatLocation()`.

La localisation antérieure du patient est de type **PatientLocation** et accessible par la méthode `PatientLocation getPriorPatLocation()`.

La date de sortie du patient est de type `Date` et accessible par la méthode `Date getDateDicharge()`. Si la date est inconnue, alors la méthode retournera une valeur nulle.

La date de naissance du patient est de type `Date` et accessible par la méthode `Date getBirth()`. Si la date est inconnue, alors la méthode retournera une valeur nulle.

Le sexe du patient est de type caractère et accessible par la méthode `char getCharSex()`.

F	Féminin
M	Masculin
O	Autre
U	Inconnu
A	Ambigüe

N Non applicable

1.5.3.2 PatientLocation

Les valeurs associées à la localisation du patient sont :

- Code de l'UF d'hébergement
- Chambre
- Lit
- Statut de la localisation
- Type de personne
- Bâtiment
- Étage
- Patient

Seulement le patient est obligatoire.

Le code de l'UF d'hébergement est de type chaîne de caractères et accessible par la méthode `String getPointOfCare()`.

La chambre est de type chaîne de caractères et accessible par la méthode `String getRoom()`.

Le lit est de type chaîne de caractères et accessible par la méthode `String getBed()`.

Le statut de localisation est de type chaîne de caractères et accessible par la méthode `String getStatus()`.

Le type de personnel est de type chaîne de caractères et accessible par la méthode `String getPersonLocationType()`. Il est non utilisé selon les recommandations IHE France

- | | |
|---|-------------------|
| C | Clinique |
| D | Département |
| H | Maison |
| N | Unité infirmière |
| O | Provider_s Office |
| P | Téléphone |

S SNF

Le bâtiment est de type chaîne de caractères et accessible par la méthode `String getBuilding()`.

L'étage est de type chaîne de caractères et accessible par la méthode `String getFloor()`.

1.5.3.3 MessageInterface

Les valeurs associées au message sont :

- Identification
- Type message
- Identifiant du message qui est accusé
- Code

Seulement l'identification et le type du message sont obligatoires.

L'identification du message est de type entier et accessible par la méthode `Integre getID()`.

Le type du message est de type chaîne de caractères et accessible par la méthode `String getType()`.

L'identification du message qui est accusé, est de type entier et accessible par la méthode `Integre getIdAck()`.

Le code du message est de type chaîne de caractère et accessible par la méthode `char getcode()`. Mais, il est également accessible avec la méthode `String getAcknowledgmentCodeString()` qui retourne un résultat plus explicite

```
AA  Application Accept
AE  Application Error
AR  Application Reject
CA  Commit Accept
CE  Commit Error
CR  Commit Reject";
```


1.6 INTERFACES TESTS

Afin de tester cette bibliothèque logicielle, deux applications de tests ont été développées. Ces dernières se nomment `ClientTest` et `ServeurTest`. L'interface graphique et les contrôles sont très rudimentaires, car le but 1^{er} était de vérifier la compatibilité des messages HL7.

1.6.1 SERVEUR

L'application `ServeurTest` permet de recevoir des messages HL7. Pour cela, elle ouvre un port de connexion et attend la réception d'un message. A la réception d'un message, ce dernier est analysé. Quelques éléments du message sont affichés dans l'interface graphique et un message d'accusé de réception est généré et envoyé à l'application émettrice du premier message.

Ci-après le message HL7 envoyé sur le port 4444 et la capture d'écran (Figure) de l'application après cette réception.

```
MSH|^~\&|||||20090318173043+0100||ADT^A01|795006462|P|2.5|||||||
EVN|A01|20090318173043+0100|||||
PID|1||245||Durant^Paul|||||||||||||||||||||N|||||||||||||||||
PV1||Emergency|
U2^405^Fenêtre^libre^C^CHU^4^|||||||||||||||||
```

Service de chirurgie cardiaque

Connexion

Port :

Déconnexion
Connexion

Admission
 Classe Patient: Emergency
 Id Patient :245
 Nom : Durant
 Prénom : Paul
 ----- Localisation -----
 Point Of Care: U2
 Bâtiment : CHU
 Étage: 4
 Chambre: 405
 Lit: Fenêtre
 Status: libre
 Type: C
 ----- Ancienne localisation -----

Figure

Le message d'accusé de réception généré par cette application est le suivant

```
MSH|^~\&|||20090318173043+0100||ACK^A01|772306277|P|2.5
MSA|AA|795006462
```

1.6.2 CLIENT

Du côté client, l'application `ClientTest` permet de générer un message HL7, de l'envoyer puis de recevoir un message d'accusé de réception.

Accueil CHU

Action : Admission d'un patient hospitalisé

Patient

Catégorie : Urgence

Identification :

Nom :

Prénom :

Date de naissance :

Sexe : Autre

Non décédé

Date :

Date d'admission :

Date sortie :

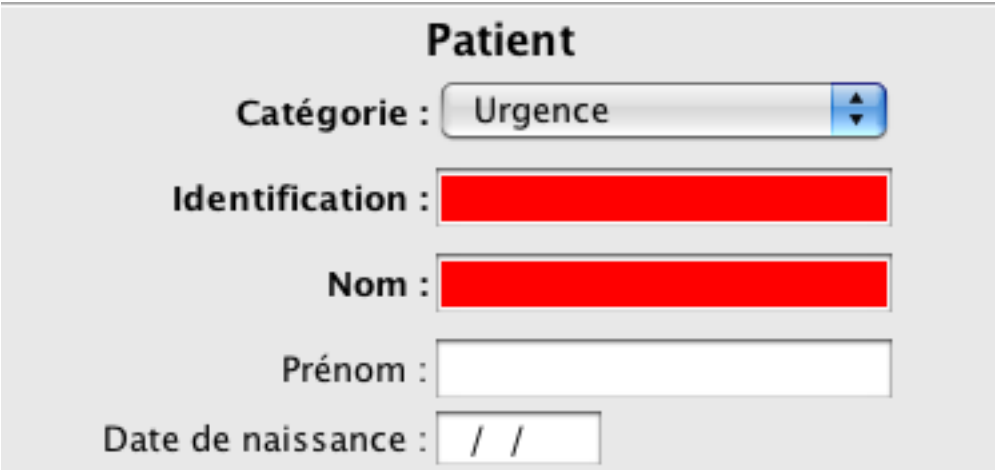
Admettre

Figure

Le menu déroulant **Action** permet de choisir le type de message :

- admission d'un patient (ADT A01)
- transfère de patient (ADT A02)
- une fin de séjour du patient (ADT A03)

Des champs sont obligatoires pour la composition des messages et ils sont mis en rouge (Figure) si l'utilisateur omet de les remplir.



Patient

Catégorie :

Identification :

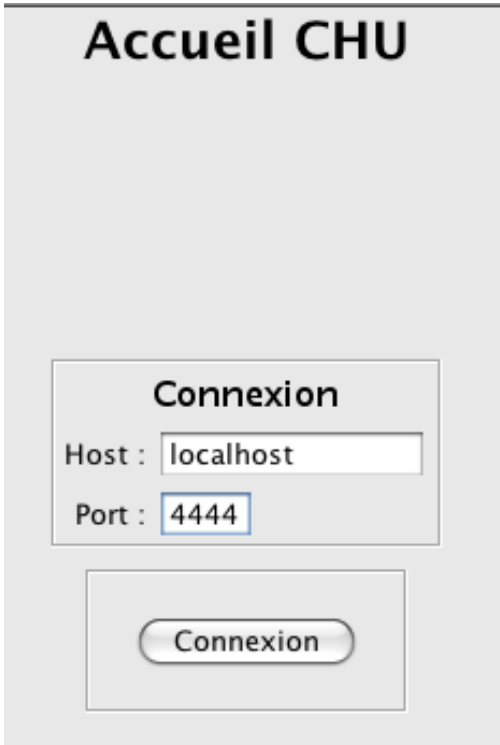
Nom :

Prénom :

Date de naissance :

Figure

Le champ *Host* doit contenir l'adresse de la machine. Le champ *port* doit contenir le numéro du port qui a été ouvert par le serveur.



Accueil CHU

Connexion

Host :

Port :

Figure

1.7 CONFIGURATION RÉSEAU

1.7.1 LOCAL

Utiliser les deux applications tests en local signifie exécuter les applications sur la même machine physique. Pour cela, le champ *host* doit contenir *localhost*.

1.7.2 CÂBLE CROISÉ

Quand deux machines sont connectées entre elles via un câble croisé, le système d'exploitation doit être en adresse statique. Par ailleurs, le pare-feu doit autoriser d'utiliser le port renseigné par l'application serveur.

Vous pouvez par exemple utiliser l'adresse IP 192.168.0.1 pour la machine A et 192.168.0.2 pour la machine B.

Propriétés de Protocole Internet (TCP/IP) [?] [X]

Général

Les paramètres IP peuvent être déterminés automatiquement si votre réseau le permet. Sinon, vous devez demander les paramètres IP appropriés à votre administrateur réseau.

☐ Obtenir une adresse IP automatiquement
☒ Utiliser l'adresse IP suivante :

Adresse IP :
 Masque de sous-réseau :
 Passerelle par défaut :

☐ Obtenir les adresses des serveurs DNS automatiquement
☒ Utiliser l'adresse de serveur DNS suivante :

Serveur DNS préféré :
 Serveur DNS auxiliaire :

Avancé...

OK Annuler

Figure