TumoroteK

Logiciel de gestion des collections de prélèvements biologiques

Documentation des interfaçages avec les Systèmes de Gestion de Laboratoires

Réf :	TumoroteK_interfaçage_SGL.doc	
Date :	26/07/2015	
Version :	2.10.6	
Nombre de pages :	11	
Création :	Auteur :	Mathieu BARTHELEMY
	Modifié :	
	Date :	

Sommaire

Sommaire

1)Présentation générale

TumoroteK est une application web n-tiers de gestion de collections biologiques. L'application propose au travers d'un navigateur client une série de formulaires permettant aux utilisateurs de manipuler le contenu d'une base de données.

Cette base de données peut également être alimentée par des interfaçages avec d'autres applications comme les Systèmes de Gestion de Laboratoires (SGL). L'objectif des interfaçages est la réduction de la double-saisie, entrainant l'amélioration de la qualité des données et un gain de temps. L'activité principale de **TumoroteK** est la gestion de collections anatomo-pathologiques; les SGLs concernés actuellement par ces interfaçages sont DIAMIC® (Infologic-Santé) et APIX® (Technidata).

2)Périmètre fonctionnel

L'interfaçage de **TumoroteK** avec un SGL consiste en **une configuration spécifique de l'application** permettant de définir:

- le protocole et le support de la communication avec le SGL en réception
- le comportement du moteur d'intégration des données reçues et envoyées (gestion des archivage, des erreurs)
- la sélection des données à exploiter, et leur association avec les champs d'informations **TumoroteK**
- le protocole et le support de la communication avec le SGL en émission de l'accusé de réception après intégration des données (DIAMIC® uniquement).

L'interfaçage implique un **champ d'information pivot** qui permet de lier une demande d'examen dans un SGL à un dossier prélèvement dans **TumoroteK**. Ce champ d'information est par défaut le *code prélèvement* dans TumoroteK et le *numéro de demande/dossier* dans le SGL.

Les données reçues depuis un SGL sont, si correctement exploitables, sélectionnées, associées aux champs d'informations **TumoroteK** puis enregistrées dans une base de données temporaire.

Si aucune liaison n'a pu être établie sur le champ d'information pivot, les données sont mises à disposition dans l'interface graphique, pour pré-remplir les formulaires lors de la création d'un nouveau prélèvement dans **TumoroteK**.

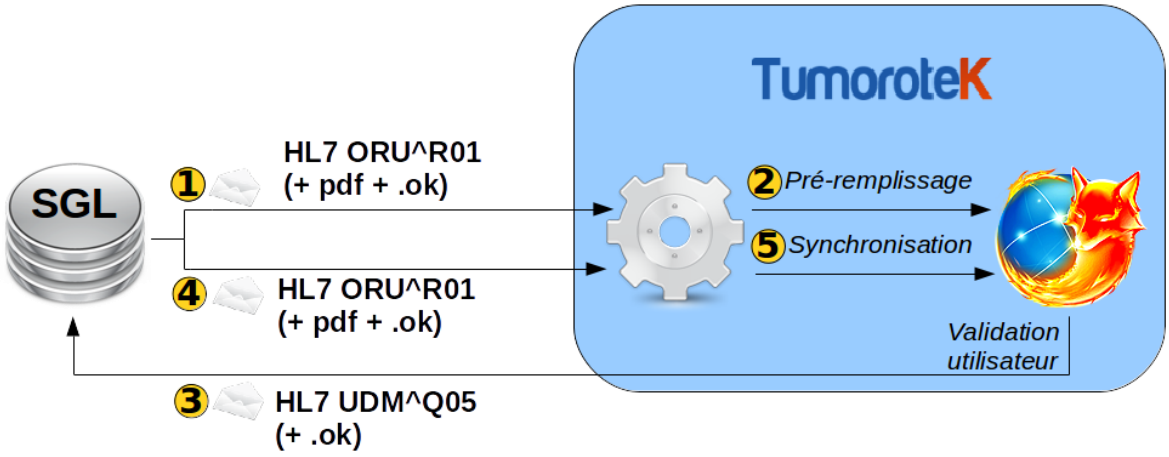
Sinon, la réception est notifiée dans l'interface graphique pour les prélèvements liés par le champ d'information pivot. L'utilisateur peut alors déclencher la comparaison et la synchronisation manuelle des informations dans **TumoroteK** à celles nouvellement envoyées par le SGL.

L'envoi de l'accusé de réception à DIAMIC® après intégration des données dans **TumoroteK** est déclenché par la création du nouveau prélèvement. L'accusé de réception contient l'URL qui permettra à DIAMIC® de fournir un lien HTTP vers la ressource *prélèvement* **TumoroteK** dans son interface graphique.

3)Diagramme de communication

Le support de transmission des données est le language HL7 version 2.5.

Le protocole TCP est proposé pour la couche de transport. Les protocoles de plus haut niveau utilisables sont à définir sur chaque installation pour effectuer l'échange de données sous la forme de fichiers (FTP, partage SMB...).



(1) Le SGL envoie, suivant un ou plusieurs évènements déclencheurs définis par l'éditeur, un message ORU contenant les informations exploitables par **TumoroteK,** et dont le segment ORC mentionne le champ d'information pivot de la communication.

(2) **TumoroteK** pré-remplit son interface avec les données récupérées depuis le SGL et propose à l'utilisateur de compléter l'enregistrement.

(3) DIAMIC®: l'utilisateur demande l'enregistrement des informations dans **TumoroteK**. Si les données sont validées par le système, **TumoroteK** envoie un message *UDM* à ***DIAMIC*** contenant l'adresse URL à laquelle la page prélèvement de **TumoroteK** correspondant au dossier ***DIAMIC*** sera accessible par un navigateur.

(4) Idem (1)

(5) L'utilisateur est notifié de la réception de nouvelles informations en provenance du SGL pour le prélèvement associé au champ d'information pivot. Il peut alors déclencher la synchronisation des données.

Le dépôt des messages HL7 peut être suivi du dépôt d'un fichier .ok, utilisé comme signal du dépôt effectif du message et donc de sa disponibilité pour consommation.

Dans le cas de DIAMIC®, la transmission des données peut contenir un pdf associé (le compte-rendu anatomopathologique), qui sera copié dans le système de fichiers de **TumoroteK** en référence au champ d'information *échantillon* correspondant.

4)Fiches de connexion

Voir la fiche de connexion détaillée établie pour chaque SGL, en collaboration avec l'éditeur du logiciel et le comité utilisateur de **TumoroteK**.

5)Environnement logiciel

Le cœur de l'application est composé de librairies codées en JAVA (frameworks). Les librairies JAVA sont servies par le serveur d'application, Apache Tomcat.

Note: Dans la suite de ce document, le chemin absolu du dossier d'installation du serveur Apache Tomcat est représenté par <TOMCAT>.

La composante interfaçage de **TumoroteK** est confiée au framework d'intégration **Apache Camel** (http://camel.apache.org/).

Le fonctionnement du framework **Apache Camel** repose sur la définition de *routes*, correspondant à une suite de *composants* paramétrables qui sont responsables d'un traitement spécifique des données. Les *routes* de traitement de la communication entre **TumoroteK** et les SGLs sont déjà entièrement définies, les différents fichiers de configuration permettent d'obtenir le démarrage, la spécificité technique et fonctionnelle d'un interfaçage.

6)

7)Marche/Arrêt du framework d'intégration

Apache Camel est embarqué dans l'applicatif **TumoroteK**. Sa prise en charge au démarrage de **TumoroteK** est conditionnée à la présence la ligne **classpath:camel-config.xml** non commentée dans le fichier de configuration <TOMCAT>/webapps/tumo2/WEB-INF/web.xml:

….

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                classpath:applicationContextDaoBase.xml
                classpath:applicationContextManagerBase.xml
                classpath:applicationContext-security.xml
                <!-- classpath:applicationContext-securityCas.xml -->
                classpath:camel-config.xml
        </param-value>
</context-param>
```

….

Le démarrage et l'arrêt de **TumoroteK** (et par extension du serveur Apache Tomcat) est nécessaire et suffisant pour démarrer et arrêter le framework d'intégration Apache Camel.

Le démarrage framework d'intégration Apache Camel est tracé dans le fichier de logs <TOMCAT>/logs/catalina.out ou <TOMCAT>/logs/catalina-stderr sous la forme:

```
INFO: Route: sgl1 started and consuming from:
Endpoint[file://c:/Users/TK.TKV2/Desktop/DIAMIC?doneFileName=%24%7Bfile%3Aname.noext%7D.ok&include
=.*%5C.hl7&noop=true]
INFO: Route: ack started and consuming from: Endpoint[direct://ack-sgl]
```

8)

9)Fichiers de configurations

10)a) Apache Camel

Les fichiers de configuration utilisés par le framework Apache Camel sont localisés par défaut dans le dossier **<TOMCAT>/conf/Catalina/localhost/camel**.

L'emplacement de ce dossier peut être modifié, et spécifié sous la forme d'une variable JNDI dans **<TOMCAT>/conf/Catalina/localhost/tumo2.xml:**

```
<Environment name="/interfaçage/conf/location"
        type="java.lang.String"
        value="/home/mathieu/apache-tomcat-7.0.47/conf/Catalina/localhost/camel/"
        override="false"/>
```

tumo2.xml est le fichier descripteur utilisé par Apache Tomcat et contient les variables d'environnement de l'application (connexions aux bases de données, chemins vers d'autres fichiers de configuration. **Ce fichier ne peut être supprimé ni déplacé en aucun cas.**

Contenu du dossier camel :

startups.properties : définit les variables interprétées par Apache Camel pour démarrer les *routes* de traitement de messages. Les deux variables doivent être définie à *true* pour démarrer les *routes* d'intégration et d'émission de messages vers le SGL.

```
sgl.startup=true
ack-sgl.startup=true
```

sgl.properties: définit le *composant* de réception des messages provenant du SGL (sgl.startpoint), et le *composant* de redirection des messages dont le traitement a produit une erreur (sgl.deadLetter).

Exemple:

```
sgl.startpoint=file:/home/mathieu2/Documents/tumorotek/interfaçages/apix_hl7_210?noop=true&include=.*\\.hl7
sgl.deadLetter=file:/home/mathieu2/Bureau/URD-deads
```

Les types des deux *composants* correspondent au protocle de communication de plus haut niveau utilisé (ici file = dépôt de fichier). Différentes options peuvent être spécifiées dans la définition des *composants*, selon leur type (voir la document officielle de Apache Camel).

Pour le *composant* de réception sgl.startpoint, les options les plus souvent utilisées pour un *composant* de type file ou ftp sont:

- noop=true (en TEST uniquement, ne supprime ni ne déplace les fichiers consommés)
- move=path (déplace les fichiers consommés dans le dossier spécifié par le path, par défaut le déplacement se fait dans le dossier local .camel)

- readLock=changed (vérifie toutes les secondes la taille du fichier et sa date de dernière modification afin de s'assurer que le fichier est entièrement déposé)
- doneFileName=${file:name.noext}.ok (se base sur l'existence d'un fichier indiquant que le message a bien été entièrement déposé. Ex: .ok)
- delete=true (supprime le fichier du disque après sa consommation)
- include=.*\\.hl7 (indique au composant de n'inclure dans la consommation que les fichiers dont l'extension est spécifiée ex: hl7).

ack.properties : définit le *composant* d'émission des messages provenant du SGL (ack-sgl.endpoint), et le *composant* de redirection des messages dont la production a levé une erreur (ack.deadLetter).

Exemple:

```
ack-sgl.endpoint=file:/home/mathieu/Desktop/URD
ack.deadLetter=file:/home/mathieu/Bureau/URD-deads
```

Les types et les options des deux *composants* suivent la même logique que celle décrite pour le fichier sgl.properties, se référer à la documentation officielle du framework Apache Camel.

11)b) TumoroteK

Le premier fichier de configuration utilisé par **TumoroteK** pour associer les SGLs émetteurs des fichiers aux groupes de collections définis **TumoroteK**, et déclarer le fichier XML d'association des champs informations est **<TOMCAT>/conf/Catalina/localhost/tumo2.properties**.

L'emplacement de ce fichier peut être modifié, et spécifié sous la forme d'une variable JNDI dans **<TOMCAT>/conf/Catalina/localhost/tumo2.xml:**

```
<Environment name="/tk/tkTumoPropertiesSystem"
        override="false"
        type="java.lang.String"
        value="/home/mathieu/apache-tomcat-        7.0.47/conf/Catalina/localhost/"/>
```

Rappel: **tumo2.xml** est le fichier descripteur utilisé par Apache Tomcat et contient les variables d'environnement de l'application (connections aux bases de données, chemins vers d'autres fichiers de configuration. **Ce fichier ne peut être supprimé ni déplacé en aucun cas.**

Les deux variables à définir sont:

```
INTERFACAGES = 1:1
INTERFACAGES_INBOXES=C:/Program Files/Apache Software Foundation/Tomcat
7.0/conf/Catalina/localhost/sgl/inboxes.xml
```

La variable INTERFACAGES définit, lors de la mise à disposition des données transmises par l'interface graphique, l'association entre le groupe de collections défini dans **TumoroteK** et

l'émetteur de messages que représente le SGL. L'objectif est de filtrer dans l'interface graphique l'affichage des données reçues en fonction du groupe de collection à laquelle l'utilisateur est connecté.

Les chiffres représentent les Ids en base de données pour les tables PLATEFORME (base tumo2) et EMETTEUR (base tumo2interfaçages ou tumo2) sous la forme :

PLATEFORME_ID : EMETTEUR_ID

La base de données doit donc contenir une entrée pour ces deux tables. L'entrée EMETTEUR référence la table LOGICIEL Chaque dossier transmis par un SGL à **TumoroteK** est attribué à un EMETTEUR lors de son enregistrement en base de données (voir INTERFACAGES_INBOXES). Les dossiers disponibles dans l'interface graphique sont donc ensuite filtrés par EMETTEUR par la variable INTERFACAGES.

Il est possible de définir plusieurs associations PLATEFORME – EMETTEUR sous la forme:

INTERFACAGES = pf1:emet1 ; pf2:emet1,emet2,emet3

La variable INTERFACAGES_INBOXES définit le chemin absolu vers le fichier de configuration **inboxes.xml**. Ce fichier définit les régles générales de parsing du message, l'attribution des données exploitables par **TumoroteK** à une entrée EMETTEUR, et la déclaration des associations entre les champs d'informations du SGL et ceux de **TumoroteK**.

inbox.xml : le segment MSH-2 du message (Sending application) doit correspondre à l'attribut 'nom' d'un bloc **BoiteFtp** :

```
<BoiteFtp nom="DIAMIC">
```

Le bloc BoiteFtp contient deux segments <Emetteur> et <Service> dont le contenu (MSH-2 et MSH-21.1) permettront de trouver l'entrée EMETTEUR en base de données qui sera associée aux données enregistrées.

```
<Emetteur>
    <Bloc>MSH</Bloc>
    <Emplacement>2</Emplacement>
</Emetteur>
<Service>
    <Bloc>ORC</Bloc>
    <Emplacement>21.1</Emplacement>
</Service>
```

Il est ainsi possible de définir pour un même SGL, plusieurs entrées EMETTEUR qui vont correspondre à des services de destinations bien distincts, et donc de router les données contenues par les messages à l'enregistrement en fonction de leur segment MSH, puis à l'affichage dans l'interface graphique.

Le bloc BoiteFtp contient un bloc **MappingXml** qui définit le chemin absolu vers le fichier d'association des champs d'informations **diamic-mapping.xml** :

```
<MappingXml>
    <XML>C:/Program Files/Apache Software Foundation/Tomcat 7.0/conf/Catalina/localhost/sgl/diamic-mapping.xml</XML>
</MappingXml>
```

diamic-mapping.xml : les associations entre les champs d'informations sont sous le forme:

```
<Bloc nom="PID">
    <Mapping>
        <Tk>
            <ChampEntite nom="Nom" idChamp="3"></ChampEntite>
            <Entite nom="Patient" idChamp="1"></Entite>
        </Tk>
        <Source>
            <Key>5.1</Key>
        </Source>
    </Mapping>
```

L'attribut nom de la balise <Bloc> correspond à l'entête des segments HL7, et contient donc plusieurs <Mapping>, pour chaque informations présente dans la ligne.

La balise <Tk> définit le champ d'information dans TumoroteK à partir d'un couple d'Ids pour des entrées des tables CHAMP_ENTITE et ENTITE en base de données.

La balise <Source> définit l'adressage de l'information dans la ligne du message suivant la configuration des séparateurs :

```
<Configuration>
    <SeparateurChamps>|</SeparateurChamps>
    <SeparateurComposants>^</SeparateurComposants>
    <SeparateurSousComposants>~</SeparateurSousComposants>
    <BlocLibreKey></BlocLibreKey
```

Exemple: <Source>5.2</Source> pour le bloc PID suivant

PID|1||98761245||TEST^FRANCOIS^^^^^L||19210519|M

Renvoie 'FRANCOIS'

9

<Source>5</Source> renvoie 'TEST^FRANCOIS^^^^^L'

Pour les associations vers les champs d'informations normalisés dans **TumoroteK** (thésaurus). Il est nécessaire d'appliquer la balise <Modifier> qui permet de définir au travers des balises <MappingThes> les associations entre l'information reçue, et celle normalisée.

Exemple avec la nature du prélèvement :

```
<Source>
        <Key>4.1</Key>
        <Modifier nom="Thesaurus">
        <MappingThes>
                <SourceThes>UVP1</SourceThes>
                <TkThes>TISSU</TkThes>
        </MappingThes>
        <MappingThes>
                <SourceThes>Prelevement Sanguin</SourceThes>
                <TkThes>SANG</TkThes>
        </Source>
```

Les valeurs UVP1 et Prelevement Sanguin envoyées par le SGL seront respectivement associées aux natures de prélèvements normalisés TISSU et SANG.

12)

13)Résolution des problèmes

Le redémarrage de **TumoroteK**, et par extension du serveur Apache Tomcat, entraine un redémarrage automatique et complet de l'interfaçage.

14) a) Réception

Les messages dont le traitement à la réception n'a pu être réalisé et a produit une erreur sont immédiatement déplacés vers le dossier **DeadLetter** défini par le composant sgl.deadLetter (voir page 6).

Une trace de l'erreur est écrite dans le fichier **<TOMCAT>/logs/sgl_trace.log**.

Il est possible de déplacer manuellement un message en erreur dans la boite de réception afin de relancer la consommation du message, et suivre la trace de l'erreur en temps réel.

Les erreurs sont le plus fréquemment causées par:

- un problème de format dans le fichier: l'éditeur du logiciel émetteur doit être informé du mauvais formatage
- une information au format inattendu par l'interfaçage: les fichiers de configurations **TumoroteK** doivent être modifiés en conséquence
- une information incohérente ou invalide selon le système de validation **TumoroteK**: l'erreur permet alors de pointer et corriger les données dans la base **TumoroteK** et/ou dans la base de données du sgl émetteur (exemples : NIPs non uniques, doublons d'identités etc...).

14.1.1.1.1.1.1.1.b) Emission

Les messages dont la production en vue d'une émission a été avortée en levant une erreur sont immédiatement déplacés vers le dossier **DeadLetter** défini par le composant ack.deadLetter (voir page 7).

Une trace de l'erreur est écrite dans le fichier **<TOMCAT>/logs/ack_trace.log**.

Les erreurs sont le plus fréquemment causées par:

- une impossibilité d'écrire le fichier sur disque (droits d'accès de l'utilisateur ayant démarré le serveur Apache Tomcat, espace disponible…).