CID : un protocole de transaction pour les documents pédagogiques

Thibaut Arribe[1], Stéphane Crozat[2], and Sylvain Spinelli[3]

[1] Université de Technologie de Compiègne,
Laboratoire Heudiasyc UMR CNRS 7253,
thibaut.arribe@hds.utc.fr,
[2] Université de Technologie de Compiègne,
Unité ICS,
stephane.crozat@utc.fr,
[3] Société Kelis,
sylvain.spinelli@kelis.fr

Abstract. Cette contribution présente le protocole Content Interactive Delivery (CID). CID vise à encadrer les transactions de documents ou de métadonnées entre deux systèmes documentaires. Son originalité réside dans la possibilité de définir des transactions entre trois acteurs : le logiciel client à la source de la transaction, le logiciel serveur cible de la transaction et l'usager à l'initiative de la transaction. L'usager peut communiquer directement avec le serveur à travers une interface web affichée par le client. CID repose sur une exposition des services accessibles par le serveur dans un fichier de manifeste et une exécution par le client des étapes définies par le serveur.

Keywords: Protocole, déploiement, ressource pédagogique, document

1 Introduction

Les environnements pédagogiques numériques sont composés d'un large panel de solutions techniques complémentaires qui manipulent les mêmes documents à différents instants de leur cycle de vie. Usuellement, les échanges de documents d'un système à un autre sont effectués à la main par les usagers. Cette manipulation est laborieuse et source d'erreurs. Cette contribution présente le projet de protocole Content Interactive Delivery[3] (CID) qui cherche à accompagner les échanges de documents ou de leurs métadonnées entre différents systèmes. CID a été conçu dans le cadre du projet investissements d'avenir SUP E-educ[6] dont l'objectif est de moderniser les dispositifs numériques de formation dans l'enseignement supérieur.

La première partie de cette contribution expose la problématique adressée. Une seconde partie traite du seul protocole standard au spectre fonctionnel proche : PENS. Le protocole CID est présenté dans une troisième et dernière partie.

2 Problématique

Trois grandes familles de systèmes participent au cycle de vie des documents pédagogiques : les systèmes de production comme les LCMS (*Learning Content Management System*) ou les chaînes éditoriales dont l'enjeu principal est d'optimiser la production de documents pédagogiques ; les systèmes d'exploitation pédagogiques comme les LMS (*Learning Management System*) ou les MOOC (*Massive Open Online Course*) dont l'enjeu est de permettre une exploitation des documents pédagogiques par les apprenants ; les systèmes de gestion et d'archivage comme les logiciels de gestion électronique de documents (GED) ou les systèmes de gestion de bibliothèque dont l'enjeu est d'accompagner le cycle de vie des documents.

Pour assurer leur fonction, ces systèmes proposent des processus fonctionnels dédiés à leur métier (comme par exemple des processus de publication, de déploiement, d'archivage ou de mise à jour). Pour être opérationnels, les systèmes déployés dans un même environnement doivent communiquer entre eux des informations (comme par exemple des métadonnées) ou directement s'échanger des documents. Ces transactions sont assurées, soit manuellement par l'usager, soit par un module de connexion développé spécifiquement pour la liaison entre deux systèmes. Dans le premier cas, les transactions manuelles sont laborieuses et sources d'erreurs. Dans le second, les contraintes techniques relatives à chaque système imposent le développement et la maintenance d'un connecteur par paire de systèmes, voire d'une version de connecteur par version de paire de systèmes.

La solution idéale serait un protocole automatisant l'ensemble du processus. Cela reviendrait à disposer d'un bouton qui, une fois cliqué, assurerait l'ensemble de la transaction, quel que soit le système source, le type de communication et le système cible. Une telle automatisation est impossible à mettre en œuvre. La représentation des documents inhérente à chaque système, les métadonnées associées ainsi que les contraintes techniques relatives au métier ne permettent pas la mise en place d'un tel protocole universel.

La médiation entre deux systèmes ne peut se faire sans l'usager. C'est sa compréhension de chacun des systèmes qui lui permet d'ajuster les processus fonctionnels utilisés. Il est le seul à savoir ce que signifie l'arrivée d'un document dans son contexte. Par exemple, seul lui sait si le transfert d'un document depuis une chaîne éditoriale vers une LMS sert à alimenter un nouveau cours (qui vient de s'ouvrir en formation continue) ou à mettre à jour un ancien cours (en formation initiale, déjà démarré).

Il y a une forte tension entre un besoin d'encadrement et d'automatisation, un contexte fortement hétérogène, et une nécessité de médiation des transactions par les usagers. Notre contribution adresse cette tension. Reformulée autrement, elle cherche à fluidifier les échanges entre systèmes documentaires hétérogènes tout en proposant une médiation optimisée de l'usager.

3 État de l'art : Package Exchange Notification Services (PENS)

Cette section présente le standard PENS[4], unique standard s'intéressant au déploiement de ressources pédagogiques. Cette section décrit son principe de fonctionnement et revient ensuite sur ses limites.

3.1 Principe

PENS standardise un principe d'échanges de notifications qui permet à un logiciel de production de documents (le client) de déployer un paquet SCORM sur une plate-forme LMS (le serveur). Le client envoie une première notification au serveur contenant une URL vers une ressource produite (accessible via les protocoles HTTP, HTTPS, FTP ou FTPS), une description du contenu, les identifiants permettant de récupérer la ressource et une URL permettant au serveur d'envoyer des notifications d'avancement. Le serveur collecte le contenu et envoie une notification de collecte réussie si l'URL de notification de retour a été spécifiée. Le serveur peut alors déployer le contenu et à nouveau envoyer une nouvelle notification.

3.2 Limites

Le standard PENS spécifie les modalités techniques de transfert d'un paquet SCORM. Ce faisant, il évite le problème de la complexité des transactions documentaires en réduisant son spectre fonctionnel à un unique cas d'usage. C'est le principal reproche que nous formulons à son égard : PENS ne répond qu'à un faible sous-ensemble des problématiques de transactions documentaires.

En outre, le standard PENS pose l'hypothèse d'une possible automatisation du déploiement d'un contenu SCORM à partir de sa description et des identifiants de l'usager. Nous réfutons cette hypothèse ; quand bien même son application est limitée à un spectre fonctionnel si réduit, la médiation de l'usager nous semble souvent nécessaire.

Tout en considérant le spectre fonctionnel restreint du standard, les modalités techniques de PENS semblent peu judicieuses. Elles nécessitent un client et un serveur adressables sur le réseau, soit deux logiciels fonctionnant sur une machine disposant d'une IP publique. L'usage d'une plate-forme LMS implique son adressabilité sur le réseau. En revanche, il est courant que des logiciels de production soient des logiciels personnels pouvant être utilisés sur un sous-réseau invisible de la plate-forme LMS (par exemple, derrière un routeur depuis une connexion internet personnelle), ou ne soient pas conçu pour écouter les requêtes issues du réseau (comme par exemple un logiciel de bureau de type outil-auteur de contenus pédagogiques).

À l'instar de débats ayant déjà eu lieu sur les standards d'encapsulation des contenus pour en permettre l'interopérabilité[5], nous soutenons que la diversité des contextes ne permet pas l'adoption d'un standard définissant un

procédé de déploiement universel. Le standard a pour rôle d'accompagner la diversité des contextes et non d'imposer leur uniformisation. Un standard pour le déploiement de ressources pédagogiques doit proposer un cadre, certes universel, mais également spécialisable pour être adapté à chaque contexte.

4 Proposition : le protocole CID

4.1 Présentation

Le protocole CID a été élaboré dans le cadre du projet investissements d'avenir SUP E-educ. Son objet est d'encadrer les transactions documentaires entre trois acteurs : un logiciel client à la source de la transaction ; un logiciel serveur, cible de la transaction ; un usager, à l'origine de la transaction.

Fonctionnement Un serveur est un logiciel qui propose des processus fonctionnels. Le protocole CID permet la spécification de ces processus dans un manifeste. Ce fichier descriptif doit être librement téléchargeable sur un serveur HTTP.

Un client est un logiciel permettant d'instancier des transactions documentaires avec un serveur. Le client doit récupérer le manifeste via une simple requête HTTP non authentifiée. En interaction ou non avec l'usager, le client doit sélectionner la transaction à effectuer et exécuter les étapes qui y sont spécifiées. Ces étapes peuvent être :

- de simple requêtes entre le client et le serveur. Elles transportent alors un fichier ou des métadonnées ;
- une interaction entre l'usager et le serveur à travers une interface web envoyée par le serveur et affichée par le client.

Le manifeste Le manifeste est composé de trois parties : une liste de processus fonctionnels, les schémas d'authentification et les spécifications de la couche transport à utiliser.

Chaque processus est composé d'une spécification des métadonnées qu'il utilise (attendues pour l'exécution ou renvoyées en résultat d'une étape) puis de l'enchaînement des étapes attendues. Les étapes peuvent être de simples échanges de métadonnées, l'envoi de fichiers vers le serveur ou des interactions directes entre l'usager et le serveur.

La partie dédiée au transport permet de lier chacune des étapes avec des modalités de transport. CID propose exclusivement l'usage du protocole HTTP pour la couche transport. La description du transport spécifie les requêtes à utiliser (GET, PUT, POST), la façon de stocker les méta-données (en paramètre de l'URL (*query string*[1]), dans l'en-tête ou dans le corps de la requête), la nécessité d'inclure des propriétés de session ou encore l'usage de cookies HTTP[2].

L'exemple de la figure 1 définit un processus de déploiement de fichiers SCORM qui se déroule en deux étapes : l'envoi du fichier (*cid:upload*) et une

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cid:manifest xmlns:cid="http://www.kelis.fr/cid/v2/core">
  <cid:process>
    <cid:label xml:lang="en">SCORM deployment</cid:label>
    <cid:meta name="Content-type">
      <cid:value>application/xx-scorm;v1.2</cid:value>
      <cid:value>application/xx-scorm;v2004</cid:value>
    </cid:meta>
    <cid:upload url="http://lms.com/upload" needMetas="Content-type" required="true"/>
    <cid:interact url="http://lms.com/interact"  required="true"/>
  </cid:process>
  <cid:authentications><cid:basicHttp/></cid:authentications>
  <cid:transports>
    <cid:webTransport needCookies="true">
      <cid:webInteract method="GET"/>
      <cid:webUpload method="PUT" properties="queryString"/>
    </cid:webTransport>
  </cid:transports>
</cid:manifest>
```

Fig. 1. Exemple de manifeste - déploiement de fichiers SCORM

interaction entre l'usager et le serveur afin de procéder au déploiement du fichier sur la plate-forme (*cid:interact*). Lors de l'envoi, le client doit spécifier le type de fichier envoyé (*needMetas="Content-type"*). Le serveur n'accepte que deux types de fichiers : les archives SCORM version 1.2 et version 2004 (*application/xx-scorm;v1.2*, *application/xx-scorm;v2004*). Le fichier doit être envoyé au serveur dans une requête HTTP PUT (*cid:webUpload method="PUT"*), la métadonnée y est stockée dans l'URL sous la forme d'une *query string*.

Pour qu'un client déploie une ressource SCORM en exploitant ce manifeste, il doit :

1. récupérer et interpréter le manifeste par une requête HTTP GET (par exemple, vers l'URL `http://www.lms.com/manifest`) ;
2. envoyer une requête HTTP PUT contenant le fichier (à l'URL `http://www.lms.com/upload?Content-type=application/xx-scorm;v1.2`) ;
3. envoyer une requête HTTP GET (vers l'URL `http://www.lms.com/interact`) et afficher le résultat dans une frame web.

Étape *interact* L'étape *interact* permet de spécifier une interaction directe entre le serveur et l'usager. Le serveur envoie une page web que le client doit afficher dans une frame web. L'interaction se termine par un événement javascript personnalisé lancé au sein de la frame (*cid-end-interaction*). Le client peut alors récupérer des métadonnées issues de l'interaction dans le corps de l'événement puis fermer la page d'interaction. Cette étape est la principale originalité de CID. Elle permet au serveur d'ajuster la transaction en médiation directe avec l'usager sans développements *ad hoc* côté client.

4.2 Exemple : Envoi d'une vidéo vers un serveur de fichiers multimédia

Cet exemple est complémentaire au manifeste de la figure 1. Il montre la diversité des situations pouvant être outillées avec le protocole CID. Dans cet exemple,

un logiciel de production souhaite envoyer et référencer un fichier vidéo sur un serveur de fichiers multimédia (Digital Asset Management - DAM). Nous suggérons un processus en trois étapes :

1. Interact - afin de permettre à l'usager de préciser la façon dont le serveur doit traiter le contenu envoyé.
2. Upload - afin d'envoyer le fichier vidéo.
3. Interact - afin d'obtenir une interface de suivi du traitement du fichier et de récupérer l'URL publique.

Les vidéos sont des fichiers volumineux et leur transit sur le réseau est long. La première étape permet de vérifier l'authentification et de proposer une médiation avant le dépôt. Ainsi, à l'issue de l'envoi du fichier, l'usager n'a aucune tâche à réaliser et peut laisser le processus se terminer en tâche de fond. Dans cet exemple, l'unique objectif de la dernière étape est de proposer une interface de suivi du déploiement (comme par exemple, une barre d'avancement ou des notifications).

5 Conclusion

Dans cette contribution, nous avons présenté notre proposition de protocole dénommé CID. De premières expérimentations techniques ont été menées dans le cadre du projet Sup E-educ en instrumentant les transactions entre la chaîne éditoriale Opale (`http://scenari-platform.org/projects/opale/fr/pres`), le logiciel de GED Nuxéo (`http://www.nuxeo.com`) et la plate-forme LMS Moodle (`https://moodle.org`). Le travail de spécification reste néanmoins à poursuivre avec l'accompagnement de nouveaux contextes techniques et fonctionnels. La poursuite de nos travaux sera dédiée à la diffusion du protocole dans un panel de contextes plus large afin d'augmenter les retours techniques et fonctionnels. Il s'agira d'affiner les spécifications du protocole et de valider les choix techniques et fonctionnels.

References

1. RFC 3986. *Uniform Resource Identifier (URI): Generic Syntax.* Internet Engineering Task Force (IETF).
2. RFC 6265. *HTTP State Management Mechanism.* Internet Engineering Task Force (IETF).
3. CID. `http://www.cid-protocol.org`.
4. CMI010. *Package Exchange Notification Services (PENS).* AICC, 2006.
5. Stéphane Crozat, Nicolas Delestre, Jacques Qyeyrut, Fabien Baillon, Fabien Gautron, Christine Vanoirbeek, Priscilla Velut, Xavier Hennequin, et al. Standardisation des formats documentaires pour les chaînes éditoriales d'unit: un schéma pivot. In *Actes de la conférence TICE*, 2006.
6. PIA SUP E-educ. `http://www.universites-numeriques.fr/SUP-E-educ/`.