Dedalus

BIOSERVEUR

Protocole de communication sécurisé Medxfer

INDEX

Révision	Date	Action
1.0	30 septembre 2021	Version initiale

1 Références

- https://datatracker.ietf.org/doc/html/rfc7233 : « Hypertext Transfer Protocol (HTTP/1.1) »
- https://datatracker.ietf.org/doc/html/rfc2045 : Multipurpose Internet Mail Extensions (MIME)

2 Objet

Ce projet définit un protocole de transfert sécurisé de fichiers de manière à pallier les problèmes liés à HPRIM Net, il permet une communication directe entre un client et un serveur.
Fonctionnalités :

- Utilisation des protocoles standards,
- authentification forte par certificats X.509 (PKI SHA-256),
- signature du transfert de manière à garantir l'intégrité et l'origine des fichiers (le fichier est exploitable uniquement après transfert complet et vérifications),
- facilement déployable au travers des firewall,
- transfert bidirectionnel client vers ou depuis le serveur (upload / download).

3 Fonctionnalités

3.1 Description du service

Le service fourni est le tranfert de fichiers avec autorisation forte du client et du serveur ainsi que le contrôle de l'intégrité du tranfert en s'appuyant sur les standards :

Protocole	Caractéristiques
HTTPS	protocole standard largement utilisé (par tous les sites Web sécurisés) passe facilement à travers les firewall chiffrage des données pendant le transfert authentification forte du serveur par certificat X.509 - SHA256 authentification forte du client par certificat X.509 - SHA256 permet la gestion des listes de révocation de certificats X.509 transfert bidirectionnel de fichiers entre le client et le serveur (HTTP/1.1) compression possible des données lors de la transmission *Ne garantit pas l'intégrité des données échangées*
MIME	Structuration des données échangées Ce format est utilisé pour associer une signature SHA1 à chaque fichier échangé afin de valider son transfert et garantir l'intégrité des données.

Les fonctions réalisées permettent :

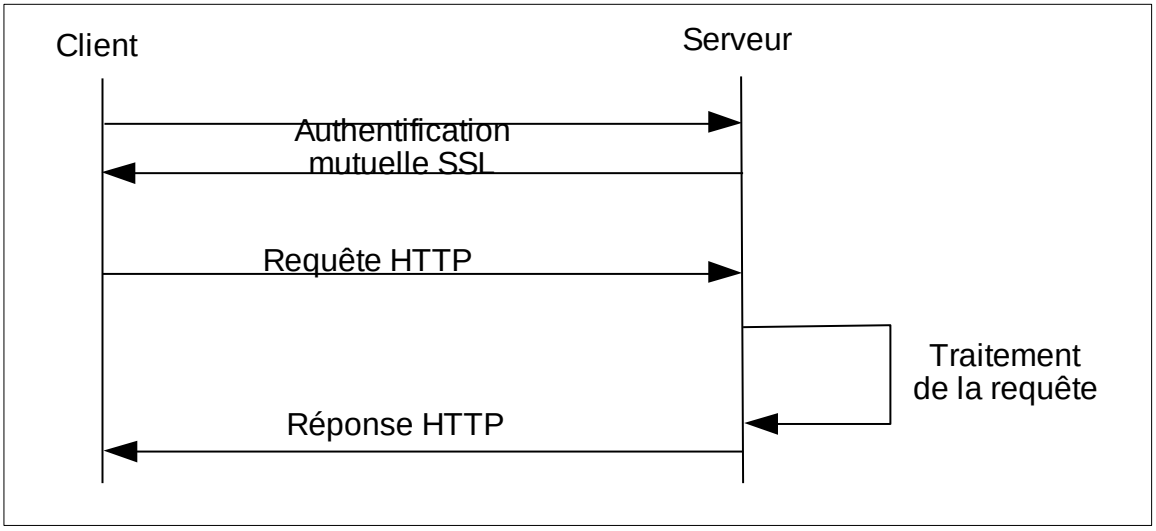
- L'envoi de fichiers sur le serveur avec contrôle d'intégrité (upload),
- récupération de la liste de groupes de fichiers disponibles sur le serveur (list),
- récupération d'un ou de l'ensemble des groupes de fichiers disponibles sur le serveur (download/downloadall).

Ces différentes fonctions sont réalisées avec un enchaînement de requêtes HTTPS décrites ci-après.

3.1.1 Description générale d'une requête HTTPS

Une requête HTTPS se déroule en deux phases :
- Une phase d'authentification basée sur le protocole SSL,
- une phase d'échange de données utilisant le protocole HTTP/1.1.

Le diagramme ci-dessous décrit le déroulement d'une requête.



3.1.1.1 Authentification SSL

Le serveur Web est configuré pour n'accepter la connexion qu'après la réception d'un certificat client valide. Le serveur permet la gestion d'une liste de révocation des certificats.

Cette phase a pour buts de :
- La validation par le client de l'identité du serveur (définie par son certificat X-509), le client dispose pour cela du certificat de l'autorité de certification (CA) ayant signé le certificat serveur,
- la validation par le serveur de l'identité et de la validité du certificat X-509 client,
- l'échange de clés pour le chiffrement des échanges HTTP/1.1.

3.1.1.2 Requête HTTP

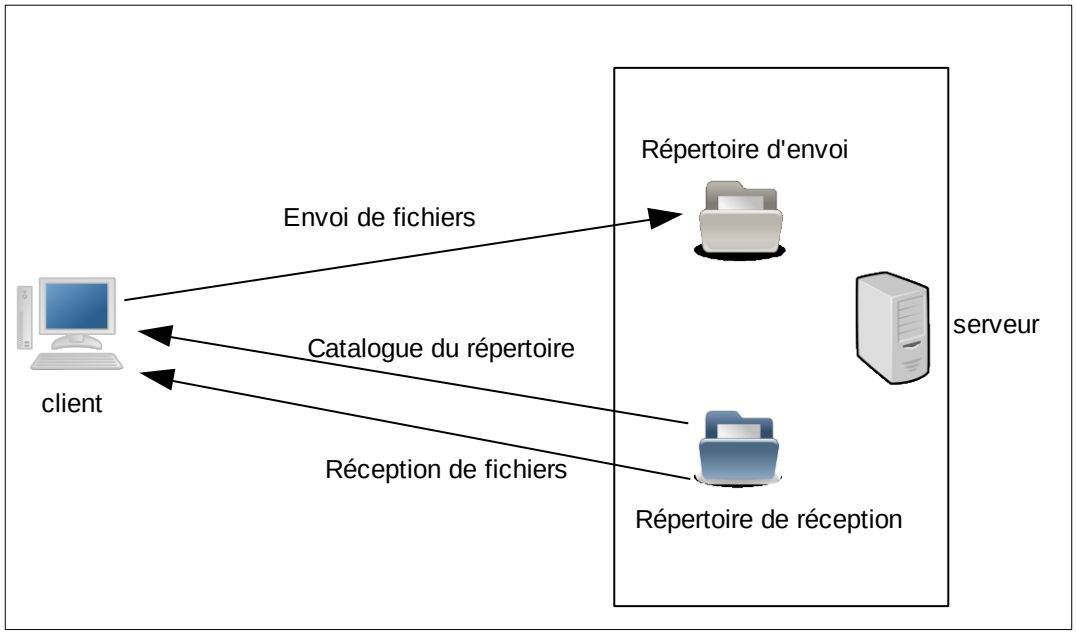
Lors de la phase d'authentification SSL, les informations d'authentification du client sont enregistrées par le serveur. Ces informations sont accessibles lors du traitement de la requête. Dans le cas du serveur Apache, ces informations sont accessibles sous forme de

variables d'environnement, par exemple la variable SSL_CLIENT_S_DN_CN définit l'identité « common name ») du client.

3.2 Protocole de communication

Ce chapitre décrit les séquences d'échanges entre le client et le serveur pour réaliser les fonctions suivantes :

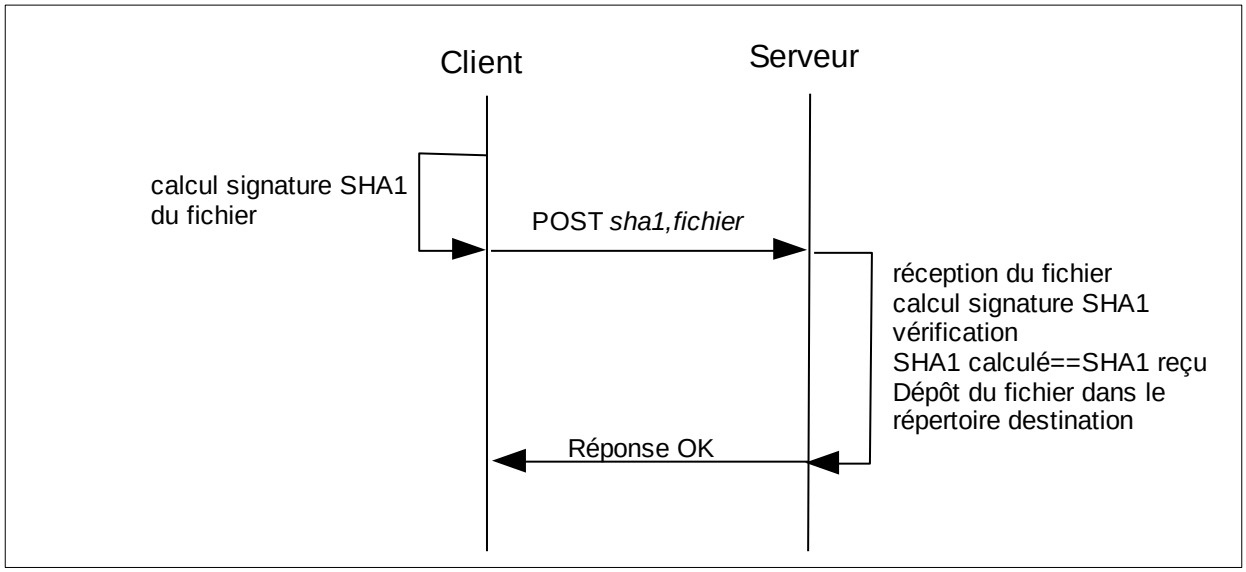
- envoi d'un fichier au serveur,
- réception de la liste des groupes de fichiers disponibles sur le serveur,
- réception d'un groupe de fichiers à partir du serveur.



Cette spécification modélise le transfert de données entre le client et le serveur sous la forme d'envoi et de récupération de fichiers sur des répertoires du serveur. Elle n'implique pas que ces répertoires ont une existence réelle. Ce protocole peut en effet être utilisé par une application sur le serveur qui traite à la volée les données reçues et/ou génère dynamiquement les données transmises à partir d'informations qui ne sont pas stockées dans des fichiers (base de données,...).

3.2.1 Envoi d'un fichier avec contrôle d'intégrité (upload)

Le schéma ci-dessous décrit la séquence correspondant à un envoi correct.



Le fichier et sa signature SHA1 sont envoyé dans une même requête POST en utilisant la structure de données MIME suivante :

```
POST /cgi/upload.pl HTTP/1.1
Host: server_hostname
Content-type: multipart/related; boundary=boundary;

--boundary
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="filename"
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: gzip

      ... contenu de filename ...
--boundary
Content-type: application/sha1-signature
Content-Transfer-Encoding: 7bit

signature_SHA1_de_filename
--boundary--
```

L'URL /cgi/upload.pl dans la directive POST est donnée à titre d'exemple.
Le contenu du fichier est compressé (gzip) puis codé en BASE64 afin de s'affranchir des problèmes de transmission de caractères spéciaux qui pourraient perturber le transfert dans le cas de données binaires.
La compression se fait avant la conversion en BASE64 et permet de réduire le temps de transfert. Le serveur doit donc décompresser les données reçues avant de les stocker dans le fichier « filename ».

De plus, la signature SHA1 doit être calculée par le client avant compression et conversion en BASE64 des données, elle est vérifiée par le serveur après décodage BASE64 et décompression.

Si aucune erreur n'est détectée, la réponse du serveur est :

```
HTTP/1.1 200 OK
```

A l'issue du transfert, le fait que le serveur réponde OK (code HTTP 200) indique que le transfert est complet et que la signature SHA1 a été vérifiée.
Le fichier reçu est déposé dans le répertoire de destination à l'issue du transfert complet et validation de sa signature SHA1.
Dans le cas où un fichier ayant le même nom existait sur le serveur avant le transfert, le contenu de l'ancien fichier est remplacé par les données reçues (si le nouveau fichier reçu est valide).
Il est possible de configurer le compte client sur le serveur pour permettre l'ajout automatique d'un préfixe d'horodatage du nom du fichier évitant ainsi l'écrasement d'une ancienne version d'un même fichier.

3.2.1.1 Gestion des erreurs

Dans tous les cas d'erreur, le serveur ne conserve pas le fichier reçu.
Conformément au standard HTTP, tout code d'erreur différent de 2xx doit être considéré comme une erreur, le serveur Web peut générer des erreurs autres que celles spécifiques à l'application de transfert de fichiers (ex: 404 « Not Found » si le client n'a pas spécifié la bonne URL dans la requête POST).

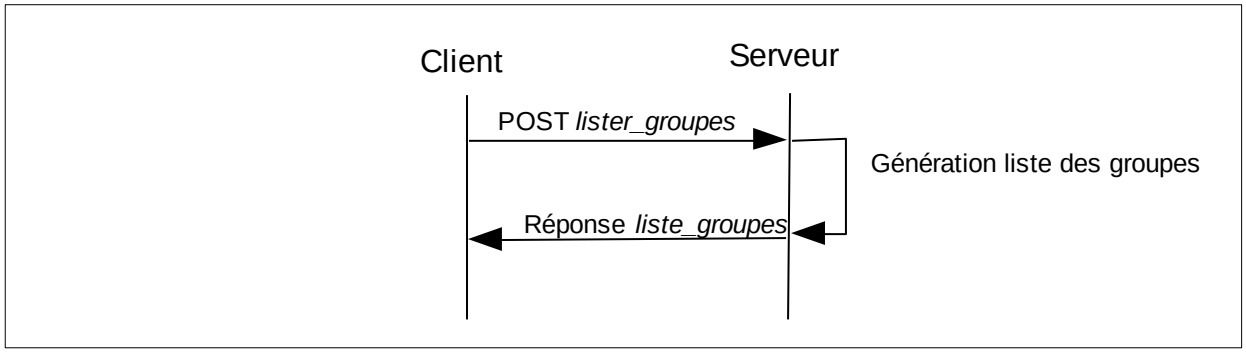
Différents code d'erreur sont possibles, le compte rendu d'erreur est fourni au format XML avec le code et le libellé détaillé de l'erreur, exemple :

```
HTTP/1.1 200 OK
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="iso-8859-1" ?>
<tranfer_error>
  <error_code>2</error_code>
  <error_message>Connection refused (Account not found)</error_message>
</tranfer_error>
```

3.2.2 Réception de la liste des groupes disponibles sur le serveur (list)

Le schéma ci-dessous décrit la récupération par le client de la liste des groupes de fichiers disponibles sur le serveur dans le dossier de réception du client.



Le serveur renvoie la liste des groupes de fichiers disponibles. Les identifiants de groupes de fichiers fournis par cette requête sont destinés à être utilisés pour la récupération d'un groupe (cf §« Réception d'un groupe de fichiers à partir du serveur (download) »).

La requête HTTP a la forme suivante :

```
GET /cgi/list_groups.pl HTTP/1.1
Host: server_hostname
```

La réponse du serveur est fournie au format XML :

```
HTTP/1.1 200 OK
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="iso-8859-1" ?>
<file_groups>
 <file_group>identifiant_groupe1</file_group>
 ...
 <file_group>identifiant_groupeN</file_group>
</file_groups>
```

3.2.2.1 Traitement des erreurs

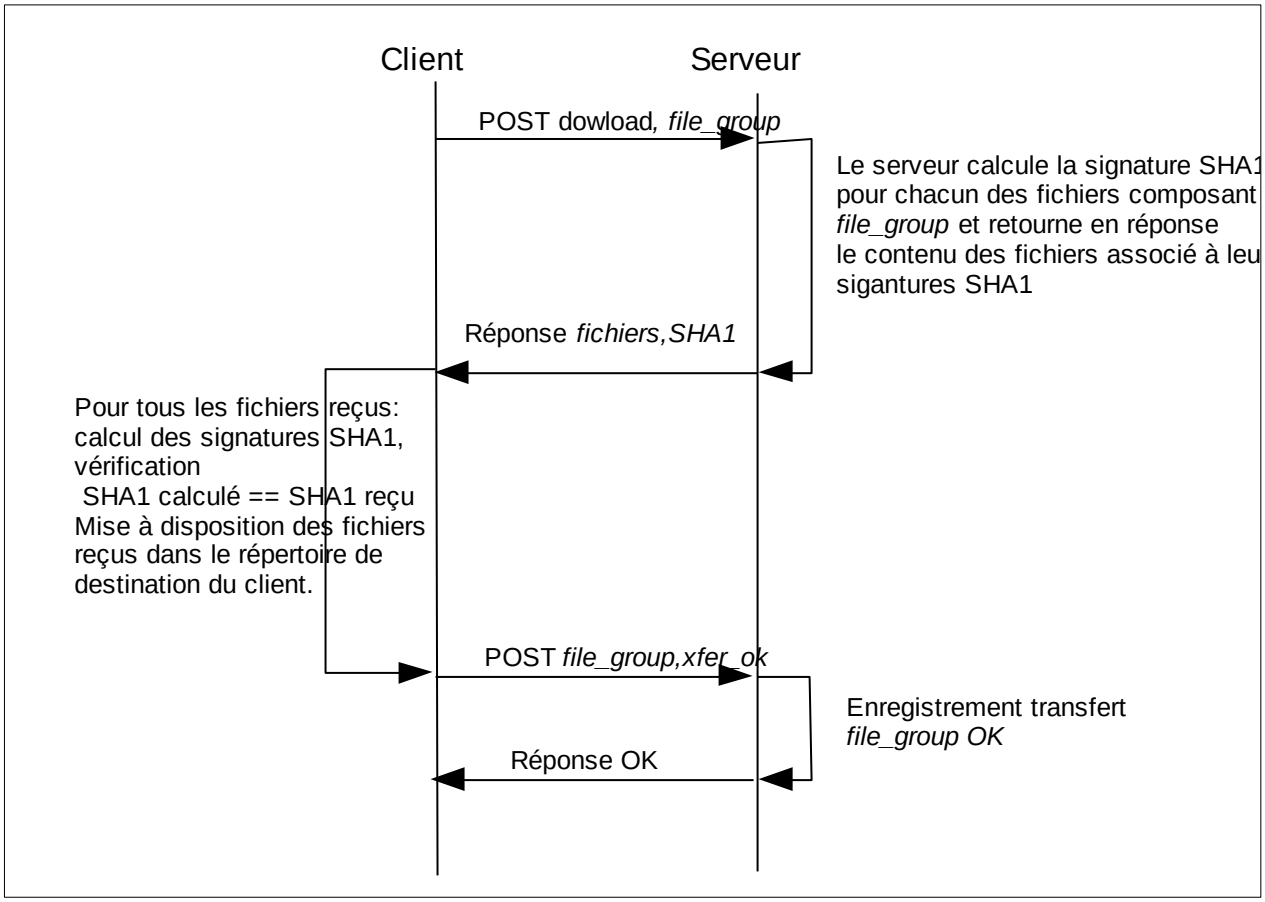
Le traitement des erreurs est identique à celui décrit dans le § « Envoi d'un fichier avec contrôle d'intégrité (upload) ».

3.2.3 Réception d'un groupe de fichiers à partir du serveur (download)

Cette fonction est utilisée pour récupérer plusieurs fichiers formant un ensemble cohérent dont le transfert n'est valide que si tous les fichiers ont été correctement transférés.
Le principe du transfert est le suivant:

- le client demande au serveur de lui envoyer un groupe de fichiers identifié par file_group,
- le serveur répond en envoyant tous les fichiers du groupe avec leurs signatures SHA1,
- le client vérifie la validité des données transmises en calculant les signatures SHA1 des fichiers reçus et en les comparant avec celles reçues,
- si tous les fichiers sont validés, ils sont mis à disposition dans le répertoire de destination du client,
- le client informe le serveur de la validité du transfert.

3.2.3.1 Requête d'envoi d'un groupe de fichiers




```
POST /download.cgi HTTP/1.1
Host: server_hostname
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="utf-8" ?>
<download>
 <file_group>identifiant_groupe</file_group>
</download>
```

Une structure de données XML est associée à la requête POST. Elle définit le groupe de fichiers à télécharger et doit correspondre à un des identifiants de groupe de fichiers retournés par la requête de demande de réception de la liste des groupes.

```
HTTP/1.1 200 OK
Content-type: multipart/related; boundary=boundary;

--boundary
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="filename1"
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: gzip

      ... contenu de filename1 ...
--boundary
Content-type: application/sha1-signature
Content-Transfer-Encoding: 7bit

signature_SHA1_de_filename1
--boundary
Content-Type: application/octet-stream
Content-Disposition: attachment; filename="filename2"
Content-Transfer-Encoding: base64
Content-Transfer-Encoding: gzip

      ... contenu de filename2 ...
--boundary
Content-type: application/sha1-signature
Content-Transfer-Encoding: 7bit

signature_SHA1_de_filename2
--boundary--
```

Le contenu du fichier est compressé puis codé en BASE64 afin de s'affranchir des problèmes de transmission de caractères spéciaux qui pourraient perturber le transfert dans le cas de données binaires.

3.2.3.2 Traitement des erreurs

Le traitement des erreurs est identique à celui décrit dans le § « Envoi d'un fichier avec contrôle d'intégrité (upload) », la réponse est donc de la forme :

```
HTTP/1.1 200 OK
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="iso-8859-1" ?>
<tranfer_error>
  <error_code>code_erreur</error_code>
  <error_message>message_erreur</error_message>
</tranfer_error>
```

3.2.3.3 Requête compte rendu de téléchargement

Cette requête n'est exécutée par le client que si le serveur n'a pas généré d'erreur à la requête précédente.
Le client informe le serveur du statut de la réceptiıon de l'envoi :

```
POST /download.cgi HTTP/1.1
Host: server_hostname
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="utf-8" ?>
<download_result>
 <error_code>0</error_code>
 <error_message>OK</error_message>
 <file_group>identifiant_groupe</file_group>
</download_result>
```

Pour des raisons d'homogénéité avec les codes d'erreur des autres échanges, la structure est la suivante :

- le code d'erreur « 0 » OK indique qu'il n'y a pas eu d'erreur,
- le libellé détaillé de l'erreur,
- l'identifiant du groupe concerné

Si aucune erreur n'est détectée, la réponse du serveur est :

```
HTTP/1.1 200 OK
```

Si une erreur se produit à cette étape dans la séquence des échanges, le serveur a envoyé le groupe de fichiers au client, celui-ci les a reçu et validé. Néanmoins, le serveur n'a pas été correctement informé de la validité du transfert. Le client considère donc que le transfert n'est pas correct et ne conserve pas les fichiers reçus.

3.2.4 Réception de tous les groupes de fichiers (downloadall)

Cette fonction a pour but de récupérer en une seule opération la totalité des groupes de fichiers disponibles en téléchargement, e
Elle s'appuie sur les fonctions « list » et « download » :

- « list » est utilisé pour récupérer la liste des groupes disponibles,
- « download » est utilisé pour récupérer tous les groupes en une seule requête.

Pour cela, cette requête est complétée pour pouvoir demander le téléchargement de tous les groupes de fichiers retournés par la commande « list » :

```
POST /download.cgi HTTP/1.1
Host: server_hostname
Content-Length: xxxx
Content-Type: text/xml; charset="iso-8859-1"

<?xml version="1.0" encoding="utf-8" ?>
<download>
 <file_group>identifiant_groupe1</file_group>
 <file_group>identifiant_groupe2</file_group>
 ...
 <file_group>identifiant_groupeN</file_group>
</download>
```

La gestion des échanges et des erreurs est identique à la requête de réception d'un groupe de fichiers.