SiTP v1.0

# Sommaire

[I. Sommaire 1](#_Toc270237111)

[II. Syntaxe 2](#_Toc270237112)

[A. Header 2](#_Toc270237113)

[B. Corps 2](#_Toc270237114)

[III. Identification 3](#_Toc270237115)

[A. Directe 3](#_Toc270237116)

[B. Indirecte 4](#_Toc270237117)

[IV. Base de données 5](#_Toc270237118)

[A. Droits 5](#_Toc270237119)

[B. Lire 6](#_Toc270237120)

[C. Tables 8](#_Toc270237121)

[D. Ajouter 8](#_Toc270237122)

[E. Modifier 9](#_Toc270237123)

[F. Supprimer 9](#_Toc270237124)

[V. Autres fonctions 10](#_Toc270237125)

[A. Statistiques 10](#_Toc270237126)

[B. Informations sur le serveur 13](#_Toc270237127)

[C. Gestion du serveur 13](#_Toc270237128)

[1. Configuration 14](#_Toc270237129)

[2. Arrêter / Redémarrer 15](#_Toc270237130)

[D. Plugins 16](#_Toc270237131)

[1. Lister 16](#_Toc270237132)

[2. Configuration 16](#_Toc270237133)

[3. Installer / Désinstaller / Charger / Décharger 17](#_Toc270237134)

[E. Télécharger la base de données 18](#_Toc270237135)

[F. Permissions 19](#_Toc270237136)

[G. Push 20](#_Toc270237137)

[H. Téléchargement 22](#_Toc270237138)

[I. Envoie 23](#_Toc270237139)

[J. Streaming 24](#_Toc270237140)

[K. Aperçu 26](#_Toc270237141)

[VI. Erreurs 27](#_Toc270237142)

# Syntaxe

## Header

Le protocole SiTP est basé sur la syntaxe de HTTP. Son header respecte les mêmes règles :

* La première ligne des requêtes est formée comme suit : "**Méthode URL Version**"
  + La **méthode** est l’action effectuée par la requête,
  + L’**URL** est l’adresse à laquelle se réfère la méthode. S’il n’y a pas d’url, mettez juste "/". Elle doit toujours être url-encodée (voir QUrl),
  + La **version** est celle du protocole (SiTP/1.0),
* La première ligne de la réponse est un peut différente : "**Version Code Message**"
  + La **version** est celle du protocole (SiTP/1.0),
  + Le **code** est une valeur qui peut représenter une erreur (404), ou que tout va bien (200)
  + Le **message** est un mot associé au code (404 = Not Found)
* Les lignes suivantes sont les propriétés du header. Elles se composent d’une clé, suivie de deux points, d’un espace, puis de la valeur. La casse n’a pas d’importance pour la clé. Exemple : "**Client: IPhone**",
* Le format des dates de SiTP est "**yyyy-mm-dd hh:mm:ss**", et ont pour référence l’heure locale du serveur,
* Chaque lignes est séparé par les caractères "**\r\n**" (saut de ligne de type Windows),
* A la fin du header, vous devez sauter deux lignes, c'est-à-dire écrite "**\r\n\r\n**".
* **Erreurs possibles :** [**400**](#E400)**,** [**405**](#E405)**,** [**501**](#E501)**,** [**505**](#E505)

## Corps

Le corps est utilisé par le client pour envoyer des données au serveur et vis versa. Il se situe sous le header, après les caractères "\r\n\r\n". La taille du corps doit être spécifiée dans le header par la propriété "**Content-length**", en octets. Le corps est facultatif (Content-length à zéro, ou absent).

Le corps peut être au format JSON ou XML, ou n’importe quoi dans le cas d’échange de données (fichier, streaming, ...). Le client peut choisir le format dans lequel le serveur lui répondra via la propriété "**Content-type**" du header : "Content-type: **application/xml**" demande une réponse en XML, et "Content-type: **application/json**" demande une réponse en JSON. Par défaut, le format **XML** est utilisé. Pour connaitre les formats disponibles, consultez la partie V.B.

|  |  |  |
| --- | --- | --- |
| **Exemple de syntaxe** | | |
| **Requête** | "GET toto.xml SiTP/1.0\r\nContent-type: application/xml\r\n\r\n" | **GET** **toto.xml SiTP/1.0**\r\n  **Content-type**: application/xml\r\n  \r\n\r\n |
| **Réponse** | "SiTP/1.0 200 OK\r\nContent-type: application/xml\r\nContent-length: 27\r\n\r\nContenu du fichier toto.xml" | **SiTP/1.0 200 OK**\r\n  **Content-type**: application/xml\r\n  **Content-length**: 27\r\n  \r\n\r\n  Contenu du fichier toto.xml |

# Identification

Toutes les opérations faites sur le serveur par un client requièrent qu’il s’identifie avec un compte valide.

Il y a deux types d’identifications : l’identification directe, et l’identification indirecte.

## Directe

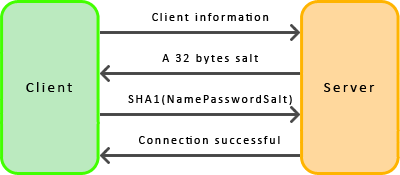
L’identification directe est le fait de se connecter directement au serveur, en sachant son adresse IP et le port d’écoute de SiTP. Elle se fait en deux étapes : La première consiste à informer le serveur qu’on souhaite s’identifier, et de lui envoyer des informations sur le client.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Le client se connecte au serveur, et envoi une requête d’identification. Le nom de la méthode est **CONNECT**. Dans cette requête, il donne des informations sur lui tels que l’heure locale (au format hh:mm), son type (C++, Android, …), sa version, etc. Ces informations sont facultatives mais doivent être renseignés autant que possible. | **CONNECT** **/ SiTP/1.0**  **Client**: IPhone  **OS**: IPhone OS 3.0  **Version**: 1.0  **Author**: StreamIt  **Time**: 17:42  **Language**: fr |
| **Réponse** | Dans sa réponse, le serveur insère un sel de 32 octets en hexadécimal généré aléatoirement. Il servira au client pour la prochaine étape. | **SiTP/1.0 200 OK**  **Salt**: acb61b8910fb97f1d043 35e738f7823c |

Dans la seconde étape, le client s’identifie au serveur. Si l’identification échoue un certain nombre de fois, le serveur refusera toute connexion de la part de cette IP pour un temps déterminé, afin d’éviter le brute force (voir erreur [**402**](#E402)). Le sel n’est valable que pour une requête. Un nouveau sel doit être demandé après chaque requête échouée.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Le client envoi un SHA1 dans la propriété "**Identifiant**" constitué de la concaténation du nom du compte, de son mot de passe (**sous la forme d’un SHA1**), et du sel précédemment envoyé par le serveur : **SHA1(NamePasswordSalt)**.Le sel n’est valable que pour une requête. De cette manière, même si une personne arrive à intercepter la requête, il ne pourra pas se connecter au serveur puisqu’il n’a ni le nom du compte, ni son mot de passe, et que le sel sera périmé. | **CONNECT** **/ SiTP/1.0**  **Identifiant**: 0b9c2625dc21ef0 5f6ad4ddf47c5f203837aa32c |
| **Réponse** | Le serveur retourne le code 200 (OK) si la connexion c’est bien passée, et le code 401 (Unauthorized) dans le cas contraire. | **SiTP/1.0 200 OK** |

Une fois un client connecté, il peut être déconnecté s’il reste inactif pendant un certain temps (voir V.B).



Ce schéma récapitule les échanges client/serveur lors de la connexion directe.

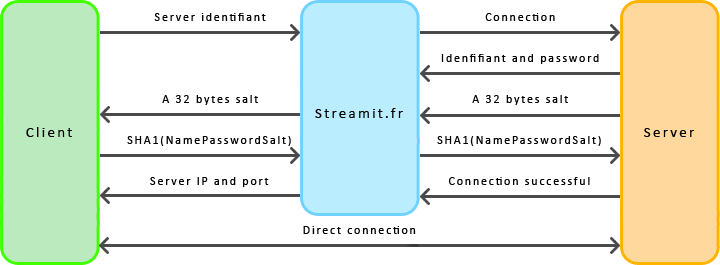
## Indirecte

Cette identification est dite indirecte car elle passe par le site StreamIt.fr pour identifier le serveur demandé par le client. Ceci permet à un client de se connecter à un serveur sans connaitre son IP ni son port.

Cela peut être très pratique dans le cas ou l’ordinateur sur lequel tourne le serveur change d’IP. De plus, les personnes peu familières avec la notion d’IP et de port n’auront pas à s’en soucier. Cependant, la connexion indirecte demande quelques prés-requis, et est plus lourde que la connexion directe.

* 1. La première étape est de créer un identifiant streamit. Cela se fait généralement lors de l’installation du serveur, ou ultérieurement. Cet identifiant est unique, et est enregistré sur le site streamit.fr. Un captcha est demandé par streamit.fr lors de cette étape afin d’éviter qu’un robot n’enregistre tous les noms possibles. Les identifiants sont supprimés après 30 jours d’inactivité. Une IP ne peut créer qu’un identifiant par minute. Lorsque l’identifiant est créé, streamit.fr envoi un mot de passe au serveur, qu’il devra utiliser pour s’identifier lors de la prochaine connexion à streamit.fr.
  2. A chaque démarrage du serveur, ce dernier se connecte à streamit.fr, et s’identifie en utilisant son identifiant et le mot de passe donné à la dernière connexion. Ensuite il donne le port sur lequel SiTP écoute. Streamit.fr lui donne un nouveau mot de passe, à utiliser pour la prochaine connexion. De cette manière, streamit.fr sait en permanence quel est l’IP et le port du serveur.
  3. La connexion indirecte à proprement parlée est constitués de plusieurs étapes :
     1. Le client se connecte à streamit.fr, et lui demande l’IP et le port du serveur dont il a l’identifiant,
     2. Streamit.fr se connecte au serveur, dont il connait l’IP et le port. Si le serveur n’est pas connecté, une erreur est retournée au client. S’il est connecté, streamit.fr demande le mot de passe et l’identifiant du serveur pour s’assurer que c’est le bon, et fait une demande de connexion (comme pour la connexion directe),
     3. Le serveur retourne un sel à streamit.fr,
     4. Streamit.fr retourne le sel au client,
     5. Le client retourne le SHA1 constitué du nom de l’utilisateur, du mot de passe, et du sel à streamit.fr,
     6. Streamit.fr envoi ce SHA1 au serveur,
     7. Si la connexion réussie, streamit.fr envoi l’IP et le port du serveur au client, et ferme toutes ses connexions (client et serveur),
     8. Le client entame une connexion directe normale avec le serveur.

**Cette architecture est toujours en cour de spécification, et sera implémentée plus tard.**



Représente la connexion indirecte. On remarque qu’elle se termine par une connexion directe.

# Base de données

## Droits

SiTP autorise les clients à effectuer quatre types d’actions sur les tables de la base de données : lire, modifier, ajouter, et supprimer. La plus part de ces actions nécessitent des droits spécifiques en fonction de la table ciblée.

Les champs **id**, **created**, et **modified** des tables ne sont jamais modifiables.

Le tableau ci-dessous associe les droits nécessaires pour accéder aux tables, en fonction des actions.

La légende qui suit liste les droits par priorité. Donc si une case du tableau est bleue, cela signifie que le propriétaire de l’élément et les administrateurs peuvent effectuer l’action.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tous | Droits | Propriétaire | Administrateur | Impossible |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Lire** | **Modifier** | **Ajouter** | **Supprimer** |
| **accounts** | Impossible de lire le mot de passe, même pour les administrateurs. | Les champs "active" et "administrator" ne sont modifiables que par les administrateurs. |  | Les administrateurs ne peuvent pas supprimer leur propre compte. |
| **accounts**  **groups** |  |  |  |  |
| **accounts\_**  **informations** |  | L'id\_account n'est modifiable que par les administrateurs. |  |  |
| **collections** | Droit de lecture. | Droit de modification. Pour modifier le champ id\_collection, droit d'ajout sur l'ancien et le nouveau id\_collection. L'id\_account n'est modifiable que par les administrateurs. | Droit d'ajout sur l'id\_collection. | Droit de suppression sur la collection, et sur toutes celles qu'elle contient. |
| **deleted** |  |  |  |  |
| **directories** | Droit de lecture. | Droit de modification. Pour modifier le champ id\_directory, droit d'ajout sur l'ancien et le nouveau id\_directory. L'id\_account n'est modifiable que par les administrateurs. | Droit d'ajout sur l'id\_directory. | Droit de suppression sur le dossier, et sur tous les objets qu'il contient. |
| **events** |  |  |  |  |
| **events\_**  **informations** |  |  |  |  |
| **files** | Droit de lecture. | Droit de modification. Pour modifier l'id\_directory, droit d'ajout sur l'ancien et le nouveau id\_directory. L'id\_account et le path ne sont modifiables que par les administrateurs, et seulement depuis l’ordinateur du serveur pour le path. | Seulement depuis l’ordinateur du serveur. | Droit de suppression. |
| **files\_**  **collections** | Droit de lecture sur la collection. |  | Droit d'ajout sur la collection, droit de lecture sur le fichier. | Droit d'ajout sur la collection. |
| **files\_**  **informations** | Droit de lecture sur l'id\_file. | Droit de modification sur l'id\_file. L'id\_file n'est modifiable que par les administrateurs. | Droit de modification sur l'id\_file. | Droit de modification sur l'id\_file. |
| **groups** |  |  |  |  |
| **limits** |  |  |  |  |
| **permissions** | Droit de lecture sur l'id\_object. |  |  |  |
| **tags** | Droit de lecture sur l'id\_object. | Droit de modification sur l'id\_object. L'id\_Object n'est modifiable que par les administrateurs. | Droit de modification sur l'id\_object. | Droit de modification sur l'id\_object. |

## Lire

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | SELECT |
| **Description** | Lit une ou plusieurs entrées de la base de données. Elle est comparable au SELECT d’une requête SQL. L’objectif n’est pas d’offrir une abstraction complète à la clause SELECT de SQL, mais de permettre aux clients d’accéder à la base de données simplement. Pour utiliser des fonctions SQL avancés, les clients doivent télécharger la base de données, et construire leur propre base en local (voir ). |
| **URL** | L’url indique le nom de la table ciblée, suivi d’un "**?**" puis des champs à lire séparés par des "**&**". Exemple : tableName**?**field1**&**field2**&**field3. Les champs sont appelés les arguments de l’url. Si aucun argument n’est précisé dans l’url, tous les champs de la table seront retournés. |
| **Propriétés** | Les propriétés du header correspondent à la clause WHERE d’une requête SQL.  Chaque champ différent ajoute une condition "**AND**" à la requête. Les champs identiques s’ajoutent avec des "**OR**". Par exemple mettre les propriétés "id, id, name, active" sera converti en "(id **OR** id) **AND** name **AND** active".  Il est également possible d’utiliser l’équivalent du LIKE de SQL, en mettant la valeur d’un champ entre guillemets, et en ajoutant les symboles "**%**" ou "**\_**". De la même manière que dans un LIKE. Exemple : « name : "Stre**\_**m**\***" » matchera pour Stre**a**m**It**, Stre**e**m**e**, Stre**a**m**Kiki**, … |
| **Droits** | Voir la partie |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Content-type** | La réponse est donnée au format **JSON** ou **XML**, en fonction de cette propriété. | Requête |
| **Limit** | Permet de limiter le nombre de résultats d’une requête. La valeur est constituée de deux nombres séparés par une virgule. Le premier indique à partir de quelle entrée afficher les résultats, et le second combien en afficher. C’est l’équivalent du LIMIT de MySQL. | Requête |
| **OrderBy** | Equivalent d’un ORDER BY de SQL. Contient le nom du champ à partir duquel trier les entrées, suivis du mot clé optionnel ASC (par défaut) ou DESC, qui permet d’ordonner les entrées de manière ascendante ou descendante. Vous pouvez enchainer plusieurs champs en les séparant pas des virgules. Exemple : "**id DESC, name**". | Requête |
| **Count** | Si cette propriété est définie à **true**, la réponse ne contiendra pas de corps, et le nombre d’entrées correspondant à la requête sera stocké dans la propriété **rows** (rows est retournée même si count n’est pas définie). | Requête |
| **Rows** | Nombre d’entrées correspondant à la requête/ | Réponse |
| **Content-Length** | Taille du corps de la réponse. | Réponse |

Voici un exemple simple qui consiste à demander des informations sur un fichier :

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Dans l’url de la requête, on met le nom de la table (**files**), suivi d’un "**?**", puis les champs que l’on souhaite récupérer (**name**, **type**, et **path**), séparés par le symbole "**&**“.  Dans les propriétés du header on donne les informations qui décrivent l’élément ciblé, ici l’**id** d’un fichier. L’équivalent de cette requête en SQL serais "**SELECT name, type, path FROM files WHERE id=** **f2aa9400-9239-11de-8a39-0800200c9a66**". | SELECT **files?name&type&path** SiTP/1.0  **Id**: f2aa9400-9239-11de-8a39-0800200c9a66  Content-type: **application/json** |
| **Réponse** | La réponse du serveur est un fichier au format JSON car le client a mis **application/json** dans le Content-type du header de la requête. Sa taille est indiquée par la propriété "**Content-Length**". L’élément **row** stock une entrée de la réponse. | SiTP/1.0 200 OK  **Content-length**: 120  Content-type: **application/json**  **Rows**: 1  {  "**files**": {  "**row**": {  "name": "toto",  "type": "avi",  "path": "C:\toto.avi"  }  } |

Dans cet exemple, un seul élément est récupéré, mais il est possible d’en obtenir plusieurs en même temps. Par exemple si on remplace "Id : f2aa9400-9239-11de-8a39-0800200c9a66" par "type : avi" dans le header de la requête, la réponse contiendra tous les fichiers ayant le type "avi".

Pour lire toutes les entrées d’une table, ne mettez pas de propriétés (ce qui équivaut à un "**SELECT \***"). Il est possible de mettre plusieurs fois la même propriété, ce qui équivaut à mettre un OR dans la clause WHERE de la requête SQL.

L’équivalent en XML de la réponse précédente est comme suit :

<files>

<row>

<name>toto</name>

<type>avi</ type >

<path>C:\toto.avi</path>

</row>

</files>

## Tables

Les tables **limits**, **events**, **permissions**, et **tags** stockent des id\_accessors ou des id\_objects. L’id d’un accessor peut représenter un compte ou un group. De façon similaire, l’id d’un object peut désigner un fichier, un dossier, ou une collection. Afin de limiter le nombre des requêtes, ces tables indiquent dans leurs réponses le type réel de leur object ou de leur accessor :

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande la liste des objects dont le tag est nature. | SELECT **tags**?**id\_object** SiTP/1.0  **Name**: nature |
| **Réponse** | Le serveur retourne les deux entrées qui ont nature dans leur tag. On peut remarquer qu’il y a un champ **object** dans chacune des entrées. Ce champ a pour valeur le nom de la table d’où provient l’id\_object, c’est à dire **files**, **directories**, ou **collections**. Ici, le premier objet est un fichier, et le second une collection. | SiTP/1.0 200 OK  Content-length: 266  Content-type: application/json  [ { "tags": [  { "row": {  "id\_object": " f2aa9...c9a66",  "**object**": "**files**" } },  { "row": {  "id\_object": "f2aa9...0c9a67",  "**object**": "**collections**"  } } ] } ] |

Le principe est le même lorsqu’un id\_accessor est présent dans une requête. Un champ accessor est ajouté, et peut valoir **accounts**, ou **groups**.

## Ajouter

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | INSERT |
| **Description** | Ajouter une entrée à la base de données. Comparable à l’INSERT d’une requête SQL. L’id de la nouvelle entrée n’a pas à être fournie. Il sera automatiquement généré par le serveur. Si un champ n’est pas spécifié, il sera ajouté avec sa valeur pas défaut. |
| **URL** | Nom de la table ciblée. |
| **Propriétés** | Les propriétés représentent les valeurs des champs de l’entrée à ajouter dans la table. Certaines tables disposent de champs qui doivent obligatoirement être remplies. |
| **Droits** | Voir la partie IV.A |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Ajoute un compte nommé **toto**. Le champ password n’est pas renseigné, et sera enregistré vide dans la base de données. L’équivalent SQL de cette requête est "**INSERT INTO accounts (name, administrator) VALUES (Toto, 0)**". | INSERT **accounts** SiTP/1.0  **Name**: Toto  **Administrator**: 0 |
| **Réponse** | La propriété **rows** indique si l’entrée a été ajoutée. Elle vaut 0 si ce n’est pas le cas. | SiTP/1.0 200 OK  **Rows**: 1 |

## Modifier

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | UPDATE |
| **Description** | Met à jour une entrée de la base de données. Comparable à l’UPDATE d’une requête SQL. |
| **URL** | Représente la table ciblée, et ses paramètres décrivent les champs à modifier. |
| **Propriétés** | Les propriétés correspondent à la clause WHERE d’une requête SQL (voir SELECT, IV.B). |
| **Droits** | Voir la partie IV.A |

La propriété **rows** de la réponse indique le nombre d’entrées modifiés.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Modifie le nom et l’extension d’un fichier. | UPDATE **files?name=tutu&type=avi** SiTP/1.0  **Id**: f2aa9400-9239-11de-8a39-0800200c9a66 |
| **Réponse** | La réponse indique si l’opération c’est bien déroulée, et retourne le nombre d’entrées affectés par le requête dans la propriété **rows**. | SiTP/1.0 200 OK  **Rows**: 1 |

## Supprimer

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | DELETE |
| **Description** | Supprime une entrée de la base de données. Comparable au DELETE d’une requête SQL. |
| **URL** | Nom de la table ciblée. |
| **Propriétés** | Les propriétés correspondent à la clause WHERE d’une requête SQL (voir SELECT, IV.B). |
| **Droits** | Voir la partie IV.A |

Si vous supprimez une entrée de la table files, le fichier concerné sera toujours présent sur le disque dur du serveur. Pour supprimer réellement un fichier, mettez la propriété **real** à **true**. Le principe est le même lorsque vous supprimez un dossier. Pour supprimer réellement les fichiers qu’il contient, utilisez real.

La propriété **rows** de la réponse indique le nombre d’entrées supprimées.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Supprime un fichier sur le serveur, virtuellement et réellement. L’équivalent SQL de cette requête est "**DELETE FROM files WHERE id=f2…66**". | DELETE **files** SiTP/1.0  **Id**: f2aa9400-9239-11de-8a39-0800200c9a66  **Real**: true |
| **Réponse** | La réponse indique si l’opération c’est bien déroulée, et retourne le nombre d’entrées affectés par le requête dans la propriété **rows**. | SiTP/1.0 200 OK  **Rows**: 1 |

# Autres fonctions

## Statistiques

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | STATISTIC |
| **Description** | Cette méthode permet aux clients d’accéder à des statistiques complètes sur le serveur. Ce dernier utilise principalement la table events pour les générer. On peut donc voir cela comme une abstraction de la table events. Notez que cette dernière n’est accessible que par les administrateurs. |
| **Fonctionnalités** | * Statistiques relatives aux comptes * Statistiques relatives au serveur * Statistiques relatives à un fichier * Statistiques complètes |
| **URLs** | Les fonctions statistiques. Les propriétés du header représentent les paramètres de ces fonctions. |
| **Droits** | Dépend des options et des fonctions statistiques. |

Toutes les fonctions statistiques peuvent prendre les propriétés **from** et **to** dans leur header. Elles permettent de spécifier de quand à quand calculer les statistiques, en jour. Par exemple si from est à 30, et to à 7, les statistiques seront calculés à partir de 30 jours jusqu’à 7 jour avant la date actuelle. From est toujours supérieur à to. Si from n’est pas défini ou est à 0, les statistiques seront calculées depuis la date d’installation du serveur. Si to n’est pas défini ou à 0, elles seront calculées jusqu’à la date courante.

Statistiques relatives aux comptes

Ces statistiques sont relatives à l’utilisateur courant. Un utilisateur normal ne peut pas voir les statistiques d’un autre utilisateur. Les administrateurs peuvent voir les statistiques des autres utilisateurs ou des groupes et ajoutant la propriété **view**, qui peut prendre indifféremment l’id d’un groupe ou d’un compte en valeur.

Un utilisateur peut voir les statistiques d’un des groupes dont il est membre en ajoutant son **id** dans **view**.

Les utilisateurs peuvent également visualiser ces statistiques relativement à tout le serveur en mettant la propriété **view** à **server**.

Les fonctions **downloadedFilesList**, **uploadedFilesList**, **streamedFilesList**, et **connectionsList** ne sont pas accessibles dans les view groupe, ou pour tout le serveur, sauf pour les administrateurs.

|  |  |
| --- | --- |
| **Fonction** | **Description** |
| **downloadedFiles** | Nombre de fichiers téléchargés. |
| **downloadedFilesList** | Liste la date, l’heure, et l’id des fichiers téléchargés. |
| **downloadedSize** | Nombre de Mo téléchargés. |
| **uploadedFiles** | Nombre de fichiers envoyés. |
| **uploadedFilesList** | Liste la date, l’heure, et l’id des fichiers envoyés. |
| **uploadedSize** | Nombre de Mo envoyés. |
| **streamedFiles** | Nombre de fichiers streamés. |
| **streamedFilesList** | Liste la date, l’heure, et l’id des fichiers streamés. |
| **streamedSize** | Nombre de Mo streamés. |
| **readFiles** | Nombre de fichier lus. Correspond à l’addition d’**downloadedFiles** et **streamedFiles**. |
| **connectedTime** | Temps connecté au serveur, en minute. |
| **connectedTimeSession** | Temps connecté pour cette session, en minute. |
| **connections** | Nombre de connexions au serveur. |
| **connectionsList** | Liste la date, l’heure, le client, le temps, et l’IP des connexions de l’utilisateur. |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Cette requête demande le nombre de fichiers téléchargés ces 30 derniers jours pour tout le serveur. Le nom de la fonction (**downloadedFiles**) est dans l’url. | **STATISTIC** **downloadedFiles** SiTP/1.0  **From** : 30  **View**: server |
| **Réponse** | Dans la réponse, le résultat est stocké dans une propriété qui possède le même nom que la fonction, ici **downloadedFiles**. Ces 30 derniers jours, 475 fichiers ont été téléchargés sur le serveur. | SiTP/1.0 200 OK  **downloadedFiles**: 475 |

L’exemple précédent s’applique aux réponses de type numérique. Pour les listes, la réponse est de cette forme :

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande la liste des fichiers téléchargés hier pour le compte courant. Le Content-type indique que la liste sera retournée au format XML. | STATISTIC **downloadedFilesList** SiTP/1.0  From: 2  To: 1  Content-type: application/xml |
| **Réponse** | La réponse est une simple liste d’éléments. Les id sont ceux des fichiers téléchargés hier. | SiTP/1.0 200 OK  Content-length: 307  Content-type: application/xml  <connectionsList>  <row>  <date>2010-03-14 16:42:12</date>  <id> f2aa9400-9239-11de-8a39-  0800200c9a66</id>  </row>  <row>  <date>2010-03-15 15:01:27</date>  <id> f2aa9400-9239-11de-8a39-  0800200c9a65</id>  </row>  </connectionsList> |

Statistiques relatives au serveur

|  |  |
| --- | --- |
| **Fonction** | **Description** |
| **serverConnectedRecord** | Nombre record de connectés sur le serveur. Retourne également la date du record dans la propriété **date** du header. |
| **serverConnectedTime** | Nombre de minutes de connexion du serveur. |
| **serverConnected** | Nombre de fois que le serveur c’est connecté. |
| **serverConnectedList** | Liste la date, l’heure, l’IP, et le port des connexions du serveur, ainsi que l’heure et la date de ses déconnexions. |

Statistiques relatives à un fichier

Toutes les fonctions relatives à un fichier prennent son **id** en propriété. Les utilisateurs ne peuvent voir que les statistiques des fichiers qu’ils ont le droit de lire. La propriété **view** décrite précédemment est accessible.

|  |  |
| --- | --- |
| **Fonction** | **Description** |
| **fileDownloaded** | Nombre de fois que le fichier a été téléchargé. |
| **fileStreamed** | Nombre de fois que le fichier a été streamé. |
| **fileRead** | Nombre de fois que le fichier a été lu, c'est-à-dire l’addition de fileDownloaded et de fileStreamed. |
| **fileModified** | Nombre de fois que le fichier a été modifié. |

Statistiques complètes

Afin de gagner du temps, vous pouvez demander au serveur de retourner d’un coup toutes les statistiques disponibles, en mettant la fonction **all** dans l’url. Vous avez toujours la possibilité de mettre les propriétés from, to, ou view. Elles seront appliquées à toutes les statistiques. Les fonctions statistiques qui retournent des listent ne sont par incluses.

Si vous mettez un id en propriété dans le header, seules les statistiques relatives au fichier seront incluses.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande toutes les statistiques de ces 30 derniers jours, pour le compte désigné dans **view**, ainsi que celles relatives au serveur. La fonction **all** est dans l’url. Cette requête requière d’être administrateur à cause de l’utilisation de la propriété view sur un compte. | STATISTIC **all** SiTP/1.0  From: 30  **view**: f2aa9400-9239-11de-8a39-0800200c9a66 |
| **Réponse** | La réponse liste toutes les statistiques possibles. | SiTP/1.0 200 OK  **downloadedFiles**: 475  **uploadedSize**: 78  **streamedFiles**: 65  …  **serverConnectedRecord**: 12  **serverConnectedTime**: 150 |

## Informations sur le serveur

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | SERVER |
| **Description** | Les clients peuvent obtenir diverse informations sur le serveur via cette méthode. Les informations retournées dépendent de l’environnement sur lequel tourne le serveur. |
| **URL** | / |
| **Droits** | Tout le monde. |

|  |  |
| --- | --- |
| **Information** | **Description** |
| **Version** | La version courante du serveur |
| **Time** | Heure courante du serveur, au format hh:mm |
| **Os** | Le système d’exploitation |
| **Connected** | Temps en minute depuis lequel le serveur est connecté |
| **Name** | Nom du serveur, donné lors de l’installation |
| **Language** | La langue du serveur |
| **Content-types** | Liste les formats dans lesquels peuvent être envoyés les réponses, séparés par des virgules. |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | La requête ne prend aucune propriété. | SERVER / SiTP/1.0 |
| **Réponse** | Les informations sur le serveur sont listées dans le header de la réponse. | SiTP/1.0 200 OK  **version**: 1.0  **Time**: 13:28  **OS**: Windows 7 64-bit  **Connected**: 142  **Name**: Maison  **Language**: fr  **Content-types**: application/xml, application/json |

## Gestion du serveur

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | SERVER |
| **Description** | Permet de gérer le serveur |
| **Fonctionnalités** | * Afficher / modifier la configuration du serveur * Arrêter / redémarrer le serveur |
| **Droit** | Administrateur |

### Configuration

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **URL** | Configuration |
| **Description** | Permet de lire ou modifier la configuration du serveur. |
| **Corps** | Si les propriétés get, set, remove, et count ne sont pas définis dans la requête, le corps de la réponse contiendra toute la configuration au format XML. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Get** | Permet de lire la valeur d’un nœud ou d’un attribut de l’XML de configuration. Il est possible de cumuler les propriétés get afin de récupérer plusieurs valeurs en même temps. Voici des exemples de valeur :  <node1>  <node2>  <node3>value 1</node3>  <node4 attribut1="value 3">value 4</node4>  </node2>  <node2>  <node3>value 2</node3>  </node2>  </node1>  "**node2[0]/node3**" retournera "**value 1**".  "**node2[1]/node3**" retournera "**value 2**".  "**node2/node4.attribut1**" retournera "**value 3**".  "**node2**" retournera une chaine vide.  Le nœud racine (node1) n’est pas nécessaire. | Requête |
| **Count** | Compte le nombre d’occurrence du nom d’un nœud dans son parent. La syntaxe est la même que pour get, sauf qu’il n’y a pas d’attributs. Le nombre d’occurrence est retourné de la même façon que pour le get. Il est possible de cumuler les propriétés count dans une même requête. | Requête |
| **Set** | Modifie la valeur d’un nœud dans le XML. La syntaxe est la même que pour get, sauf que vous devez rajouter "**=**" suivi de la nouvelle valeur. Si un nœud ou l’attribut n’existe pas, il sera créé. Il est possible de cumuler les propriétés set dans une même requête. | Requête |
| **Remove** | Supprime un nœud ou un attribut du XML. La syntaxe est la même que pour get. Si un nœud ou l’attribut n’existe pas, rien ne se passe. Il est possible de cumuler les propriétés remove dans une même requête. | Réponse |
| **[Valeur Get/Count]** | Contient le résultat d’une propriété get de la requête. Le nom de cette propriété est la même que la valeur de la propriété get dont elle a la valeur. Si le nœud demandé par un get de la requête n’existe pas dans le XML, il ne sera pas affiché dans la réponse. | Réponse |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande la langue du serveur, le type de sa base de données, et met son nombre maximum de thread à 10. La valeur toto/tutu n’existant pas, elle ne sera pas présente dans la réponse. Compte aussi le nombre de nœuds "plugin" dans le nœud "plugins". Enfin, le 3éme nœud plugin est supprimé. | SERVER **configuration** SiTP/1.0  **Get**: language  **Get**: database/type  **Get**: toto/tutu  **Get**: database/type  **Set**: maxThread=10  **Count**: plugins/plugin  **Remove**: plugins/plugin[2] |
| **Réponse** | Affiche les valeurs demandées. | SiTP/1.0 200 OK  **Language**: fr  **Database/type**: SQLITE  **plugins/plugin**: 42 |

Il est important de signaler que la plus part des modifications apportées au fichier XML via cette méthode ne seront appliquées qu’au prochain démarrage du serveur.

### Arrêter / Redémarrer

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **URL** | Stop, restart |
| **Description** | Arrête ou redémarre le serveur. Le serveur ne commencera la procédure d’arrêt qu’une fois la réponse à cette requête envoyée. Cet arrêt n’est pas immédiat puisque tous les threads doivent interrompre leurs opérations le plus proprement possible. |

## Plugins

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | PLUGIN |
| **Description** | Cette fonctionnalité permet aux clients de manipuler les plugins présents sur le serveur. |
| **Fonctionnalités** | * Lister les plugins désinstallés, installés, et chargés * Afficher / modifier la configuration d’un plugin * Charger / décharger un plugin * Installer / Désinstaller un plugin |
| **Droit** | Administrateur |

### 

### Lister

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **URL** | List |
| **Description** | Permet de lister tous les plugins présents sur le serveur, et d’obtenir leur id, leur nom, ainsi que leur état. |
| **Corps** | Le corps contient la liste des plugins :  <plugins>  <plugin>  <id>SiTP</id>  <name>myPlugin</name>  <version>4.2</version>  <state>loaded</state>  </plugin>  <plugin>  <id>UPnP</id>  <name>UPnP</name>  <version>4.2</version>  <state>unloaded</state>  </plugin>  </plugins>  L’**id** est le nom du dossier contenant le plugin.  **Name** est le nom que le plugin se donne dans son fichier de configuration.  C’est l’id qui est utilisé par le serveur pour manipuler le plugin. Deux plugins peuvent avoir le même nom, mais un id est unique.  **Version** est bien entendu la version du plugin.  **State** est l’état actuel du plugin. Il peut valoir **loaded** s’il est chargé, **unloading** s’il est en cour de déchargement, **unloaded** s’il est installé mais pas chargé, et **uninstalled** s’il n’est pas installé. |

### Configuration

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **URL** | Configuration |
| **Description** | Permet de lire ou modifier la configuration d’un plugin. Fonctionne de la même manière que la configuration du serveur. Voir . |
| **Corps** | Si les propriétés get, set, remove, et count ne sont pas définis dans la requête, le corps de la réponse contiendra toute la configuration au format XML. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Id** | L’id du plugin dont on veut lire ou modifier la configuration. | Requête |
| **Get** | Lit la valeur d’un nœud ou d’un attribut. | Requête |
| **Count** | Compte le nombre d’occurrence du nom d’un nœud dans son parent. | Requête |
| **Set** | Modifie la valeur d’un nœud ou d’un attribut. | Requête |
| **Remove** | Supprime un nœud ou un attribut. | Requête |
| **[Valeur]** | Voir . | Réponse |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande le nom et l’email du plugin SiTP, et modifie sa version en 1.1. La valeur toto/tutu n’existe pas. Elle ne sera donc pas présente dans la réponse. | PLUGIN **configuration** SiTP/1.0  **Id**: sitp  **Get**: name  **Get**: email  **Get**: toto/tutu  **Set**: version=1.1 |
| **Réponse** | Affiche les valeurs demandées. | SiTP/1.0 200 OK  **Name**: SiTP  **Email**: team@stream-it.fr |

### Charger / Décharger / Installer / Désinstaller

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **URLs** | load, unload, Install, uninstall |
| **Description** | Ces commandes permettent respectivement de charger, décharger, installer, et désinstaller un plugin. Le serveur n’attend pas que l’opération soit terminée pour retourner sa réponse. Charger un plugin non installé l’installera automatiquement. A l’inverse, désinstaller un plugin chargé le déchargera. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Id** | L’id du plugin sur lequel sera effectuée l’opération. | Requête |

## Télécharger la base de données

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | SELECT |
| **Description** | Il est possible de télécharger directement toute la base de données du serveur. L’objectif principal de cette méthode est d’éviter aux clients mobiles de faire trop de requêtes sur le réseau, ce qui est gourmand en termes de batterie et de bande passante. |
| **URL** | Database |
| **Corps** | La réponse est au format JSON ou XML en fonction du Content-type de la requête, et sous la forme suivante :  <database>  <tableName>  <**action**>  <fieldName>fieldValue</fieldName>  </**action**>  </tableName>  </database>  Remplacez tableName, fieldName, et fieldValue respectivement par le nom de la table, le nom du champ, et la valeur du champ. Le nœud **action** est à remplacer par l’opération à effectuer pour l’entrée. |
| **Droit** | Seules les données auxquels l’utilisateur a accès lui seront envoyées. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Update** | La propriété update indique au serveur de lister les modifications apportées à la base de données depuis la date qu’elle contient. Sa valeur est au format "**yyyy-mm-dd hh:mm:ss**". | Requête |
| **Table** | Nom de la table à télécharger. Il est possible d’indiquer plusieurs tables en les séparant pas des virgules, ou en mettant plusieurs fois cette propriété. Si elle n’est pas définie ou qu’elle vaut "**\***", toutes les tables de la base de données seront retournées. | Requête |
| **Content-type** | Décrit le format dans lequel sera le corps de la réponse. Peut valoir JSON ou XML. | Requête |

Dans le corps de la réponse, chaque entrée est associée à une **action**, qui peut valoir added, deleted, ou modified :

* **Added** indique qu’il s’agit d’une nouvelle entrée. Tous les champs des entrées sont affichés,
* **Modified** indique que l’entrée a été modifiée. Tous les champs des entrées modifiées sont affichés,
* **Deleted** indique qu’elle a été supprimée. Seul le champ id des entrées supprimées est retourné.

Si update n’est pas définie ou à une date invalide, le serveur retournera toute la base de données avec des actions added sur chaque entrées. De cette manière, les clients peuvent garder une copie de la base de données en local, et la garder à jour de manière légère.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande les mises à jour de la base de données à partir du 2010-03-21. | SELECT **database** SiTP/1.0  **Update**: 2010-03-21 17:44:01  Content-type: application/xml |
| **Réponse** | Retourne les modifications apportées à la base de données. On peut voir qu’un fichier a été supprimé depuis la dernière mise à jour. | SiTP/1.0 200 OK  Content-length: 99  Content-type: application/xml  <database>  <files>  <**deleted**>  <id>f2…66</id>  </**deleted**>  </tableName>  </database> |

## Permissions

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | PERMISSION |
| **Description** | Permet de savoir si un utilisateur dispose de la permission d’effectuer une action sur un objet. Par exemple, cela peut être utilisé pour savoir si l’utilisateur courant (celui avec lequel c’est connecté le client) peut modifier un fichier. |
| **URL** | L’**id** de l’objet dont on doit vérifier les permissions. Il est également possible de mettre le chemin de l’objet dans l’arborescence virtuelle du serveur, avec ses **dossiers** séparés par des "**/**" en partant de la racine du serveur, suivi de son **nom**. Si l’objet est une collection, son id est obligatoire. |
| **Droit** | Tout le monde, sauf si l’id d’un accessor est précisé. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Right** | Nom du droit à vérifier. Peut valoir **read**, **add**, **modify**, ou **delete**. | Requête |
| **IdAccessor** | Permet de préciser pour quel accessor (un compte ou un groupe) on doit vérifier le droit. Seuls les administrateurs peuvent utiliser cette propriété. Par défaut c’est l’utilisateur courant qui est utilisé. | Requête |
| **Granted** | Indique si la permission est accordée (***true***), ou refusée (***false***). | Réponse |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande si l’utilisateur courant peut supprimer la vidéo nature.avi. | PERMISSION **video/nature.avi** SiTP/1.0  **Right**: delete |
| **Réponse** | La permission de supprimer le fichier est accordée. | SiTP/1.0 200 OK  **Granted**: true |

## Push

**Cette méthode est toujours en cour de spécification.**

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | PUSH |
| **Description** | Le push désigne un échange que le serveur initie. Le but est d’informer le client d’événements qui se produisent sur le serveur en temps réel. Pour SiTP, cela peut être la connexion d’un utilisateur, la mise à jour de la base de données, … Pour utiliser cette fonctionnalité, le client doit se connecter au serveur, s’identifier (connexion directe ou indirecte), puis lui envoyer la méthode PUSH. Ensuite, le client doit conserver la connexion ouverte, et écouter les réponses du serveur. A chaque fois qu’un événement se produit, ce dernier enverra une réponse le décrivant. Le client ne doit pas envoyer de requêtes sur cette connexion (elles ne seront pas traitées). |
| **URL** | Aucune, c'est-à-dire mettre "/" |
| **Droit** | Les événements envoyés dépendent des droits de l’utilisateur. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **From** | Permet d’être informé de tous les événements qui se sont produit depuis une date déterminée. Sa valeur est au format "**yyyy-mm-dd hh:mm:ss**". Le serveur enverra tous les événements qui se sont produit depuis cette date dans le corps de la réponse au format XML ou JSON en fonction du Content-type dans la requête. | Requête |
| **Events** | **Noms** des événements pour lesquels le client souhaite souscrire. Chaque nom est séparé par une **virgule**. Les événements possibles sont définis plus bas. Si cette propriété n’est pas définie, tous les événements seront envoyés au client. | Requête |
| **Table** | Permet de spécifier les tables dont on souhaite surveiller les modifications via l’événement DatabaseModified. Les tables doivent être séparées par des virgules. Si cette propriété n’est pas définie, toutes les tables seront surveillées. | Requête |
| **Content-type** | Décrit le format dans lequel sera le corps de la réponse. Peut valoir JSON ou XML | Requête |
| **Name** | Nom de l’événement | Réponse |
| **[Event]** | Le nom des propriétés de la réponse correspondent aux informations sur l’événement envoyé par le serveur. Ces informations dépendent des événements. | Réponse |

Si la propriété **From** est définie et sa date valide, la première réponse du serveur contiendra tous les événements qui se sont produit depuis cette date. Ensuite le serveur enverra les événements qui se produisent normalement. Le corps de la réponse au From est de la forme suivante :

<events>

<eventName1 date="yyyy-mm-dd hh:mm:ss" property1="value1" property2="value2" />

<eventName2 date="yyyy-mm-dd hh:mm:ss" property1="value1" property2="value2" />

</events>

EventName est ne nom de l’événement. Les propriétés sont des informations supplémentaires sur l’événement. Certaines de ces propriétés requièrent des droits spécifiques pour être reçus.

Voici la liste des événements que le serveur peut retourner. D’autres seront probablement ajoutés dans le futur.

|  |  |
| --- | --- |
| **Evénements** | |
| **AccountConnected** | Un utilisateur c’est connecté. L’uuid de l’utilisateur est en propriété. |
| **AccountDisconnected** | Un utilisateur c’est déconnecté. L’uuid de l’utilisateur est en propriété. |
| **DatabaseModified** | La base de données a été mise à jour. La propriété table indique le nom de la table modifiée (si possible). |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande au serveur d’être informé lorsqu’un utilisateur se connecte ou se déconnecte du serveur. | PUSH **/** SiTP/1.0  Events: **AccountConnected**, **AccountDisconnected** |
| **Réponse** | Un utilisateur vient se connecter. Son uuid est indiqué dans la propriété uuid. | SiTP/1.0 200 OK  Name: **AccountConnected**  **Uuid**: f2aa9400-9239-a39-0800200c9a66 |
| **Réponse** | L’utilisateur c’est déconnecté. | SiTP/1.0 200 OK  Name: **AccountDisconnected**  **Uuid**: f2aa9400-9239-a39-0800200c9a66 |

## Téléchargement

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | GET |
| **Description** | Télécharge un fichier depuis le serveur. Il est possible de lui demander de convertir le fichier dans un autre format avant de le télécharger. Ceci n’est possible que si le serveur dispose des outils pour convertir le format d’origine du fichier vers celui demandé par le client. L’objectif est de permettre aux clients de consulter des fichiers qu’ils ne peuvent nativement pas lire. |
| **URL** | L’**id** du fichier. Il est également possible de mettre son chemin dans l’arborescence virtuelle du serveur, avec ses **dossiers** séparés par des "**/**" en partant de la racine du serveur, suivi de son **nom**, et de son **extension**. |
| **Corps** | Le corps de la réponse contient le fichier. |
| **Droit** | L’utilisateur doit pouvoir lire le fichier. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Convert** | Demande au serveur de convertir le fichier avant de l’envoyer. Le serveur doit disposer des plugins nécessaires pour réaliser l’opération. Suivant les cas, il est possible que le serveur envoie le fichier en même temps qu’il le converti. Sinon, le fichier est envoyé à la fin de la conversion. Le contenu de cette propriété est le type MIME de destination.  Pour connaitre les types disponibles pour le fichier ciblé, mettez la valeur **list**. La réponse sera la liste des types MIME au format XML ou JSON. | Requête |
| **Content-type** | Type MIME du fichier dans le corps. | Réponse |
| **Content-length** | La taille du corps, et donc du fichier, en octet. | Réponse |

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | L’id en url est celui d’un fichier. La vidéo ne sera pas convertie avant d’être envoyée car la propriété **convert** n’est pas définie. | GET **f2aa9400-9239-11de-8a39-0800200c9a66** SiTP/1.0 |
| **Réponse** | Le fichier est une vidéo de 182694 octets. Son contenu est dans le corps de la réponse. | SiTP/1.0 200 OK  **Content-type**: video/x-msvideo  **Content-length**: 182694  Le contenu du fichier |

## Envoie

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | PUT |
| **Description** | Envoie un fichier sur le serveur |
| **URL** | L’id du **dossier** de destination du fichier à envoyer, ou son chemin virtuel depuis la racine, suivi d’un "**/**", puis du nom du **fichier** avec son **extension**. Pour envoyer un fichier à la racine du serveur, ne mettez pas de dossier, seulement le nom du fichier. |
| **Corps** | Le corps de la requête contient le fichier. |
| **Droit** | Dépend de la propriété ifExists. |

Le fait d’envoyer un nouveau fichier ajoutera automatiquement une nouvelle entrée à la table files, et l’utilisateur en deviendra propriétaire. Si un fichier est remplacé ou modifié, il ne change pas de propriétaire.

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **ifExists** | Dans le cas ou le fichier envoyé existe déjà, cette propriété permet de définir le comportement du serveur. Si sa valeur vaut **replace**, le fichier du serveur sera remplacé. Si elle vaut **append**, le nouveau fichier sera mis à la fin de l’ancien. La valeur **rename** demande au serveur de changer le nom du nouveau fichier (par exemple en ajoutant un numéro à la fin de son nom). Le nouveau nom du fichier est retourné dans la propriété **name** de la réponse. Le client peut aussi décider de changer lui même nom du nouveau fichier, en y ajoutant un numéro par exemple. Si ifExists n’est pas définie et qu’il y a un conflit, le fichier n’est pas envoyé. | Requête |
| **Content-type** | Type MIME du fichier envoyé. Est optionnel. | Requête |
| **Content-length** | Taille du corps, et donc du fichier, en octet. | Requête |
| **Name** | Si la propriété **ifExists** vaut **rename**, cette propriété contiendra le nom du fichier envoyé sur le serveur. | Réponse |

L’envoi d’un fichier vers le serveur requière des droits spécifiques. Tout d’abord, pour créer un nouveau fichier, l’utilisateur doit avoir le droit d’ajouter un fichier dans le dossier ciblé. Pour effectuer les actions d’ifExists, l’utilisateur doit avoir le droit de modifier le fichier du serveur.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Envoi une vidéo de 182694 octets sur le serveur à l’emplacement indiqué dans l’url. Le fichier se nommera **toto.avi**. Si le fichier existe déjà sur le serveur, il sera remplacé par celui qu’on envoi, car la propriété **ifExists** vaut **replace**. | PUT **f2aa9400-9239-11de-8a39-0800200c9a66**/**toto.avi** SiTP/1.0  **IfExists**: **replace**  Content-type: video/x-msvideo  **Content-length**: 182694  Le contenu du fichier |
| **Réponse** | La réponse indique simplement que tout c’est bien passé, ou qu’une erreur c’est produire. | SiTP/1.0 200 OK |

## Streaming

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | STREAM |
| **Description** | Permet de streamer une vidéo ou une musique dans le format et à la qualité voulue par le client. |
| **URL** | L’**Id** du fichier. Il est également possible de mettre son chemin dans l’arborescence virtuelle du serveur, avec ses **dossiers** séparés par des "**/**" en partant de la racine du serveur, suivi de son **nom**, et de son **extension**.  Ce fichier doit être une vidéo ou une musique lisible par le serveur. |
| **Corps** | Le corps contient le fichier streamé. Typiquement, il est lu en même temps qu’il est reçu par le client. |
| **Droit** | L’utilisateur doit pouvoir lire le fichier |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Format** | Nom de l’extension du format qui contient la vidéo et/ou l’audio (avi, mkv, mp4, mp3, 3gp, ogg…). Si le format n’est pas défini, celui du fichier source est conservé. | Requête |
| **VideoCodec** | Codec du flux vidéo. La valeur copy permet de conserver le codec d’origine. | Requête |
| **AudioCodec** | Codec du flux audio. | Requête |
| **VideoBitRate** | Débit des données de la vidéo en bits par seconde. | Requête |
| **AudioBitRate** | Débit des données de l’audio en bits par seconde. | Requête |
| **VideoFrameRate** | Nombre de frame par seconde. | Requête |
| **AudioFrequency** | Fréquence de la piste audio. | Requête |
| **Seek** | Temps de début du flux en seconde. | Requête |
| **Duration** | Durée de la vidéo en seconde. | Requête |
| **x264** | Si l’extension FFmpeg est installée sur le serveur, cette propriété permet d’encoder la vidéo en h.264 via des profils prédéfinis. La propriété **VideoCodec** est ignorée si x264 est définies. Il est toujours possible de jouer avec le bitRate, le frameRate. Voici la liste des profils disponibles :   |  |  | | --- | --- | | * *faster* * *ultrafast* * *superfast* * *veryfast* * *fast* * *medium* * *placebo* * *slow* | * *veryslow* * *slower* * *lossless\_max* * *lossless\_ultrafast* * *lossless\_fast* * *lossless\_medium* * *lossless\_slow* * *lossless\_slower* |   Avec certains profils (comme slower) et en fonction de l’ordinateur, la conversion d’une seconde de vidéo met plus d’une seconde, ce qui empêche le streaming. | Requête |
| **Width / Height** | La largeur et la hauteur de la vidéo. Si l’un des deux n’est pas défini, elle sera proportionnelle à l’autre (le ratio est préservé). Leurs valeurs doivent être paires. | Requête |
| **Content-length** | Taille du corps, en octet | Réponse |
| **Content-type** | Type MIME du corps | Réponse |

Les valeurs par défaut des propriétés non définies dans la requête privilégient la qualité sur la taille, ce qui peut mener à des vidéos plutôt lourdes. C’est donc au client de dégrader la vidéo en fonction de ses besoins et de ses possibilités.

Manipuler le bitRate permet de réduire la taille de la vidéo ou de la musique, au détriment de sa qualité. De même, Width et height permettent de réduire la taille de la vidéo, et donc son volume.

Afin d'éviter que le streaming ne prenne tout le CPU du serveur en convertissant d'un coup toute une vidéo, les clients sont encouragés à découper les vidéos en petits morceaux de 10 secondes, qui seront générés régulièrement. Le client les télécharge un par un, et les affichent comme étant une seule et même vidéo.

De plus, ceci permet aux clients de diminuer le volume de ces morceaux, en abaissant leur bit rate (**bitRate**), et en réduisant leur résolution (**width**/**height**).

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande au serveur de lui envoyer les 10 premières secondes de la vidéo **video.avi**, au format h264 avec un bitRate de 1 000 000. Le morceau n’est envoyé qu’une fois qu’il a été totalement converti. | STREAM **video.avi** SiTP/1.0  **X264** : faster  **BitRate** :1000000 |
| **Réponse** | Le fichier est une vidéo de 182694 octets de 10 secondes. Son contenu est dans le corps de la réponse. | SiTP/1.0 200 OK  **Content-type**: video/mp4  **Content-length**: 182694  Le contenu du fichier |
| **Requête** | Une fois le morceau reçu, le client le lit sur son lecteur, et demande le morceau suivant (le client peut aussi commencer à lire le premier morceau pendant qu'il le reçoit). | STREAM **video.avi** SiTP/1.0  **X264** : faster  **Seek**: 10  **Duration**:10 |
| **Réponse** | Le morceau suivant de 10 secondes | SiTP/1.0 200 OK  **Content-type**: video/mp4  **Content-length**: 182694  Le contenu du fichier |
| **Requête** | Si le client n'a pas beaucoup d'avance sur la lecture, et s'aperçoit que la qualité de la connexion se dégrade, il diminue le bit rate et si nécessaire la résolution des prochains morceaux, afin de garantir la continuité de la lecture, jusqu'à ce que le débit soit meilleur. Certains clients peuvent préférer garder une qualité de vidéo optimale, et demander à l’utilisateur de patienter le temps de prendre de l’avance sur la lecture. | STREAM **video.avi** SiTP/1.0  **X264** : faster  **Seek**: 20  **Duration**:10  **VideoBitRate**: 100000  **Width**: 400  **Height**: 300 |
| **Réponse** | Ce morceau a une qualité réduite. | SiTP/1.0 200 OK  **Content-type**: video/mp4  **Content- length**: 182694  Le contenu du fichier |

## Aperçu

|  |  |
| --- | --- |
| **Caractéristiques** | |
| **Méthode** | PREVIEW |
| **Description** | Cette méthode permet de télécharger l’aperçu d’un fichier. Un aperçu est une image miniature illustrant un fichier. Il peut être généré pour une image, une vidéo, un texte, voir n’importe quel fichier, du moment que le serveur en est capable. |
| **URL** | L’**Id** du fichier. Il est également possible de mettre son chemin dans l’arborescence virtuelle du serveur, avec ses **dossiers** séparés par des "**/**" en partant de la racine du serveur, suivi de son **nom**, et de son **extension**. |
| **Corps** | Le corps de la requête contient l’aperçu. |
| **Droit** | L’utilisateur doit pouvoir lire le fichier. |

|  |  |  |
| --- | --- | --- |
| **Propriétés** | | |
| **Content-type** | Format d’image dans lequel l’aperçu sera envoyé. Il peut valoir JPEG, PNG, GIF, TIFF, BMP, ou PDF. Le type MIME peut également être fourni : image/jpeg, image/png, image/gif, image/tiff, image/bmp, ou application/pdf. Le format par défaut est JPEG. | Requête |
| **Width** | Largeur de l’aperçu en pixel. | Requête |
| **Height** | Hauteur de l’aperçu en pixel. | Requête |
| **Content-type** | Type MIME du fichier reçu. Il correspond au format demandé dans le content-type de la requête. | Réponse |
| **Content-length** | Taille du corps, en octet. | Réponse |

Si width ou height n’est pas défini, il sera évalué en fonction de l’autre, de manière à garder les proportions originales du fichier. Par défaut, la taille de l’aperçu est de 100x75 pixels. Si le serveur ne trouve pas de moyen de générer l’aperçu, une erreur est retournée. La taille maximale est de 1024x768 pixels.

|  |  |  |
| --- | --- | --- |
| **Exemple** | | |
| **Requête** | Demande l’aperçu de l’image river.jpg au format png, avec une dimension de 400x300 pixels. | **PREVIEW** France/Paris/river.jpgSiTP/1.0  **Width**: 400  **Heigth**: 300  **Content-type**: png |
| **Réponse** | Le serveur retourne une image au format png de 128152 octets avec une résolution de 400x300 pixels. | SiTP/1.0 200 OK  **Content-type**: png  **Content-length**: 128152  Le contenu de l’image |

# Erreurs

Cette partie liste des codes d’erreur possible pour toutes les fonctionnalités de SiTP. Le seul code qui n’est pas une erreur est **200**. Son message est **OK**, et indique que tout c’est bien passé. Ces erreurs correspondent avec celles de HTTP, mais avec quelques modifications.

Le serveur est libre de décrire plus précisément l’erreur dans le corps de la réponse.

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Message** | | **Description** |
| **400** | **Bad Request** | La requête ne peut pas être comprise par le serveur à cause d’une erreur de syntaxe. Le client ne doit pas répéter la requête sans modification. | |
| **401** | **Unauthorized** | L’identification n’a pas été autorisée par le serveur. Ceci peut être dû à un mauvais mot de passe, ou à un mauvais identifiant. | |
| **402** | **Banned** | Lorsque deux erreurs **401** provenant d’un client non connecté avec la même IP sont retournées en moins de cinq minutes, et qu’une troisième est générée, l’erreur **402** est retournée à la place. Elle indique au client ne plus essayer de se connecter durant la période indiquée dans la propriété **Time** du header en secondes. | |
| **403** | **Forbidden** | Le client n’a pas les droits nécessaires à l’exécution de sa requête. | |
| **404** | **Not Found** | La ressource demandée dans l’url n’a pas été trouvée. | |
| **405** | **Unknown property** | L’une des propriétés du header est inconnue. | |
| **406** | **Missing Information** | Le serveur estime qu’une information lui manque pour d’exécuter la requête. | |
| **500** | **Internal Server Error** | Le serveur a rencontré une condition inattendue qui l’empêche de traiter la requête. | |
| **501** | **Not Implemented** | Le serveur ne supporte pas la fonctionnalité requise pour traiter la requête. C’est le cas lorsque le serveur n’implémente pas la méthode demandée par le client. | |
| **505** | **Version Not Supported** | La version de SiTP utilisée par le client est différente de "**SiTP/1.0**". La version 1.0 n’est pas compatible avec les versions antérieures. | |