IoTSocket - Protocole (v1.01)

Définition d'un protocole de communication ouvert et dédié au domaine de l'IoT.

Objectif du protocole

- S'inscrire comme une solution de communication pensée pour les objets connectés.
- Combler les manquements rencontrés dans l'exploitation des architectures IoT actuelles.
- Répondre aux dernières exigences en matière de transmission et de sécurité des données.
- Tenir compte des contraintes impliquées par l'aspect mobilité et embarqué des objets.
- Faciliter les interconnexions et les échanges avec les applications d'aujourd'hui.
- Offrir une stratégie d'implémentation rapide par le biais de spécifications ouvertes.

Apport du protocole

- Support de différents types d'éléments clés constituant un réseau orienté IoT.
- Etablissement de connexions entre initiateurs et points de terminaison accessibles.
- Transport s'appuyant sur des liaisons persistantes bidirectionnelles (full-duplex).
- Prise en charge de redirection de connexions si besoin pour routage ou équilibrage.
- Echanges sécurisés par l'utilisation transparente du protocole standardisé TLS.
- Authentification des connexions avec maintient et reprise de sessions actives.
- Agrégation et orchestration des flux au moyen de concentrateurs intermédiaires.
- Echanges par logique de requêtes-réponses ou par paquets de télémétrie.
- Données structurées au format JSON ou bien directement en mode binaire.
- Stratégie de contrôle des flux pour la gestion différée des sessions non connectées.
- Architecture facilitant la prise en charge d'APIs Web ainsi que la scalabilité.

SAS Hyper Connect Concept (HC²)
57 rue de la Division Leclerc, 78830 Bonnelles
CAPITAL 80 000 € - SIRET 82338914300012 - RCS VERSAILLES

Description de l'élément « Objet »

- Prend en considération les contraintes liées aux ressources et à la communication.
- Etablit une connexion sécurisée ou non vers un élément « Concentrateur ».
- S'identifie sur le réseau et tente de négocier une ouverture ou une reprise de session.
- Prend connaissance des éventuelles règles nécessaires à la suite des échanges.
- Transmet des requêtes vers l'élément « Central » puis reçoit les réponses afférentes.
- Réceptionne des requêtes provenant de l'élément « Central » puis y répond.
- Remonte des paquets de télémétrie vers l'élément « Central ».
- Structure ses données au format JSON ou bien directement en mode binaire.
- Clôture la connexion en suspendant la session ou en y mettant un terme.

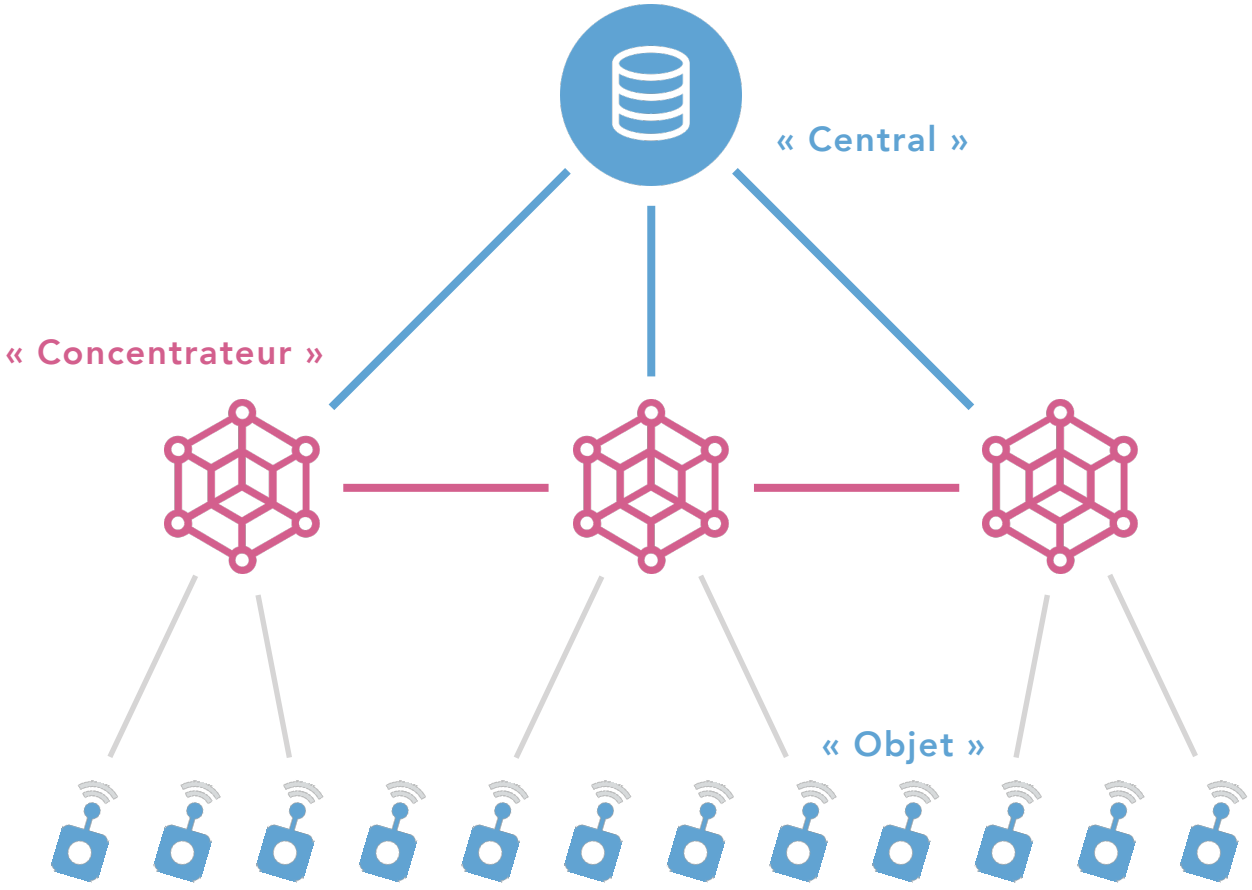
Description de l'élément « Concentrateur »

- Se positionne entre les éléments de type « Central » et « Objet » du réseau.
- Définit différentes règles de connexions et d'échanges de données.
- Maintient une communication sécurisée, persistante ou non, avec l'élément « Central ».
- Prend en charge un cluster composé d'éléments de type « Concentrateur ».
- Traite l'ensemble des connexions, authentifications et ouvertures de sessions.
- Agrège et orchestre les flux selon des stratégies appliquées à différents groupes.
- Permet l'intégration d'une couche de transcodage par application de codecs spécifiques.
- Rend persistantes les sessions déconnectées, par le maintient d'états de flux.
- Supporte un jeu de commandes dédié aux sessions et à la sécurité des authentifications.
- Supporte un jeu de commandes dédié aux échanges d'informations du mode cluster.
- Offre le support de commandes additionnelles dans le cadre d'extensions intégrées.

Description de l'élément « Central »

- S'authentifie et communique de manière sécurisée avec un élément « Concentrateur ».
- Dialogue avec l'élément « Concentrateur » par le biais de commandes.
- Transmet des requêtes vers l'élément « Objet » puis reçoit les réponses afférentes.
- Réceptionne des requêtes provenant de l'élément « Objet » puis y répond.
- Reçoit des données de télémétrie par paquets depuis l'élément « Objet ».
- Structure ses données au format JSON ou bien directement en mode binaire.
- Utilise le jeu de commandes dédié aux sessions et à la sécurité des authentifications.

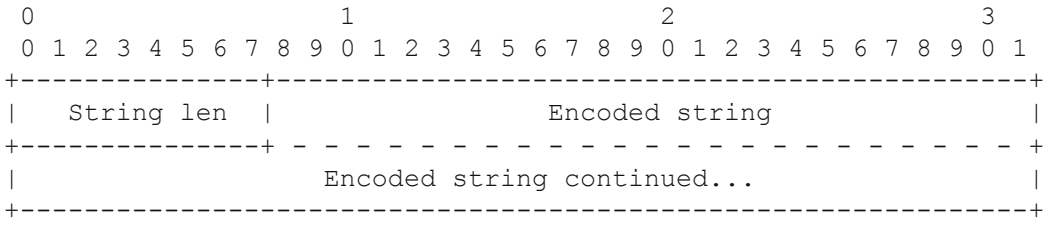
Modèle d'architecture en cluster



Structures binaires des entités utilisées dans les échanges de données

Structure d'une chaîne de caractères UTF-8 à longueur variable

Contient une chaîne de caractères vide ou d'une longueur variable, encodée en UTF-8.



String len : Longueur de la chaîne encodée pouvant aller de 0 à 255.
Encoded string : Nul ou données binaires de la chaîne encodée en UTF-8.

Structure d'un nom de groupe

Représente le nom d'un groupe pouvant contenir un ensemble d'éléments du réseau.
Il s'agit de 16 bytes (128 bits) dont tous les premiers bytes fixés à 0x00 sont ignorés.

Structure d'un identifiant unique

Représente l'identité unique d'un élément du réseau.
Il s'agit de 16 bytes (128 bits) dont tous les premiers bytes fixés à 0x00 sont ignorés.
L'identifiant unique de l'élément « Central » est toujours vide (16 x 0x00).

Structure d'une clé d'authentification

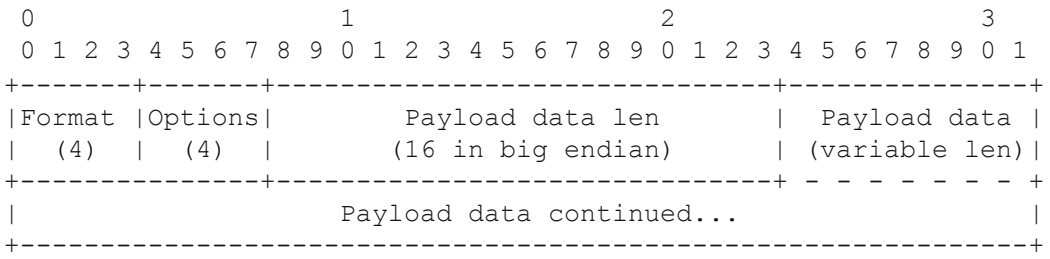
Représente la clé secrète associée à l'identifiant unique d'un élément du réseau.
Cette clé permet de valider une authentification auprès de l'élément « Concentrateur ».
Sa longueur est fixée à 16 bytes (128 bits).

Structure d'un jeton de télémétrie

Représente un jeton unique et temporaire utilisable pour les paquets de télémétrie.
Sa longueur est fixée à 8 bytes (64 bits).

Structure d'un bloc de données utiles

Représente un bloc de données utiles dont le format est spécifié.



Format : Détermine le format des données utiles :
- 0x00 : Aucun format, données binaires brutes.
- 0x01 : Format texte encodé en ASCII.
- 0x02 : Format texte encodé en UTF-8.
- 0x0A : Format JSON encodé en UTF-8.

Options : Options applicables selon le format utilisé.

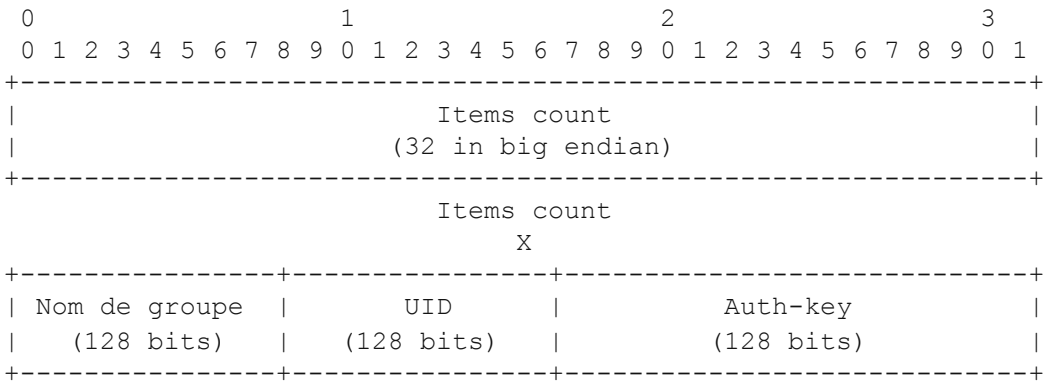
Payload data len : Longueur des données binaires pouvant aller de 0 à 65 535 (2^{16}-1).

Payload data : Nul ou données binaires utiles.

Structure d'une liste d'accès autorisés (ACL)

Représente une liste d'accès autorisés à des éléments de type « Objet »

Chaque accès est définit par un élément, sa clé d'authentification ainsi que son groupe.

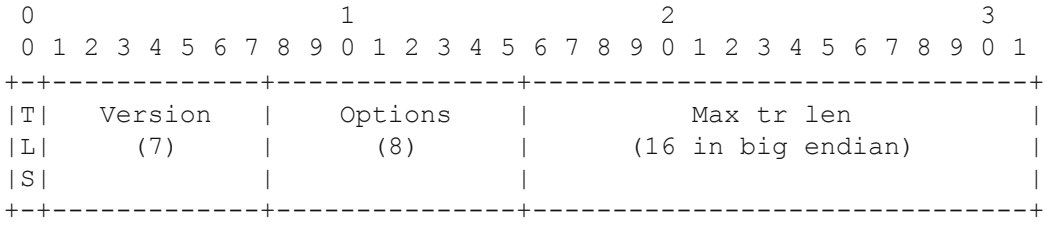


Items count : Définit le nombre d'accès (peut être de zéro).

Structures binaires des échanges de données

Initiation de connexion

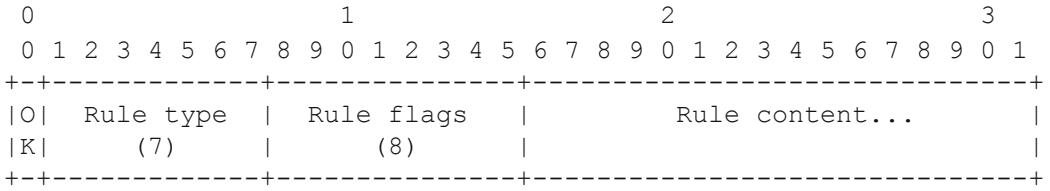
Contient les informations transmises à l'élément « Concentrateur » lors d'une connexion.



TLS : Indique la nécessité d'appliquer le protocole de sécurité TLS.
Version : Indique le numéro de version du protocole IoTSocket : 0x01.
Options : Options utilisables librement selon l'implémentation.
Max tr len : Taille maximum des transmissions pouvant être traitée.

Réponse à initiation de connexion

Contient la règle retournée par l'élément « Concentrateur » après initiation de connexion.



OK : Indique si la connexion est possible ou si elle doit se terminer.
Rule type : Détermine le type de règle dont il s'agit :
- 0x00 : Aucune règle n'est présente.
- 0x01 : Règle de redirection de la connexion.

Rule flags : Drapeaux à prendre en considération selon le type de règle.
Rule content : Contenu selon le type de règle (Rule type) :
- 0x00 : Aucune règle (n'existe pas).
- 0x01 : Nom d'hôte (chaîne de caractères UTF-8).
  Numéro de port (16 bits big endian).

Demande de challenge d'authentification

Contient une suite de 16 bytes (128 bits) générée par l'élément « Concentrateur » et correspondant au challenge d'authentification demandé.

```
+-----------------------------------------------------------------+
|                            Challenge                            |
|                            (128 bits)                           |
+-----------------------------------------------------------------+
```

Réponse à demande de challenge d'authentification

Représente la réponse au challenge d'authentification demandé par l'élément « Concentrateur » afin de permettre ou non l'ouverture d'une session sécurisée.
Contient un identifiant unique suivi des 256 bits correspondant au challenge d'authentification signé par la clé d'authentification.
L'algorithme de signature numérique appliqué est : HMAC-256.
L'identifiant unique correspondant à l'élément « Central » est toujours vide (16 x 0x00).

```
+----------------+-------------------------------------------------+
|       UID      |       HMAC-256(auth-key, auth-challenge)        |
|   (128 bits)   |                  (256 bits)                     |
+----------------+-------------------------------------------------+
```

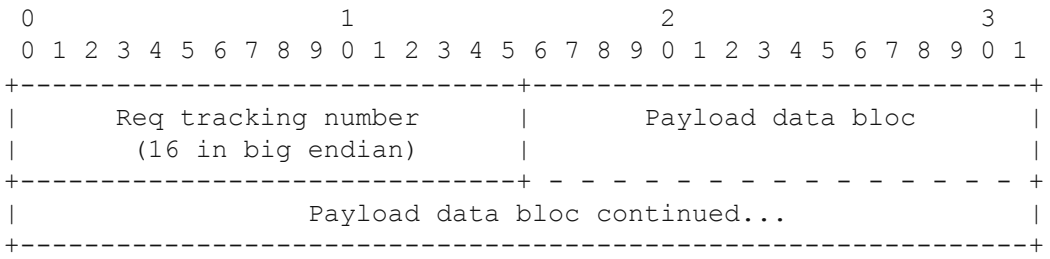
Paquet de télémétrie

Représente des données de télémétrie transmises via un paquet non garanti.
Contient un jeton de télémétrie de 64 bits suivi d'un bloc de données utiles.

```
+----------------+-------------------------------------------------+
|      Token     |              Payload data bloc                  |
|    (64 bits)   |                                                 |
+----------------+-------------------------------------------------+
```

Requête

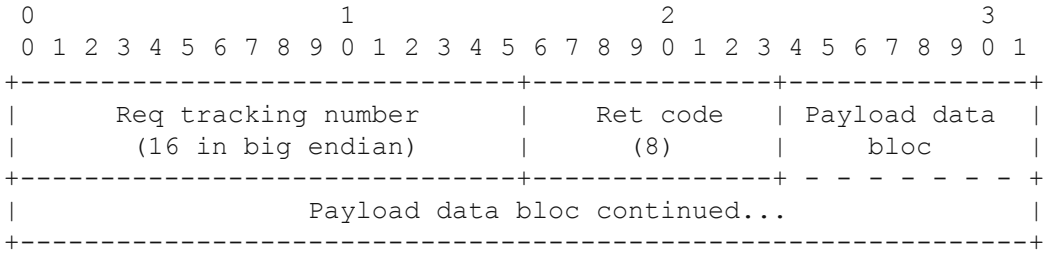
Représente une requête contenant un bloc de données utiles et pouvant être suivie.



Req tracking number : Numéro de suivi correspondant à la requête.
Payload data bloc : Bloc de données utiles (toujours présent).

Réponse à requête

Représente une réponse à requête contenant un bloc de données utiles.



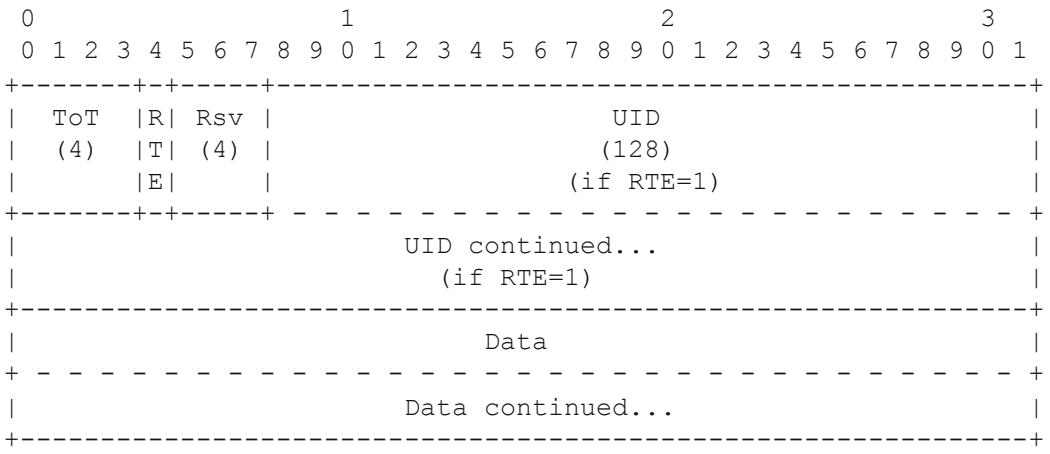
Req tracking number : Numéro de suivi correspondant à la requête ayant précédé.
Ret code : Code de retour indiquant le type de réponse :

- 0x00 : Retour valide.
- 0x01 : Retour non valide.
- 0xA0 : Destination introuvable.
- 0xA1 : Temps maximum écoulé (timeout).
- 0xA2 : Requête identique en cours.

Payload data bloc : Bloc de données utiles (toujours présent).

Transmission de données

Représente la transmission d'un ensemble de données entre deux éléments.



ToT : Indique le type de transmission :
- 0x00 : Liste d'accès autorisés (ACL).
- 0x01 : Ping.
- 0x02 : Pong.
- 0x03 : Requête.
- 0x04 : Réponse à requête.
- 0x05 : Jeton de télémétrie.
- 0x06 : Télémétrie identifiée.
- 0x0F : Clôture de connexion.

RTE : Indique si un UID est impliqué dans la transmission.

Rsv : Réservé.

UID : Identifiant unique impliqué.
N'existe que si RTE vaut 1.

Data : Données selon le type de transmission (ToT) :
- 0x00 : Contient une liste d'accès autorisés.
- 0x01 : Aucune donnée (n'existe pas).
- 0x02 : Aucune donnée (n'existe pas).
- 0x03 : Contient une requête.
- 0x04 : Contient une réponse à requête.
- 0x05 : Contient un jeton de télémétrie.
- 0x06 : Contient un bloc de données utiles.
- 0x0F : Contient un code de clôture (8 bits).

Scénario de connexion vers l'élément « Concentrateur »

1. Choix du protocole de transport :
   - Utilisation d'un protocole de transport de flux en mode connecté (TCP, WebSocket).

2. Envoi d'une initiation de connexion :
   - La nécessité d'utiliser la couche de sécurité TLS est indiquée.
   - TLS est recommandé sauf s'il est déjà appliqué sur le protocole de transport.
   - Le numéro de version du protocole IoTSocket actuel est fixé à 0x01.
   - Les options sont laissées libres et dépendent de l'implémentation.
   - La taille maximum des transmissions pouvant être traitée est précisée ou fixée à zéro.

3. Réception de la réponse à initiation de connexion :
   - Clôture de la connexion dans le cas où OK vaut 0.
   - Selon le type de règle :
     - 0x00 > Aucune règle : Passage à l'étape 4.
     - 0x01 > Redirection : Déconnexion puis connexion vers l'hôte et le port précisés.

4. Application de la couche de sécurité TLS si cela est spécifié à l'étape 2.

5. Réception de la demande de challenge d'authentification :
   - Les bytes du challenge sont générées aléatoirement à chaque fois.
   - Le challenge est mémorisé par l'élément « Concentrateur » pour cette connexion.

6. Envoi de la réponse à demande de challenge d'authentification :
   - L'identifiant unique de l'élément se connectant est précisé.
   - Le résultat du challenge correspond aux bytes aléatoires signées par la clé d'authentification.
   - La clé d'authentification doit être celle correspondant à l'identifiant unique.
   - L'élément « Concentrateur » calcul également le résultat du challenge via l'UID.
   - L'identifiant unique de l'élément « Central » est toujours vide (16 x 0x00).
   - La validation s'effectue en comparant les deux résultats du challenge (client = serveur ?).
   - L'élément « Concentrateur » valide le résultat en envoyant OK (0x01) ou ERREUR (0x00).
   - En cas d'erreur, la connexion est aussitôt fermée par chacune des parties.

7. Si la connexion est valide, la session est créée ou reprise.

Déroulement et obligations d'une connexion valide

- L'élément « Concentrateur » détermine quel élément est connecté de par son identifiant.
- Des transmissions de données peuvent avoir lieu à tout moment et dans les deux sens.
- La réception d'un « Ping » entraîne toujours l'envoie d'un « Pong » en retour.
- Une « Réponse à requête » devrait toujours faire suite au traitement d'une « Requête ».
- Une « Réponse à requête » inclus toujours le numéro de suivi de la « Requête » afférente.
- Seul l'élément « Concentrateur » est apte à envoyer des « Jetons de télémétrie ».
- Un « Jeton de télémétrie » peut être envoyé dès la connexion valide.
- Seul l'élément « Objet » peut recevoir des « Jetons de télémétrie ».
- Seul l'élément « Concentrateur » est apte à envoyer des « Télémétries identifiées ».
- Seul l'élément « Central » peut recevoir des « Télémétries identifiées ».
- L'élément « Objet » envoie et reçoit toujours des transmissions avec RTE=0 (pas d'UID).
- L'élément « Concentrateur » route les « Requêtes » et « Réponses à requêtes » :
  - Entre l'élément « Central » et l'élément « Objet ».
  - Avec comme UID impliqué, celui de l'élément « Objet » expéditeur ou destinataire.
  - En provenance de l'élément « Objet » : Ajout de l'UID (RTE=1).
  - A destination de l'élément « Objet » : Suppression de l'UID (RTE=0).
- L'élément « Central » envoie et reçoit le plus souvent des transmissions avec RTE=1 (UID).
- Lors de la déconnexion d'un élément :
  - Une « Clôture de connexion » est envoyée juste avant de provoquer la déconnexion.
  - Selon le code de clôture, l'élément « Concentrateur » maintiendra ou non la session.
  - L'élément « Concentrateur » maintiendra la session suite à une déconnexion inopinée.

Types de transmissions de données

0x00 : Liste d'accès autorisés (ACL)

- Contient une structure binaire « Liste d'accès autorisés (ACL) ».
- Envoyé uniquement par l'élément « Central » vers l'élément « Concentrateur ».
- Définit l'ensemble des accès autorisés pour les éléments de type « Objet » uniquement.
- Ne contient jamais d'UID (RTE=0).

0x01 : Ping

- Envoyé à l'élément afin de vérifier sa présence et son temps de réponse.
- Ne contient jamais d'UID (RTE=0) ni de données.
- Lors de sa réception, une transmission de données « Pong » doit être envoyée.

0x02 : Pong

- Aussitôt envoyé à l'élément après réception d'une transmission de données « Ping ».
- Ne contient jamais d'UID (RTE=0) ni de données.

0x03 : Requête

- Contient une structure binaire « Requête ».
- Envoyée à un élément dans le but d'exécuter un traitement et d'en obtenir la réponse.
- Contient un UID (RTE=1) se référant à l'élément « Objet » correspondant.
- Le numéro de suivi est toujours fixé par l'élément créateur de la « Requête ».

0x04 : Réponse à requête

- Contient une structure binaire « Réponse à requête ».
- Envoyée comme réponse à l'élément ayant expédié une « Requête » au préalable.
- Contient un UID (RTE=1) se référant à l'élément « Objet » correspondant.
- Le numéro de suivi est toujours identique à celui de la « Requête » afférente.
- Contient un code de retour indiquant le type de réponse :
  - 0x00 > Retour valide : L'élément destinataire a bien retourné cette réponse.
  - 0x01 > Retour non valide : L'élément destinataire n'a pas répondu correctement.
  - 0xA0 > Destination introuvable : La requête n'a jamais pu être transmise à l'élément.
  - 0xA1 > Timeout : La requête n'a pas donné lieu à une réponse dans le délai imparti.
  - 0xA2 > Requête identique en cours : Le même n° de tracking est en cours d'utilisation.

0x05 : Jeton de télémétrie

- Contient une structure binaire « Jeton de télémétrie ».
- Envoyé uniquement par l'élément « Concentrateur » vers l'élément « Objet ».
- Peut être envoyé aussitôt la connexion validée.
- Permet à un élément « Objet » de transmettre des paquets de télémétrie.
- Peut être envoyé de nouveau en cas de renouvellement.
- Lors de sa réception, les paquets de télémétrie suivants devront y faire référence.

0x06 : Télémétrie identifiée

- Contient un bloc de données utiles, formaté et à longueur variable.
- Envoyé uniquement par l'élément « Concentrateur » vers l'élément « Central ».
- Embarque l'UID de l'élément « Objet » ayant remonté la télémétrie.

0x0F : Clôture de connexion

- Contient un code de clôture 8 bits.
- Envoyé juste avant toute déconnexion voulue.
- Selon le code de clôture, la session peut être maintenue pour reprise ultérieure.
- Codes de clôture :
  - 0x00 > Clôture pour erreur de protocole, session fermée.
  - 0x01 > Clôture pour intervention de maintenance, session fermée.
  - 0x02 > Clôture pour nombre trop élevé de transmissions à traiter, session fermée.
  - 0x03 > Clôture pour erreur de traitement interne, session fermée.
  - 0xA0 > Clôture pour mise en sommeil temporaire, session maintenue.
  - 0xA1 > Clôture pour libération de ressources, session maintenue.

Remontée d'un paquet de télémétrie

1. Choix du protocole de transport :
   - Utilisation d'un protocole de transport de paquets en mode déconnecté (UDP).
   - Le paquet transmis n'a pas pour vocation à être garanti et peut être perdu.

2. Remontée d'un paquet de télémétrie :
   - Contient une structure binaire « Paquet de télémétrie ».
   - Seul l'élément « Objet » peut transmettre un paquet de télémétrie.
   - Seul l'élément « Concentrateur » peut recevoir un paquet de télémétrie.
   - Embarque le dernier jeton de télémétrie reçu depuis l'élément « Concentrateur ».
   - Transporte toujours un bloc de données utiles, formaté et à longueur variable.

Maintient d'une session par l'élément « Concentrateur »

- Seul l'élément « Concentrateur » est apte à maintenir une session.
- Une session est maintenue après la déconnexion d'un élément.
- Une session maintenue peut être reprise ultérieurement sous un certain délai.
- Une session maintenue garde les « Requêtes » et « Réponses à requête » en attente.
- Lors de la reprise d'une session maintenue, les transmissions gardées sont envoyées.
- Lors d'une clôture de connexion et selon le code, la session sera maintenue ou non.
- Une déconnexion inopinée entraine toujours le maintient de la session.

Présentation de l'extension IoTSocket sur HTTP(S)

Description

IoTSocket a la capacité de se placer au dessus du protocole HTTP(S) au niveau de la connexion reliant l'élément « Central » à l'élément « Concentrateur ».
Le protocole HTTP(S) définissant déjà une logique de requêtes-réponses encapsulant des données dont la taille est précisée, le mode de transmission ne peut donc pas être identique à celui reposant sur une couche de transport de flux tel que TCP.
Il est alors essentiel que les techniques de transmissions décrites par IoTSocket adhèrent entièrement au schéma d'échanges que propose HTTP(S).
Par ailleurs, IoTSocket sur HTTP(S) permet de se rapprocher de méthodologies employées couramment dans le domaine des applications Web mais surtout, celui des APIs Web.
Son objectif principal est alors d'abstraire les mécanismes souvent complexes des protocoles bas-niveau afin de répondre à une intégration facilitée de plus haut-niveau.

Principe de fonctionnement

IoTSocket sur HTTP(S) se présente comme une API Web, offrant alors la possibilité d'effectuer des requêtes depuis l'objet « Central » vers l'objet « Concentrateur », mais également, permettant l'inverse au moyen de WebHooks depuis l'objet « Concentrateur » vers l'objet « Central ».
Il n'y a donc pas dans ce cas présent de logique évènementielle en transmissions continues, mais directement l'exécution de requêtes délivrant chacune la réponse adéquate dans un même processus.
Aussi, l'authentification par négociation lors de l'ouverture d'une connexion ne peut plus être employée dans ce contexte et laisse place à l'utilisation d'un mode « clé d'API », accompagnant chaque requête ou WebHook.

Performances et sécurité

IoTSocket sur HTTP(S) requiert plus de ressources dans son exploitation du fait qu'il implique l'intégration d'un serveur et d'un client Web ainsi que la gestion intensive de multiples connexions simultanées dont les échanges de données s'opèrent en mode texte et au format JSON.
L'application de la couche TLS (HTTPS) rendra d'autant plus gourmandes les communications mais sera impérative dans le cadre d'une architecture déployée sur un réseau publiquement accessible.

Authentification dans IoTSocket sur HTTP(S)

Requête exécutée vers l'objet « Concentrateur »

Chaque requête exécutée depuis l'objet « Central » vers l'objet « Concentrateur » utilise l'authentification HTTP de type « Bearer », en transmettant dans son en-tête d'autorisation, le jeton d'accès sous forme de chaîne hexadécimale.

WebHook déclenché depuis l'objet « Concentrateur »

Chaque requête exécutée depuis l'objet « Concentrateur » vers l'objet « Central » utilise l'authentification HTTP de type « Bearer », en transmettant dans son en-tête d'autorisation, le jeton d'accès sous forme de chaîne hexadécimale.

Rappel sur l'authentification HTTP de type « Bearer »

Une en-tête HTTP « Authorization » doit être présente et contenir la méthode utilisée « Bearer » suivie du jeton d'accès.
Exemple d'en-tête avec le jeton d'accès « a1b2c3d4e5f6 » :
« Authorization: Bearer a1b2c3d4e5f6 »

SAS Hyper Connect Concept (HC²)
57 rue de la Division Leclerc, 78830 Bonnelles
CAPITAL 80 000 € - SIRET 82338914300012 - RCS VERSAILLES

Requête dédiée à l'ACL dans IoTSocket sur HTTP(S)

URI et méthode HTTP

L'URI est direct et ne contient jamais aucun paramètre « query string ».
La méthode HTTP utilisée pour les requêtes dédiées à l'ACL est « POST ».

Formatage et encodage du contenu

Le corps d'une requête dédiée à l'ACL est au format JSON, encodé en UTF-8.
Le type de contenu « Content-Type » est alors fixé à « application/json ».

Structuration des données

La structuration des données d'une requête dédiée à l'ACL s'apparente à une transmission de données de type « Liste d'accès autorisés (ACL) ».
Les différents paramètres et valeurs de chacun des accès autorisés sont définis dans des objets JSON contenus dans un tableau JSON :
« GroupName » :
- Toujours présent et de type chaîne de caractères.
- Contient le nom du groupe auquel appartient l'élément « Objet ».
- Dont l'encodage UTF-8 ne doit jamais dépasser 16 bytes.

« UID » :
- Toujours présent et de type chaîne de caractères.
- Contient l'identifiant unique de l'élément « Objet » dont l'accès est autorisé.
- Dont l'encodage UTF-8 ne dépasse jamais une longueur de 16 bytes.

- « AuthKey » :
- Toujours présent et de type chaîne de caractères.
- Contient la clé d'authentification de l'élément « Objet » en chaîne hexadécimale.

Réponse HTTP

Dans le contexte d'un traitement correct, le statut de réponse HTTP doit être 200 (OK).
Le contenu de la réponse sera ignoré et doit donc être vide.

Requête et WebHook dans IoTSocket sur HTTP(S)

URI et méthode HTTP

L'URI est direct et ne contient jamais aucun paramètre « query string ».
La méthode HTTP utilisée pour les requêtes et les WebHooks est « POST ».

Formatage et encodage du contenu

Le corps d'une requête ou d'un WebHook est au format JSON, encodé en UTF-8.
Le type de contenu « Content-Type » est alors fixé à « application/json ».

Structuration des données

Qu'il s'agisse d'une requête ou d'un WebHook, la structuration des données est identique et s'apparente à une transmission de données de type « Requête ».
Il n'y a cependant pas de numéro de suivi, le protocole HTTP(S) permettant d'abstraire cette logique de par son mécanisme de requête-réponse contenu dans un même processus.
Les différents paramètres et valeurs sont définis dans un objet JSON :

- « UID » :
  - Obligatoire, non-nul et de type chaîne de caractères.
  - Contient l'identifiant unique de l'élément destinataire/expéditeur.
  - Dont l'encodage UTF-8 ne dépasse jamais une longueur de 16 bytes.
- « Timeout » :
  - Facultatif ou non-nul et de type nombre entier supérieur à zéro.
  - Définit le délai maximum, en secondes, autorisé avant le retour de la réponse.
- « Payload » :
  - Obligatoire, non-nul et de type spécifique selon le format précisé.
  - Contient les données utiles structurées au format précisé.
  - Doit contenir n'importe quel objet JSON si « Format » vaut « JSON ».
  - Doit contenir une chaîne de caractères si « Format » vaut « ASCII » ou « UTF8 ».
  - Doit contenir un tableau de valeurs de bytes si « Format » vaut « BINARY ».
- « Format » :
  - Obligatoire, non-nul et de type chaîne de caractères.
  - Indique la manière dont les données « Payload » sont à traiter.
  - Peut prendre les valeurs « JSON », « ASCII », « UTF8 » ou « BINARY ».

Réponse à requête et à WebHook dans IoTSocket sur HTTP(S)

Réponse HTTP

Dans le contexte d'un traitement correct, le statut de réponse HTTP doit être 200 (OK).

Formatage et encodage du contenu

Le corps d'une réponse à requête ou à WebHook est au format JSON, encodé en UTF-8.
Le type de contenu « Content-Type » est alors fixé à « application/json ».

Structuration des données

Qu'il s'agisse d'une réponse à requête ou à WebHook, la structuration des données est identique et s'apparente à une transmission de données de type « Réponse ».
Il n'y a cependant pas de numéro de suivi, le protocole HTTP(S) permettant d'abstraire cette logique de par son mécanisme de requête-réponse contenu dans un même processus.
Il n'y a également pas d'identifiant unique d'élément destinataire/expéditeur, celui-ci étant implicitement le même que celui présent dans la requête ou le WebHook.
Les différents paramètres et valeurs sont définis dans un objet JSON :

- « Code » :
  - Obligatoire, non-nul et de type nombre entier.
  - Définit le code de retour de la réponse :
    - 0x00 : Retour valide.
    - 0x01 : Retour non valide.
    - 0xA0 : Destination introuvable.
    - 0xA1 : Temps maximum écoulé (timeout).
    - 0xA2 : Requête identique en cours.
- « Payload » :
  - Obligatoire, non-nul et de type spécifique selon le format précisé.
  - Contient les données utiles structurées au format précisé.
  - Doit contenir n'importe quel objet JSON si « Format » vaut « JSON ».
  - Doit contenir une chaîne de caractères si « Format » vaut « ASCII » ou « UTF8 ».
  - Doit contenir un tableau de valeurs de bytes si « Format » vaut « BINARY ».
- « Format » :
  - Obligatoire, non-nul et de type chaîne de caractères.
  - Indique la manière dont les données « Payload » sont à traiter.
  - Peut prendre les valeurs « JSON », « ASCII », « UTF8 » ou « BINARY ».

WebHook dédié à la télémétrie dans IoTSocket sur HTTP(S)

URI et méthode HTTP

L'URI est direct et ne contient jamais aucun paramètre « query string ».
La méthode HTTP utilisée pour les WebHooks dédiés à la télémétrie est « POST ».

Formatage et encodage du contenu

Le corps d'un WebHook dédié à la télémétrie est au format JSON, encodé en UTF-8.
Le type de contenu « Content-Type » est alors fixé à « application/json ».

Structuration des données

La structuration des données d'un WebHook dédié à la télémétrie s'apparente à une transmission de bloc de données utiles, formaté et à longueur variable.
Les différents paramètres et valeurs sont définis dans un objet JSON :

- « UID » :
  - Toujours présent et de type chaîne de caractères.
  - Contient l'identifiant unique de l'élément « Objet » ayant remonté les données.
  - Dont l'encodage UTF-8 ne dépasse jamais une longueur de 16 bytes.
- « Payload » :
  - Obligatoire, non-nul et de type spécifique selon le format précisé.
  - Contient les données utiles structurées au format précisé.
  - Doit contenir n'importe quel objet JSON si « Format » vaut « JSON ».
  - Doit contenir une chaîne de caractères si « Format » vaut « ASCII » ou « UTF8 ».
  - Doit contenir un tableau de valeurs de bytes si « Format » vaut « BINARY ».
- « Format » :
  - Obligatoire, non-nul et de type chaîne de caractères.
  - Indique la manière dont les données « Payload » sont à traiter.
  - Peut prendre les valeurs « JSON », « ASCII », « UTF8 » ou « BINARY ».

Réponse HTTP

Dans le contexte d'un traitement correct, le statut de réponse HTTP doit être 200 (OK).
Le contenu de la réponse sera ignoré et doit donc être vide.

SAS Hyper Connect Concept (HC²)
57 rue de la Division Leclerc, 78830 Bonnelles
CAPITAL 80 000 € - SIRET 82338914300012 - RCS VERSAILLES