Langage de Commande V5.0

Communication IP

L’objectif de ce document est de spécifier la communication sur support IP entre une station d’acquisition de surveillance de la qualité de l’air et un poste central.

Ce document décrit l’encapsulation des commandes et des fichiers du langage de commande dans le protocole http.

[I. Introduction 3](#_Toc441241767)

[A. Technologies Utilisées 3](#_Toc441241768)

[B. Encapsulation du LC 3](#_Toc441241769)

[1. Principe 3](#_Toc441241770)

[2. Fichiers échangés 3](#_Toc441241771)

[II. Établissement et gestion de la connexion TCP/IP 4](#_Toc441241772)

[A. Mode « classique » 4](#_Toc441241773)

[B. Mode « push » 4](#_Toc441241774)

[1. Contexte 4](#_Toc441241775)

[2. Fonctionnement 5](#_Toc441241776)

[III. Description des messages http 6](#_Toc441241777)

[A. Généralités 6](#_Toc441241778)

[1. Structure des requêtes 6](#_Toc441241779)

[2. Structure d’une réponse 7](#_Toc441241780)

[3. Syntaxe d’entêtes 7](#_Toc441241781)

[B. Initialisation du mode push 8](#_Toc441241788)

[1. Requête de la station 8](#_Toc441241789)

[2. Réponse du poste central 8](#_Toc441241790)

[C. Envoi d’une commande 9](#_Toc441241791)

[1. Ligne de commande 9](#_Toc441241792)

[2. Les entêtes 9](#_Toc441241793)

[3. Corps de la requête 9](#_Toc441241794)

[D. Réponse à une commande 12](#_Toc441241795)

[1. Ligne de statut 12](#_Toc441241796)

[2. Les entêtes 12](#_Toc441241797)

[3. Corps de la réponse 13](#_Toc441241798)

[E. Autres principes ? 14](#_Toc441241799)

# Introduction

## Technologies Utilisées

Les technologies choisies pour réaliser les échanges sous IP sont :

* Le protocole de communication HTTP (Hyper Text Transfer Protocol) en version HTTP/1.1 (RFC 2616),
* L’authentification intégrée à chaque requête (RFC 2617),
* Le type de requête : « POST de formulaire multi-part » (RFC 1867, RFC 2854, RFC 2388, HTML 4.01-cf. w3c).
* Utilisation de l’algorithme de compression référencé dans la RFC 1952 « GZIP file format specification version 4.3 » pour l’envoi de fichiers compressés.

## Encapsulation du LC

### Principe

En dehors de la procédure d’initialisation de la connexion, tous les échanges sont à l’initiative du poste central. Ils servent à transmettre à la station d’acquisition les commandes du langage de commande.

Un échange http encapsulant le langage de commande prend la forme suivante :

1. *Poste central :*

Envoi à la station d’une requête http contenant une commande du langage de commande ainsi qu’un éventuel fichier complémentaire.

1. *Station :*

Envoi au poste central de la réponse http contenant la réponse à la commande reçue.

Remarques :

* L’authentification est réalisée dans chaque requête http,
* Chaque requête http correspond à l’envoi d’une commande et une seule du LC.
* La commande est transmise directement dans le corps de la requête http complété ou pas par un fichier complémentaire.
* Une réponse valide de la station à la commande reçue contient obligatoirement un fichier du LC (fichier acquittement ou fichier demandé).

La réponse de la station peut ne pas contenir de fichier en cas d’erreur ou de refus d’authentification.

### Fichiers échangés

Les fichiers échangés correspondent aux fichiers du langage de commande. Le principe de nommage et les extensions doivent être respectés.

Les présentes spécifications distinguent les types de fichiers suivants :

* Fichier texte au format ASCII,
* Fichier binaire.

Ces fichiers peuvent être compressés au format GZIP (l’extension .GZ est ajoutée à l’extension de base du fichier),

Dans tous les cas, les fichiers reçus par l’un des dispositifs (client ou serveur) doivent être lu en mode binaire. Notamment pour éviter les erreurs de décompression et les mauvaises interprétations des caractères de fin de ligne.

# Établissement et gestion de la connexion TCP/IP

L’initialisation de la connexion doit être possible selon deux modes :

* Le mode « classique » : l’initialisation de la connexion est réalisée par le poste central.
* Le mode « push » : l’initialisation de la connexion est réalisée par la station d’acquisition. Dans ce cas de figure la connexion est maintenue ouverte par mécanisme de « keep-alive ».

## Mode « classique »

En mode « classique », la connexion est établie par le poste central. Les échanges se déroulent de la manière suivante :

Étape 1 : Ouverture de la connexion par le poste central.

Étape 2 : Série d’échanges http pour transmettre les commandes du langage de commande.

Étape 3 : Fermeture de la connexion par le poste central.

## Mode « push »

### Contexte

Le « mode push » est complémentaire du mode « classique » déjà mis en œuvre sur les stations.

Ce mode est activable ou désactivable dans la configuration de la station.

Il vise à répondre à l’ensemble des problématiques liées aux contraintes d’adressage IP et de traversée de pare-feu rencontrées sur le fonctionnement IP classique.

En effet, le fonctionnement IP classique impose :

* Que la station dispose d’une adresse IP publique, ou bénéficie d’un mécanisme de transfert de port (« port forwarding ») d’un routeur disposant lui-même d’une IP publique,
* Que la station dispose d’une adresse IP fixe,
* Qu’aucun dispositif, de type pare-feu ou autre, ne fasse obstacle à la communication TCP provenant du poste central à destination de la station.

En pratique, ces conditions sont difficiles à réunir sur certains réseaux, notamment les réseaux de téléphonie mobile (2G/3G/4G).

Des solutions de contournements ont déjà été mises en place par certains, mais présentes des contraintes :

* « Mire Iseo », permettant la gestion des IP dynamiques, mais nécessitant tout de même une IP publique
* Mise en place de tunnels VPN (configuration relativement complexe, et nécessitant l’utilisation de routeurs avec des capacités avancées)

Les connexions http étant réalisées depuis l’équipement terminal (station), vers le serveur (poste central), on se trouve du point de vue des équipements réseau dans un fonctionnement similaire à celui de la navigation Web, scénario supporté par l’ensemble des solutions d’accès à Internet du marché.

### Fonctionnement

Dès que le mode « push » est activé, la station initie une communication avec le poste central. Cette connexion est ensuite utilisée par le poste central dès qu’il souhaite transmettre des commandes à la station. Un mécanisme de Keep-Alive, géré par la station, est en charge de maintenir la connexion ouverte et de la rétablir en cas de coupure.

#### Établissement de la connexion par la station

La station établie une connexion http à destination du poste central, avec demande de la connexion selon le protocole Reverse http (Voir draft du protocole Reverse HTTP : <https://tools.ietf.org/html/draft-lentczner-rhttp-00>).

L’établissement de la connexion par la station se déroule de la manière suivante :

1. *Station :*

Envoi d’une requête http au poste central pour :

* s’authentifier auprès du poste central,
* demander l’établissement du dialogue selon le protocole Reverse http.
1. *Poste central :*

Réponse http à la station :

* pour accepter le mode de connexion Reverse http si l’authentification de la station est correcte,
* pour refuser si l’authentification de la station est invalide.

Après cet échange, le poste central peut initier, immédiatement ou pas, les envois de commandes du langage de commande.

#### Maintien de la connexion

La connexion TCP établie par la station d’acquisition doit être ouverte avec l’option « SO\_KEEPALIVE ».

Les paramètres suivants doivent être gérés et configurables dans la station d’acquisition :

* tcp\_keepalive\_time : temps en secondes avant le premier envoi du premier probe keepalive (Fixé à 900 par défaut)
* tcp\_keepalive\_intvl : intervalle de temps en secondes entre les envois de probe keepalive (Fixé à 180 par défaut)

Si la pile TCP/IP de la station détecte une fermeture de la connexion TCP (ce qui se produit en cas d’absence de réponse à un certain nombre de probes), la station doit initier une nouvelle connexion.

#### Gestion des connexions par le poste central

Le poste central doit tenir à jour une liste des connexions actives ouvertes par les stations en mode push.

Règles de gestion :

* Lorsque l’envoi d’une commande à destination d’une station est demandé, le poste central vérifie s’il dispose d’une connexion Push vers la station en question. Si c’est le cas il réutilise cette connexion.
* Lorsque la connexion TCP avec une station est fermée, celle-ci est supprimée de la liste des connexions actives.
* Si une station établit une nouvelle connexion alors que la précédente est encore considérée comme active, seule la connexion la plus récente est conservée.

# Description des messages http

## Généralités

Toutes les lignes du protocole HTTP doiventêtre terminées par CR/LF (les caractères ASCII 13 et 10 respectivement).

### Structure des requêtes

Une requête HTTP correspond à un ensemble de lignes envoyées au serveur par le client. Elle comprend:

* Une ligne de commande (obligatoire) comprenant trois éléments séparés par un espace :
* la méthode,
* l'url,
* la version du protocole utilisé par le client (HTTP/1.1).
* Plusieurs lignes d'en-têtes. Ces lignes sont de la forme « En-tête: valeur ».
* Une ligne vide (obligatoire).
* Un corps de message. Le coprs de message peut être constitué d’un fichier, de données d'une requête de type POST, etc. Il peut contenir plusieurs lignes, ou même des données binaires.

Une requête HTTP a donc la syntaxe suivante :

METHODE URL VERSION<crlf>

1ère EN-TETE: Valeur<crlf>

...

Dernier EN-TETE: Valeur<crlf>

Ligne vide<crlf>

CORPS DE LA REQUETE (0 à plusieurs lignes)

*Ligne de commande :*

La méthode utilisée pour les requêtes HTTP sera toujours la méthode « POST » qui transmet les données dans le corps de la requête.

Pour des raisons d’intercompatibilité, la valeur de l’URL utilisée, quelque soit le sens des échanges sera fixée à : « /script/LCV5.0/trans\_http ».

Ainsi la ligne de commande d’une requête http sera toujours :

POST /script/LCV5.0/trans\_http HTTP/1.1<crlf>

### Structure d’une réponse

Une réponse HTTP correspond à un ensemble de lignes envoyées au client par le serveur. Elle comprend :

* une ligne de statut (obligatoire) comprenant trois éléments séparés par un espace :
* la version du protocole utilisé,
* le code de statut (état du traitement de la requête),
* la signification du code (texte explicatif).
* Zéro ou plusieurs lignes d'entêtes. Ces lignes sont de la forme ‘Entête: valeur’.
* Une ligne vide (obligatoire).
* Le corps de la réponse: il contient normalement le document demandé ou un fichier d’acquittement.

Une réponse HTTP a donc la syntaxe suivante :

VERSION-HTTP CODE EXPLICATION<crlf>

1ère EN-TETE: Valeur<crlf>

…

Dernière EN-TETE: Valeur<crlf>

<crlf>

*CORPS DE LA REPONSE* (0 à plusieurs lignes)

### Syntaxe d’entêtes

Cette partie présente la syntaxe de certains entêtes communs aux différents types d’échanges décrits ci-après.

Les autres entêtes utilisés seront présentés au cas par cas directement dans la description des échanges http.

#### Host

Cet entête a pour rôle d’identifier la machine qui reçoit la requête par son URI (Uniform Resource Identifiers – voir RFC 2396).

Le format est le suivant :

Host: *adresse\_server*:*port\_http*<crlf>

Remarques :

* Le port http doit être spécifié s’il est différent de 80.
* L’adresse peut être spécifiée sous format IPv4 ou nom de domaine.

Ex. :

Host: 192.168.79.200:3005

#### Authorization

L’entête «Authorization» est utilisé par le client pour s’authentifier auprès du serveur. Le format est le suivant :

Authorization: *login:passwordbase64*<crlf>

La méthode d’authentification choisie est de type BASIC et est encodée en base 64. Elle est décrite dans la RFC 2617. **base64** est un encodage et non un cryptage.

Quelque soit le sens de communication, le login correspond au numéro de la station participant à l’échange.

Le mot de passe, correspond à un utilisateur définit pour l’un des profils disponibles pour accéder à la station ou au poste central.

Exemple : 09054:argopol

La valeur transmise est encodée en base 64. Pour l’exemple « 09054:argopol», l’entête «Authorization » sera :

Authorization: MDkwNTQ6YXJnb3BvbA==<crlf>

#### Date

L'entête « Date» représente la date à laquelle le message a été transmis. Le format de date de l’exemple suivant est à respecter :

Date: Mon, 13 Apr 2015 13:35:21 GMT<crlf>

## Initialisation du mode push

### Requête de la station

Pour l’initialisation de mode Push, la station d’acquisition envoie au poste central une requête http afin de s’authentifier et de demander la poursuite des échanges selon le protocole http reverse.

Cette requête doit prendre la forme suivante :

POST /script/LCV5.0/trans\_http HTTP/1.1<crlf>

Host: *adr\_poste\_central*:*port\_http*<crlf>

Upgrade: PTTH/1.0<crlf>

Connection: Upgrade<crlf>

Authorization: *id\_station:mdp\_poste\_centralbase64*<crlf>

### Réponse du poste central

Si le poste central accepte la requête station il répond par :

HTTP/1.1 101 Switching Protocols<crlf>

Upgrade: PTTH/1.0<crlf>

Connection: Upgrade<crlf>

Si le poste central refuse la demande pour des raisons d’authentification, la première ligne de la réponse sera la suivante :

HTTP/1.1 401 Unauthorized<crlf>

## Envoi d’une commande

L’envoi d’une commande du langage de commande est réalisé par l’envoi d’une requête http par le poste central. Cette requête contient la commande, ses arguments ainsi qu’un éventuel fichier complémentaire.

### Ligne de commande

La ligne de commande de la requête sera toujours :

POST /script/LCV5.0/trans\_http HTTP/1.1

### Les entêtes

Les quatre entêtes suivants doivent être utilisés, dans cet ordre :

Host: *adresse\_station*:*port\_http*<crlf>

Authorization: *id\_station:mdp\_stationbase64*<crlf>

Content-Type: multipart/form-data; boundary=*boundary*<crlf>

Content-Length: *taille\_corps\_message*<crlf>

Le *boundary* de l’entête « content-type » est une chaine de caractères aléatoires utilisée comme délimiteur/séparateur pour les différents éléments du corps de la requête (voir ci-dessous). Le boundary est créé par le client.

Aucune occurrence du boundary ne doit figurer dans le corps du message

Exemple de boundary :

Content-Type: multipart/form-data; boundary=-------166801948522407

### Corps de la requête

Le corps de la requête contient la commande du langage de commande à transmettre, ses éventuels arguments et l’éventuel fichier complémentaire.

#### Structure du corps de la requête

La structure du corps de requête est constituée de plusieurs « blocs » servant chacun à transmettre un élément.

Les différents « blocs » sont séparés par le boundary défini dans l’entête « Content-Type » de la requête. Pour débuter chaque bloc du corps de message, le boundary est précédé par deux tirets -- (code ASCII 45). La fin du corps de message est signifiée par le boundary précédé et suivi par deux tirets --.

Structure du corps de la requête :

--boundary<crlf>

*1er bloc*

--boundary<crlf>

*2ème bloc*

--boundary<crlf>

…

--boundary<crlf>

*dernier bloc*

--boundary--<crlf>

#### Transmission des éléments

Les éléments transmis dans les blocs du corps de la requête sont transmis en utilisant l’entête « Content-Disposition: form-data » (voir RFC 2183) selon le modèle suivant :

--boundary<crlf>

Content-disposition: form-data; name="*nom\_element*"<crlf>

<crlf>

*valeur de l’élément*<crlf>

Les éléments transmis sont les suivants :

* Le nom de la commande du LC,
* Éventuellement le 1er argument de la commande,
* Éventuellement le 2ème argument de la commande,
* …
* Éventuellement le fichier complémentaire.

##### Nom de la commande

Pour la transmission de la commande, le paramètre « *nom\_élément* » de l’entête « Content-Disposition » peut prendre les valeurs suivantes :

* CDE
* ou CDZ si les fichiers transmis et attendus sont compressés.

La valeur de l’élément correspond au libellé de la commande (sans ses arguments).

Ex :

--gc0p4Jq0M2Yt08jU534c0p<crlf>

Content-Disposition: form-data; name="**CDE**"<crlf>

<crlf>

**CONFIG\_STAT**<crlf>

##### Les arguments

Pour la transmission des éventuels arguments de la commande, le paramètre « *nom\_élément* » de l’entête « Content-Disposition » prend la valeur : ARG*n,* avec *n* le numéro de l’argument.

La valeur correspond à la valeur de l’argument du LC.

Ex :

--gc0p4Jq0M2Yt08jU534c0p<crlf>

Content-Disposition: form-data; name="**ARG1**"<crlf>

<crlf>

**T**<crlf>

S’il y a plusieurs arguments pour une commande, ils sont transmis séparément : un par bloc.

Remarque : Dans le cas où un fichier complémentaire est transmis avec la commande, le nom du fichier n’est pas considéré comme un argument de la commande. Le fichier et son nom sont transmis selon le principe décrit au paragraphe suivant.

##### Le fichier complémentaire

Dans le bloc contenant l’éventuel fichier complémentaire, deux ou trois entêtes sont utilisés :

Entête « Content-Disposition » :

Le paramètre « *nom\_élément* » de l’entête « Content-Disposition » prend la valeur : file.

L’entête est complétée par le paramètre « filename » qui indique le nom du fichier transmis.

Ex :

Content-Disposition: form-data; name="file"; filename="nom\_fichier"

Nom du fichier :

* Le nom du fichier est indiqué entre guillemets.
* Pour les fichiers compressés, l’extension du fichier doit être complétée par « .GZ »

Entête « Content-Type » :

L’entête « Content-Type » complète le bloc pour spécifier la nature du fichier transmis. Les valeurs possibles pour « Content-Type » sont :

* « text/plain » pour un fichier texte ASCII,
* « application/octet-stream » pour un fichier binaire.

Si le fichier transmis est compressé on ajoute l’entête « Content-Encoding: gzip».

Ex : envoi d’un fichier de configuration non compressé

--gc0p4Jq0M2Yt08jU534c0p<crlf>

Content-Disposition: form-data; name="**file**"; filename="**40005060.CFG**"<crlf>

Content-Type: text/plain<crlf>

<crlf>

*Contenu du fichier*<crlf>

--gc0p4Jq0M2Yt08jU534c0p--<crlf>

Ex : envoi d’un fichier de configuration compressé

--gc0p4Jq0M2Yt08jU534c0p<crlf>

Content-Disposition: form-data; name="**file**"; filename="**40005060.CFG.GZ**"<crlf>

Content-Type: text/plain<crlf>

Content-Encoding: gzip<crlf>

<crlf>

*Contenu du fichier compressé*<crlf>

--gc0p4Jq0M2Yt08jU534c0p--<crlf>

## Réponse à une commande

La réponse de la station

### Ligne de statut

La ligne de statut, 1ère ligne de la réponse, sera toujours dans le format suivant :

HTTP/1.1 *code* *signification\_code*<crlf>

Si la station accepte la requête du poste central la ligne de statut sera toujours :

HTTP/1.1 200 OK<crlf>

Si la station refuse l’authentification, la ligne de statut sera toujours :

HTTP/1.1 401 Unauthorized<crlf>

Si la station ne reconnaît pas la version du langage de commande utilisé, la ligne de statut sera :

HTTP/1.1 404 Version LC inconnue<crlf>

D’autres réponses peuvent être nécessaires, elles ne seront pas spécifiées dans ce document et laissées à l’appréciation des constructeurs.

Remarque : le code statut renseigne sur le bon déroulement de l’échange http et de l’authentification. Les erreurs liées aux traitements de la commande sont décrites dans un ficher d’acquittement du LC.

### Les entêtes

Les entêtes suivants doivent être utilisés, dans cet ordre :

Date: *date*<crlf>

Content-Length: *taille\_corps\_message*<crlf>

Content-Type: *valeur*; filename="*nom\_fichier*"<crlf>

Eventuellement, si le fichier transmis est compressé, l’entête « Content-Encoding: gzip» est ajouté.

L’entête « Content-Type » indique le type du fcihier transmis et son nom.

Le champ « *valeur* » indique la nature du fichier transmis :

* « text/plain » pour un fichier texte,
* « application/octet-stream » pour un fichier binaire.

Nom du fichier :

* Le nom du fichier est indiqué entre guillemets.
* Pour les fichiers compressés, l’extension du fichier doit être complétée par « .GZ »

Exemple pour la transmission d’un fichier de configuration compressé :

HTTP/1.1 200 OK<crlf>

Date: Mon, 18 Jan 2016 13:35:21 GMT<crlf>

Content-Length: 1060<crlf>

Content-Type: text/plain; filename="99007018.CFG.GZ"<crlf>

Content-Encoding: gzip<crlf>

<crlf>

*Corps de message*

### Corps de la réponse

Le corps de la réponse contient directement le contenu du fichier transmis par la station en réponse à la commande du LC.

Questions :

Entêtes de requêtes :

Autres entêtes acceptées ? :

* Xr utilise 3 entêtes supplémentaires dans ces requêtes :
* Accept: \*/\*
* Expect: 100-continue qui signifie que Xr attend un retour de la station avant d’envoyer le reste de sa requête.
* Accept-Encoding: gzip dans le cas où le fichier attendu en réponse doit être compressé.
* Dans le cas d’un échec d’authentification, il y a d’autres entêtes pour la réponse.

Ordre prédéfini ?

* Polair accepte les entêtes dans n’importe quel ordre.

Entêtes réponses :

Autres entêtes acceptées ?

* Polair accepte qu’il y ait d’autres entêtes dans les réponses, il les ignore.
* Station FDE utilise en plus
	+ l’entête « content-language »
* Stations ISEO utilisent en plus : « Allow: GET, HEAD, POST »

Définir un ordre ? un découpage ?

* Les stations FDE transmettent tous les entêtes de leur réponse dans le premier paquet IP transmis.
* Les stations Iseo découpent systématiquement leur réponse après l’en-tête « date ».
* Polair a besoin que tous les entêtes soient nécessairement dans le 1er paquet transmis. Les paquets suivants sont considérés comme contenant des morceaux du fichier.

## Autres principes ?

Règles supplémentaires sur la séparation des envois, l’ordre des entêtes, etc ?

* Découpage des envois en plusieurs paquets : il n’y avait aucune règle de précisée à ce sujet :
* Pour la réponse, Polair a besoin que tous les entêtes soient contenus dans le premier paquet reçu. Le début du fichier peut être dans le premier paquet ou dans le suivant ça ce n’est pas gênant.
* Dans l’envoi de ses requêtes aux stations ISEO, Xr utilise un entête « expect continue ». Cela implique que la station indique au client qu’il peut continuer avant que celui-ci ne poursuive l’envoi de sa requête.

Annexes

Exemples d’échanges