Data Logging Service (DLS)
Version 1.2

Florian Pose, fp@igh-essen.com

Ingenieurgemeinschaft IgH

"DLS" est un système d'archivage capable de collecter des données hautes fréquences pendant une longue période et de les stocker sous forme très compressée. Son objectif est d'offrir à l'utilisateur un accès performant et sans limite aux données enregistrées, aussi bien la vue d'ensemble annuelle qu'une infime fluctuation pendant une fraction de seconde.

Essen, 6 janvier 2020

Révision –revision–

Traducteur : Sébastien BLANCHET

Table des matières

1. Généralités

1.1. Principe de l'acquisition de données

"Data Logging Service" (ci-après *DLS*) est un système d'enregistrement de données qui est non seulement capable de collecter, compresser et stocker tout type de mesures pendant une longue période, mais aussi de les afficher rapidement chaque fois que nécessaire.

Le prérequis pour un processus d'acquisition de données est une *source de données*, qui fournit les données mesurées. Dans ce cas, il s'agit d'un serveur réseau qui fournit les données mesurées via *rt_lib*, une bibliothèque dévelopée par *IgH*. La communication avec la source de données est décrite dans sous-section 2.2.3.

Toutes les données à livrer sont organisées en *canaux*. Un canal est une abstration d'une quantité physique mesurable fournie par la source de données. Les propriétés du canal sont l'unité, la fréquence maximale d'échantillonnage et le type de donnée.

DLS peut se connecter aux sources de données puis interroger les informations via les canaux fournis. De la même manière, il est alors capable de demander et de recevoir les données pour des canaux spécifiques.

1.2. Tâches d'acquisition

DLS enregistre les données par l'intermédiaire de *tâches d'acquisition*. Elles comprennent les spécifications générales pour l'acquisition de données dont la liste des canaux à inclure et leurs spécifications. Une tâche d'aquisition est toujours liée à une source particulière de données. Un nombre quelconque de tâches d'aquisition existent simultanément.

Une tâche d'acquisition peut recevoir simultanément les données de différents canaux. Cela nécessite que chaque canal soit étiqueté avec une *spécification de canal* qui récaptitule les conditions de rappatriement et de stockage des données. Ceci inclut la fréquence d'échantillonnage, la taille des blocs, les méta-données à récupérer (voir sous-section 2.2.2), le ratio de méta-réduction et la méthode de compression.

1.3. Stockage des données

Les données capturées sont stockées dans le répertoire de données de DLS conjointement avec les spécifications de l'acquisition, le temps et les informations du canal puis triées par tâche d'acquisition. Par défaut, il s'agit du répertoire courant. Si un autre emplacement de stockage est choisi, cette information peut être communiquée aux programmes de DLS soit via la ligne de commande (option `-d`), soit via la variable d'environnement $DLS_DIR. L'ordre de prévalence est toujours le suivant : ligne de commande – variable d'environnement – répertoire courant.

Plusieurs répertoires de données DLS peuvent coexister simultanément à condition qu'ils soient gérés par différentes instances du daemon DLS (voir section 2.1).

Une description détaillée de la structure du répertoire de données DLS et des données qu'il contient, se trouve dans chapitre 3.

1.4. Outils

DLS Manager Les tâches d'acquisition peuvent être éditées via l'interface graphique de *DLS Manager* (voir chapitre 4). Le programme permet de créer de nouvelles tâches et de modifier celles qui existent déjà, même pendant les aquisitions. L'utilisateur peut aussi visualiser la liste des tâches en cours d'exécution.

DLS View Un outil graphique *DLS View* permet de visualiser les données (voir chapitre 5). L'utilisateur peut afficher n'importe quel canal au-dessus d'une échelle de temps commune. Il est alors possible de naviguer dans la fenêtre temporelle et de lire les valeurs individuelles.

dls L'outil en ligne de commande *dls* permet de visualiser et d'exporter les données déjà enregistrées. Voir section D.6 pour la liste des sous-commandes.

Script d'initialisation Un script d'initialisation est fourni pour démarrer et stopper dlsd et les services associés. Il reconnaît les paramètres `start`, `stop`, `restart` et `status`. La configuration du service est effectuée via le fichier sysconfig qui contient toutes les variables nécessaires. Il est recommandé d'exporter les variables incluses via le script profile également fourni, pour les rendre ainsi accessibles à tous les utilisateurs.

dls_status Le script *dls_status* sert à la supervision générale des acquisitions. Cet un outil en ligne de commande peut afficher l'état de santé fondamental de l'acquisition pour un répertoire de données DLS spécifique. Il indique si le processus parent DLS est en cours d'exécution, quelles tâches d'acquisition sont disponibles et si les processus correspondants sont en cours de fonctionnement.

2. Le daemon DLS (dlsd)

Le *daemon DLS* (en abrégé : *dlsd*) sert à l'ensemble de l'acquisition et du stockage de données. C'est un processus qui tourne en tâche de fond (sans être connecté à la console) et qui lié à un répertoire spécifique de données DLS. Il doit toujours être en fonctionnement lors d'une acquisistion.

Figure 2.1 représente l'architecture du système.

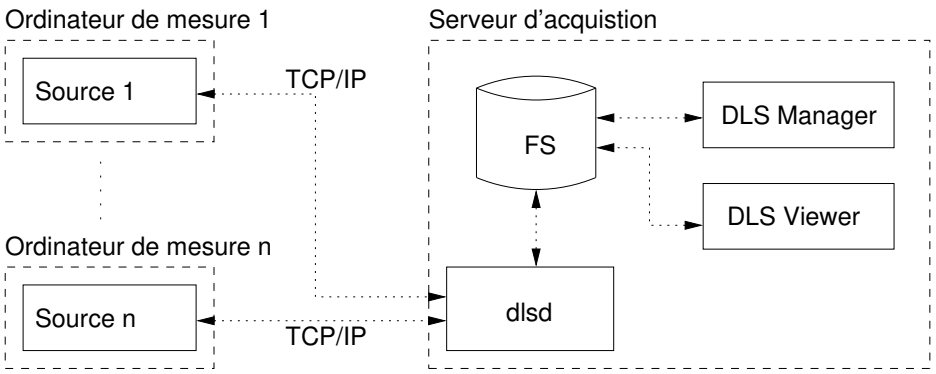

FIGURE 2.1. – Architecture

2.1. Le processus parent dlsd

La commande `dls start` démarre un processus dlsd de surveillance des tâches d'acquisitions. Il est appelé le *processus parent dlsd*. Il contient une copie des spécifications des tâches existantes dans le répertoire de données DLS, ainsi que les informations des processus d'acquisition associés.

Chaque processus parent doit fonctionner dans un répertoire de données DLS différent. Ceci est garanti par les mécanismes de protection mis en oeuvre (*fichiers PID*, voir Appendice C).

2.1.1. Comportement du processus parent dlsd

Au démarrage, le processus parent lit toutes les spécifications des tâches d'acquisition dans le répertoire de données DLS et crée un processus enfant pour chaque tâche

d'acquisition active (*processus d'acquisition dlsd*, voir section 2.2), qui servira ensuite à l'acquisition de données de la tâche concernée. Ensuite, il effectue périodiquement les actions suivantes :

— Vérifier si des signaux Unix ont été reçus entre temps. Ceci peut arriver, par exemple, lorsque dlsd doit s'arrêter ou lorsque l'un des processus d'acquisition a été interrompu. (voir sous-section 2.1.3).
— Vérifier le répertoire de file d'attente pour de nouvelles entrées. Si des fichiers se trouvent dans ce répertoire, ils sont évalués comme des informations de mise en file d'attente et traités comme décrit dans sous-section 2.1.2.
— Vérifier les processus d'acquisition. Le processus parent est responsable que chaque tâche d'acquisition possède son processus acquisition en cours d'exécution.

2.1.2. File d'attente

Afin de garantir une modification fluide des tâches d'acquisition pendant la capture des données, une file d'attente est utilisée.

Sans elle, l'organisation des tâche d'acquisition en fichiers pourrait conduire à la lecture par dlsd de fichiers encore en cours d'édition par l'utilisateur. Ce qui pourrait alors générer des erreurs de traitement.

C'est pourquoi, le répertoire de données DLS contient un sous-répetoire `spool`. Le processus parent dlsd vide ce répertoire au démarrage puis lit une seule fois toutes les spécifications de tâche. Après cela, il n'accède en lecture aux spécifications de tâche qu'en cas de demande explicite par une commande qui attend dans la file.

Pour dlsd, une commande valide en attente est un fichier de nom quelconque qui se trouve dans le répertoire de file d'attente et qui contient uniquement un identifiant de tâche (ID) sous la forme d'un entier positif codé en ASCII. Cet ID permet à dlsd de décider ce qu'il doit faire :

— S'il ne connait pas encore de tâche avec cet ID, il suppose que le tâche a été nouvellement créé. Il importe donc le processus d'acquisition et le démarre si nécessaire.
— S'il connait déjà une tâche avec cet ID **et** que le fichier avec les spécification du tâche existe (*job.xml* voir section 3.2), les spécifications vont être relues et le processus d'acquisition sera, le cas échéant, démarré, stoppé ou notifié.
— S'il connait une tâche avec cet ID, mais que **le fichier de spécification n'existe pas**, dlsd supposera que le tâche a été supprimé. Il terminera le processus d'acquisition en cours et retirera la spécification de sa liste.

En général, le fichier de file d'attente est supprimé pour confirmer que la nouvelle information a bien été reçue. Si une erreur se produit durant la procédure, le fichier sera laissé dans le répertoire de file d'attente.

2.1.3. Traitement des signaux

Les signaux UNIX ci-dessous sont traités par le processus parent dlsd :

SIGCHLD Un processus enfant est terminé. Ceci peut avoir différentes raisons :
- Le processus a été explicitement terminé et sa valeur de retour est 0 (pas d'erreur). Dans ce cas, le processus enfant sera redémarré lors de la prochaine vérification, car son arrêt n'est pas conforme aux spécifications et l'acquisition des données doit reprendre aussi vite que possible.
- Le processus a détecté une erreur interne et s'est arrêté de lui-même avec la valeur de retour -1. Cette information sera prise en compte par le processus parent et le processus enfant ne sera pas redémarré. Un message sera envoyé à *syslogd*, mais l'utilisateur ne recevra pas d'avertissement explicite à propos de l'arrêt des acquisitions.
- Le processus a subit une incohérence temporelle et s'est arrêté de lui-même avec la valeur de retour -2. Le processus enfant sera redémarré par le processus parent après l'écoulement d'un laps de temps défini.

SIGINT/SIGTERM dlsd doit s'arrêter. Le processus parent fait suivre le signal à tous les processus enfants, attend qu'ils aient enregistré leurs données et s'arrête alors de lui-même avec la valeur de retour 0 (pas d'erreur).

SIGSEGV et autres Ces signaux sont surveillés par sécurité et traités de la même façon par le processus parent dlsd et les processus d'acquisition. Si un tel état arrive, le processus laissera un fichier nommé *error_<PID>* dans le répertoire de données DLS avec les informations concernant le signal reçu, puis s'arrêtera aussitôt de lui-même avec la valeur de retour -3.

2.2. Le processus d'acquisition dlsd

Le processus d'acquisition qui a essaimé du processus parent dlsd, sert à la communication avec la source de données, la compression des données reçues et l'enregistrement des données compressées sur le disque dur. Il est associé à une tâche d'acquisition spécifique qui lui a été assignée par le processus parent dlsd au démarrage. Cette tâche d'acquisition est identifiée de manière unique par le répertoire de données DLS et l'identifiant (ID) de tâche d'acquisition correspondant. Toutes les données qui se réfèrent à cette tâche sont enregistrées dans le sous-répertoire *job<ID>* (voir chapitre 3).

2.2.1. Comportement du processus d'acquisition

Le processus d'acquisition dlsd commence par importer les spécifications des tâches d'acquisition depuis le fichier central des spécifications *job<ID>/job.xml* puis se connecte

via TCP/IP à la source de donnée (pour la description du protocole de communication, voir sous-section 2.2.3).

Pour chaque canal, les données capturées vont dans un tampon appellé *block buffer*. Lorsqu'il est plein, les données sont compressées par bloc et horodatées avec la date de début du bloc. L'avantage de cette approche est que la compression n'a pas besoin de processus de flux et que les données individuelles sont clairement identifiables par la suite. La taille du bloc (et donc celle de *block buffer)* peut être définie par l'utilisateur dans les spécifications des canaux.

Au sein de cette connection, une séquence chronologique et continue de blocs constitue un tronçon (*chunk*). Il contient les données capturées pour un canal individuel pendant un intervalle de temps complet et continu. Il combine les spécifications du canal sous-jacent, ses propriétés réelles et les données enregistrées (voir section 3.4).

2.2.2. Génération des méta-données

Dans le but de fournir une pré-visualisation rapide d'une grande quantité de données, le processus d'acquisition DLS stocke non-seulement les données ("génériques") *(generic)* qui ont été reçues depuis la source de données mais aussi des données aggrégées pendant un laps de temps et appelées "méta-données" *(meta data)*. Elles existent elles-même à plusieurs niveaux de réductions, appelés "méta niveaux" *(meta levels)*. Un processus de lecture peut alors - en fonction de la résolution choisie - decider quel méta-niveau il va utiliser pour charger les données plus rapidement.

D'un point de vue mathématique, cela signifie que la complexité de l'algorithme pour charger n valeurs appartenant à l'intervalle de temps Δt avec une fréquence d'échantillonnage f, c'est-à-dire $n = \Delta t \cdot f$, n'est plus $O(n)$, i. e. linéairement dépendant du nombre de valeurs dans l'intervalle, mais uniquement dépendant du nombre de points supports désirés dans la résolution courante. Ce nombre de points supports est indépendant du nombre de valeurs sous-jacentes. Par conséquent, d'un point de vue du temps d'exécution et de l'accès au stockage, l'algorithme est de l'ordre de $O(1)$.

Pour générer les méta-données (redondantes) un *méta-tampon (meta buffer)* est disponible en parallèle du tampon de bloc, dans le processus d'acquisition dlsd. Ce tampon a une capacité de u valeurs où u est est le rapport de *méta-réduction (meta reduction ratio)*, que l'utilisateur peut définir dans les spécifications du canal. Le rapport de méta-réduction s'applique à tous les niveaux. Lorsque le méta-tampon est plein, une "méta-valeur" de niveau 1 va être générée à partir des u valeurs génériques, et stockée là dans un bloc et un méta-tampon. Pour ces tampons, les mêmes règles s'appliquent aussi comme pour le niveau des données génériques, c'est-à-dire que les méta-niveaux sont générés en "cascade". Par conséquent, l'espace de stockage pour un nouveau niveau est réservé uniquement lorsque la première méta-valeur est disponible.

Il existe différents types de méta-données qui peuvent être générés simultanément. Jusqu'à présent, les types suivants sont supportés :

Valeur moyennes ("mean", bit de masque 0) Une méta-valeur de niveau n est la moyenne arithmétique des u valeurs de niveau $n-1$.

Minima ("min", bit de masque 1) Une méta-valeur de niveau n est le minimum des u valeurs de niveau $n-1$.

Maxima ("max", bit de masque 2) Une méta-valeur de niveau n est le maximum des u valeurs de niveau level $n-1$.

Le type de méta-valeurs à générer pendant l'acquisition peut être spécifié par l'utilisateur dans les spécifications du canal via le *méta-masque (meta mask)*. Il est calculé par l'opération OU binaire des bits du masque indiqué. (Exemples : "Valeurs moyennes + minima + maxima" correspond au méta-masque 7, "seulement les valeurs moyennes" correspond au méta-masque 1, alors que "minima + maxima" correspond au méta-masque 6).

De manière quasi systématique, lors de l'achèvement d'un tronçon, il y aura moins de u valeurs dans un niveau. Pour ces valeurs, **aucune** méta-valeur ne sera générée car l'affichage de ces données sera selon toute probabilité plus étroit qu'un pixel. Les valeurs restantes dans les méta-tampons seront par conséquent éliminées.

2.2.3. Communication avec la source de données

La communication avec la source de donnée est effectué via le protocole *rt_lib* (version ≥ 2.7) de *IgH* en XML. La connexion à la source de donnée est établie via TCP/IP (port 2345).

Identification de la source de donnée Une fois que la connexion a été établie, la balise <*connected*> est attendue avec l'attribut *name* qui doit contenir la valeur *MSR* (en allemand „Messen – Steuern – Regeln", ce qui signifie "Mesurer - Contrôler - Réguler"). L'attribut *version* permet de vérifier le numéro de version de la source de donnée et sa compatibilité avec la version courante de dlsd. En outre, la balise <*connected*> peut contenir l'attribut *arch* qui contient l'architecture ("endianness") de la source de données et donc le type de codage des mots binaires. Les valeurs possibles sont *big* (pour "big endian") ou *little* (pour "little endian"). Si l'attribut *arch* n'existe pas, l'architecture supposée de la source est alors "little endian" et un avertissement est émis.

Determination de la fréquence maximale d'échantillonnage Lorsque la balise <*connected*> qui a été envoyée par la source de donnée, a été reçue et vérifiée, le processus d'acquisition va interroger le paramètre MSR */Taskinfo/Abtastrate* (fréquence d'échantillonnage) et attendre la réponse. Cette valeur (la fréquence maximale d'échantillonnage de la source de donnée) sera nécessaire par la suite pour vérifier que les spécifications du canal sont plausibles et pour calculer le ratio de réduction des fréquences d'échantillonnage.

Lecture de tous les canaux Par la suite, la liste complète des canaux fournis par la source de données peut être obtenue par la commande <*rk*>. La réponse de la source de données doit commencer par la balise <*channels*> suivie des canaux individuels qui sont chacun décrits par la balise <*channel*>. La fin de la liste est marquée par la balise </*channels* >.

Exemple d'une réponse de la source de donnée à la commande <*rk* > :

```
<channels>
 <channel name="/Time" unit="s" alias="" index="0"
  typ="TDBL" bufsize="50000" HZ="10000" value="1112814601.3209"/>
 <channel name="/Taskinfo/Controller_Execution_Time" unit="us"
  alias="" index="6" typ="TUINT" bufsize="50000" HZ="10000" value="22"/>
 <channel name="/Taskinfo/Controller_Call_Time" unit="us" alias=""
  index="7" typ="TUINT" bufsize="50000" HZ="10000" value="99"/>
 <channel name="/Istwert/Kraft" unit="N" alias="" index="9"
   typ="TDBL" bufsize="50000" HZ="10000" value="-0.6745"/>
 <channel name="/Istwert/Druck" unit="bar" alias="" index="12"
  typ="TDBL" bufsize="50000" HZ="10000" value="0.1372"/>
</channels>
```

Les attributs suivants de la balise <*channel*> seront conservés pour un usage ultérieur :

name – Nom du canal (unique)

unit – Unité du canal (optionnel, enregistrée en tant que chaîne de caractères)

index – Position du canal dans la liste. Cette information servira pour identifier le répertoire de stockage du canal dans le répertoire de données DLS (voir chapitre 3).

typ – Type de donnée. Il doit correspondre à un type connu de données dans la table Appendice B, pour activer ultérieurement le traitement des données reçues.

bufsize – Taille du tampon circulaire au sein de la source de données. Cette information sera utilisé ultérieurement pour vérifier que la fréquence d'échantillonnage :

$$\text{BlockSize} \cdot \text{Reduction} \overset{!}{\leq} \frac{\text{BufferSize}}{2}$$

HZ - Spécifique au canal, fréquence maximale d'échantillonnage.

Tous ces détails à propos des canaux sont sauvegardés dans la mémoire de chaque processus d'acquisition et seront utilisés pour des vérifications de plausibilité lors de l'ajout ou de la modification des spécifications d'un canal.

Démarrage de l'acquisition de données Quand le processus d'acquisition reconnaît la liste des canaux, l'acquisition de données va commencer (à moins qu'il ne soit nécessaire d'attendre auparavant le paramètre de déclenchement). C'est fait via la commande <*xsad*>, qui est envoyée une fois pour chaque canal à collecter. Les attributs de la commande sont :

channels - Contient l'index du canal demandé dans la liste des canaux,

reduction - le facteur (entier) de réduction de la fréquence maximale d'échantillonnage du canal pour décrire la fréquence absolue d'échantillonnage.,

blocksize - le nombre de valeurs à envoyer dans un bloc (cette valeur est complètement indépendante de la taille de bloc dans les spécifications du canal), et

coding - l'encodage des données, actuellement défini en *Base64*.

Une commande typique de démarrage de l'acquisition de donnée ressemblera à ceci :

```
<xsad channels="7" reduction="100" blocksize="1000" coding="Base64"/>
```

Réception des données Le processus d'acquisition attend maintenant la balise <*data*> qui marque le début d'un bloc de balises de données de canaux. Cette balise doit obligatoirement contenir l'attribut *time* correspondant à l'horodatage de toutes les dernières valeurs de données, chacune dans les balises de données suivantes. Il s'agit des balises <*F*>, qui contienent les dernières valeurs mesurées pour un canal unique avec les attributs *c* (index du canal) et *d* (donnée mesurée et encodée). La dernière balise est obligatoirement suivie par la balise </*data*> qui remet le processus d'acquisition en état d'attente.

Modification des spécifications d'un canal Si les spécifications d'un canal sont modifiées pendant l'acquisition des données, le processus d'acquisition enverra une autre balise <*xsad*>, qui contiendra les nouvelles spécifications du canal et l'attribut *id* représentant sans équivoque l'identifiant de la commande. Si la source de donnée a accepté les spécifications du canal et si les prochaines données du canal concerné sont compatibles avec les **nouvelles** spécifications, la source de donnée enverra préalablement la balise <*ack*>, puis les attributs de la balise <*xsad*> relative à l'identifiant ID de la commande. Enfin le processus d'acquisition sera finalement modifié pour être conforme aux nouvelles spécifications du canal.

2.2.4. Limitation du volume des données (quota)

dlsd possède des mécanismes pour limiter l'espace de stockage requis pour les données acquises via les tâches d'acquisitions. Différents critères sont supportés pour tout dépassement de ces limites :

Quota de données Le répertoire de la tâche d'acquisition ne doit pas dépasser une certain taille dans le système de fichiers.

Quota de temps L'intervalle de temps des données acquises par une tâche d'acquisition ne doit pas dépasser une largeur définie.

Si l'utilisateur a activé un ou plusieurs quotas et que l'ensemble des données acquises dépasse un ou plusieurs critères, les tronçons les plus anciens des cas concernés seront supprimés, jusqu'à ce que les critères ne soient plus remplis. Cependant, le tronçon le plus récent de chaque canal n'est jamais supprimé, car ici une acquisition de données pourrait avoir lieu.

Le daemon DLS quota prend en charge la suppression. Ce dernier doit toujours fonctionner en parallèle de dlsd dès qu'un quota a été configuré. Le démarrage peut être réalisé manuellement avec :

```
$ dls_quota
```

Pour les paramètres de ligne de commande se référer à section D.7.

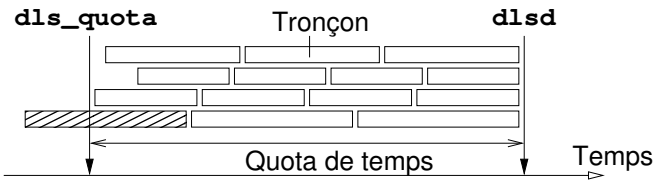

FIGURE 2.2. – Conformité avec les quotas de temps

Puisque le daemon DLS quota ne peut supprimer que des tronçons entiers, une suppression successive serait impossible, si les données acquises pour un canal consistaient seulement en un tronçon unique. C'est pourquoi, il faut garantir qu'il existe toujours plusieurs petits tronçons, même si le processus d'acquisition dlsd n'a pas été interrompu.

Par conséquent, le processus d'acquisition dls surveille les critères de quota pour lui-même. Lorsque les quotas sont activés, il s'assure que suffisament de tronçons individuels sont produits au sein de la zone critique (Figure 2.2), Dans ce but, il découpe chaque ensemble de quota en portions égales et commence indépendament un nouveau tronçon dès qu'un de ces critères plus fins est dépassé.

Comme l'achèvement d'un tronçon peut être très coûteux en temps, ce qui est incompatible avec les contraintes temps-réel du processus d'acquisition dlsd, un processus dédié est généré pour enregistrer les données acquises restantes. Ce "processus de nettoyage" va enregistrer maintenant toutes les données du tronçon courant, tandis que le processus d'acquisition s'en débarasse et s'occupe de l'acquisition des données du nouveau tronçon qui commence. Après l'achèvement du "vieux" tronçon, le processus de nettoyage s'arrête automatiquement.

2.2.5. Messages depuis la source de données

La source de donnée peut – en plus des données mesurées – envoyer des messages à n'importe quel moment. Ces messages contiennent des notes de l'utilisateur qui doivent être incorporées dans le flux de données, des avertissements ou des conditions d'erreurs. Les messages sont parmi les types suivants :

info - Information qui concerne seulement le processus d'acquisition courant.

warn - Un avertissement (warning) en provenance de la source de données.

error - Une erreur s'est produite dans la source de données.

crit_error - Une erreur rendant difficile ou impossible la suite des opérations s'est produite dans la source des données.

broadcast - Un message pour tous les processus qui sont actuellement connectés à la source de données.

Un message est toujours envoyé par la source de données sous la forme d'une balise XML unique, dont le titre inclu le type de message. En outre, le titre contient un attribut *time* qui indique l'horodatage du message en secondes et - en fonction du message - un attribut *text*, qui ne sera pas évalué par dlsd.

Une balise typique de message ressemblera à ceci :

```
<broadcast time="1093072549.866241" text="test message"/>
```

Le processus d'acquisition dlsd enregistre les messages dans le sous-répertoire *messages* du répertoire de la tâche (voir section 3.6). De la même manière que les données mesurées, les messages sont organisées en tronçons. Cependant, le concept de tronçon est légèrement différent ici : les tronçons contiennent des messages discontinus dans le temps et sont créés uniquement pour faciliter ultérieurement la suppression d'intervalle de temps dans les messages.

2.2.6. Traitement des signaux UNIX

Les signaux UNIX suivants sont traités par le processus d'acquisition dlsd :

SIGINT/SIGTERM Le processus d'acquisition doit se terminer. Aussitôt, il met fin à la connection vers la source de données et sauvegarde les données encore en mémoire sur le disque dur. Ceci peut prendre quelques secondes, car beaucoup de fichiers ont potentiellement besoin d'être écrits. Si aucune erreur ne se produit pendant l'opération, le processus se terminera avec la valeur de retour 0.

SIGHUP Quand le processus d'acquisition reçoit ce signal, il doit relire son fichier de spécifications. Par conséquent, il vérifiera immédiatement, - en fonction des nouvelles spécifications -, s'il doit continuer à acquérir des données. Sinon, il entamera une procédure d'arrêt comme dans le cas de *SIGINT* ou *SIGTERM*. Autrement, il enverra potentiellement les spécifications modifiées à la source de données et continuera à acquérir des données après confirmation (voir sous-section 2.2.3) des nouvelles conditions.

SIGCHLD Un "processus de nettoyage" (voir sous-section 2.2.4) s'est achevé. Il sera seulement enregistré via *syslogd*.

SIGSEGV et autres Le traitement est similaire à celui du processus parent (voir sous-section 2.1.3).

3. Le répertoire de données DLS

Toutes les données persistentes du système DLS sont organisées dans *les répertoires de données DLS* . La structure de base est indiquée dans Figure 3.1.

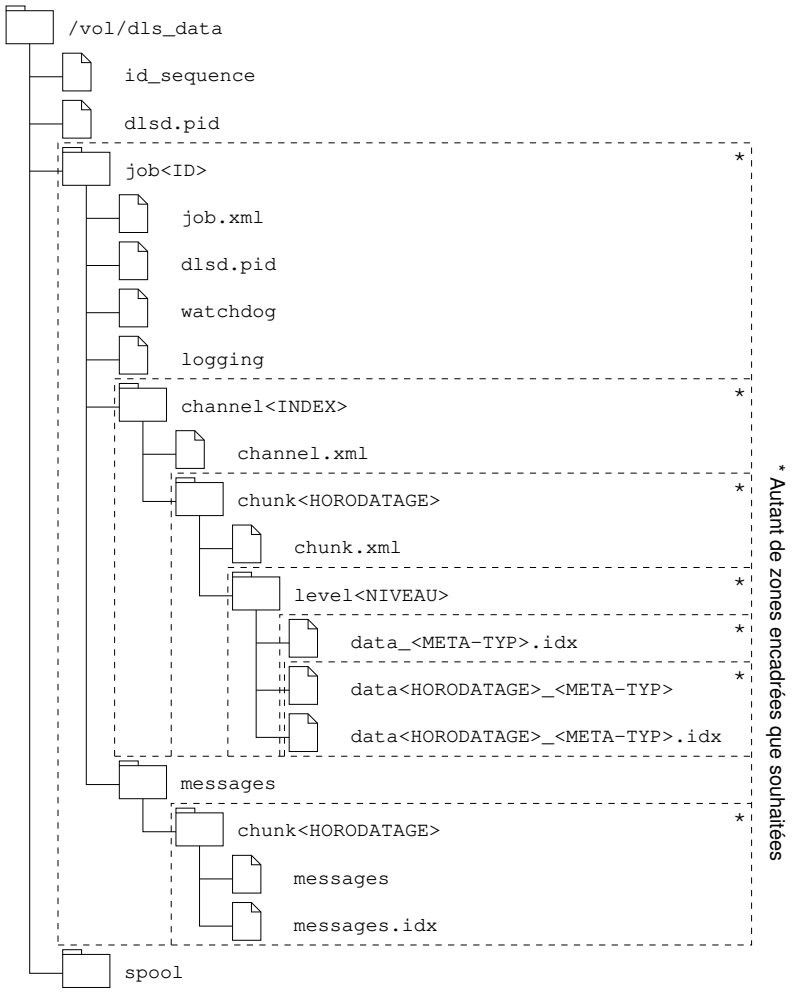

FIGURE 3.1. – Structure du répertoire de données DLS

3.1. Répertoire racine

Le répertoire racine est le répertoire de niveau le plus élevé au sein de l'arborescence de données DLS. La plupart des fichiers qui s'y trouvent, appartiennent au processus parent dlsd. De plus, le répertoire racine comprend tous les répertoires des tâches (voir section 3.2).

Fichiers et sous-répertoires dans le répertoire racine :

id_sequence Ce fichier contient le prochain identifiant (ID) disponible pour une tâche, sous la forme d'un nombre en ASCII. Il est nécessaire à DLS Manager lors de la création d'une nouvelle tâche. DLS Manager lit l'identifiant, l'utilise pour la nouvelle tâche, puis l'incrémente, avant d'écrire le nouveau identifiant dans le fichier.

dlsd.pid C'est le fichier *PID* du processus parent dlsd (voir Appendice C). Il est automatiquement créé à l'exécution et indique qu'un processus parent dlsd est en cours d'exécution.

jobXXX Chaque répertoire de tâche est présent dans le répertoire racine DLS. Le nom est toujours *job*, suivit par l'identifiant de tâche. Voir section 3.2.

spool C'est le répertoire de file d'attente du processus parent dlsd. Une description est donnée dans sous-section 2.1.2.

3.2. Répertoires de tâches

Pendant le fonctionnement en cours, chaque répertoire de tâche (*job<ID>*) est traité par un processus dédié d'acquisition dlsd.. Ce dernier lit ses spécifications et à partir de là, y écrit également les données acquises.

Fichiers et répertoires dans le répertoire de tâche :

job.xml Le fichier central des spécifications de la tâche. Il contient les spécifications de la tâche et celles des canaux. S'il est édité alors que le processus d'acquisition correspondant est en cours d'exécution, une commande de mise en file d'attente (voir sous-section 2.1.2) doit être générée pour que le processus adopte les nouvelles spécifications. DLS Manager le fait automatiquement.

Les spécifications sont dans le format XML et contiennent les informations suivantes (dont l'ordre est obligatoire !) :

```
<dlsjob>
 <description text="description"/>
 <state name="(running|paused)"/>
```

```
<source address="IP address or host name"/>
<quota size="data quota" time="time quota"/>
<trigger parameter="trigger parameter"/>

<channels>
 <channel name="channel name"
  frequency="sampling frequency"
  block_size="block size"
  meta_mask="meta mask"
  meta_reduction="meta reduction ratio"
  format="compression format"
  mdct_block_size="MDCT block size"
  mdct_accuracy="MDCT accuracy"
  type="data type"/>
 </channels>
</dlsjob>
```

Les attributs *mdct_block_size* et *mdct_accuracy* sont nécessaires uniquement si le format de compression choisi est MDCT (voir section 6.2).

Les spécifications peut être éditées avec DLS Manager. Les paramètres individuels sont décrits dans section 4.2 et section 4.5.

Chien de garde et journalisation Deux fichiers vides sont utilisés pour la surveillance des processus d'acquisitions dls par DLS Manager. Si un processus d'acquisition est en cours d'exécution pour un répertoire de tâche, il modifiera chaque seconde l'horodatage du fichier *watchdog*. Si le processus est aussi en train d'acquérir des données, il procèdera de la même manière avec le fichier *logging*. DLS Manager vérifie régulièrement l'horodatage de ces fichiers et peut ainsi connaître l'état du processus d'acquisition et en informer l'utilisateur.

dlsd.pid C'est le fichier *PID* du processus d'acquisition (voir Appendice C). Il est automatiquement créé au moment de l'exécution et indique que le processus d'acquisition dlsd est en cours de fonctionnement.

channelXXX Les données acquises continuent d'être organisées en canaux qui ont chacun leur propre répertoire (voir section 3.3). L'index dans le nom du répertoire de canal correspond à l'index de canal que la commande <*rk*> a renvoyée lors de l'énumération des canaux de la source de données pendant la séquence de démarrage du processus parent dlsd (voir sous-section 2.2.3).

messages Chaque processus d'acquisition stocke dans ce répertoire, les messages qu'il a reçu de la source pendant l'acquisition des données. Si le répertoire est manquant, le processus le créera au moment requis. De manière similaire aux données mesurées, les messages sont organisés en canaux. Voir section 3.6.

3.3. Répertoire de canal

L'ensemble des données acquises pour un canal donné sont rangées dans le répertoire de canal (*channel<INDEX>*). Un répertoire de canal est assigné en permanence à un canal spécifique de la source. Les propriétés du canal sont décrites dans le fichier *channels.xml* qui contient :

```
<dlschannel>
 <channel name="channel name" index="Index"
  unit="unit" type="Data type"/>
</dlschannel>
```

Le fichier sert non seulement à décrire les données dans les répertoires de tronçons mais aussi par le processus d'acquisition dlsd à chaque fois qu'une nouvelle acquisition de données doit être faite dans un répertoire de canal. Ceci ne doit avoir lieu que si les caractéristiques du canal (name, index, unit and type) n'ont pas changé.

3.4. Répertoire de tronçons

Les données acquises pour un canal sont organisées en *tronçons* (en anglais *chunks*) (*chunk<TIME>*). Un tronçon est une série de données dont l'acquisition est terminée et qui a été obtenue avec les mêmes spécifications de canal depuis une origine de temps donnée. L'horodatage du répertoire est le même que celui de la première donnée du tronçon. Les propriétés du tronçon sont décrites dans le fichier *chunk.xml* qui contient :

```
<dlschunk>
 <chunk sample_frequency="Fréquence d'échantillonnage"
  block_size="Taille de bloc pour les données"
  meta_mask="Méta-Masque"
  meta_reduction="Méta-réduction"
  format="Format de compression"
  mdct_block_size="Taille de block MDCT"
  mdct_accuracy="Précision MDCT"
  architecture="Architecture (Endianess)"/>
</dlschunk>
```

Les attributs *mdct_** existent si et seulement si le format de compression selectionné est MDCT (voir section 6.2).

Dans chaque répertoire de tronçon, les données sont rangées dans les répertoires correspondant à leur méta-niveau (les données génériques sont dans le répertoire *level0*, les données du premier méta-niveau sont dans le répertoire *level1* et ainsi de suite).

3.5. Répertoire de données

Les répertoires de données (*level<meta level>*), qui représent le tri des données en fonction de leur méta-niveau, sont arrangé en bas de la hierarchie des répertoires. Les fichiers de données et les fichiers d'index associés sont détaillé ci-dessous :

Fichier de données Les fichiers de données contiennent les données mesurées acquises. Ils sont créés séparément pour chaque méta-type. C'est pourquoi, le fichier suit la convention de nommage suivante :

data<HORODATAGE>_<METATYPE>

L'horodatage dans le nom de fichier est l'horodatage de la première valeur dans le premier bloc du fichier.

Dans le répertoire *level0*, le méta-type est toujours *gen* ("generic").

Les fichiers de données ont une structure XML simple. Chaque bloc de donnée apparaît dans une balise *<d>*. Celle-ci contient l'horodatage de la première valeur du bloc dans l'attribut *t* ("time"), le nombre de valeurs compressées dans l'attribut *s* ("size") et les données codées dans l'attribut *d*("data").

Les fichiers de données ont une taille maximale définie. Lorsque le processus d'acquisition dlsd est sur le point de dépasser cette taille en ajoutant un bloc, il débutera à la place un nouveau fichier de donnée.

Les paramètres qui ont été finalement utilisés pour l'acquisition de données et le type de compression employé ne peuvent être déterminés qu'en consultant les fichiers de description de haut-niveaux *chunk.xml* et *channel.xml*.

Fichiers d'index Les fichiers d'index sont des fichiers binaires avec une longueur d'enregistrement fixe, qui sont assignés à un fichier de données. Ils fournissent des informations qui peuvent être lues très rapidement, à propos des blocs dans les fichiers correspondants. La convention de nommage est similaire à celle des fichiers de données, mais avec un suffixe supplémentaire.

data<HORODATAGE>_<METATYPE>.idx

Les enregistrements du fichier d'index correspondent toujours à un bloc dans le fichier de données. La structure d'un enregistrement est expliquée dans Figure 3.2.

Chaque enregistrement contient 20 octets. Les 8 premiers octets contiennent l'horodatage en microsecondes (type *long long*) de la première valeur du bloc. Les 8 octets suivants correspondent de manière similaire à l'horodatage de la dernière valeur dans le bloc. Enfin, Les 4 derniers octets sont le décalage (type *unsigned int*) de la balise du bloc dans le fichier de donnée, c'est à dire la position du caractère initial "<".

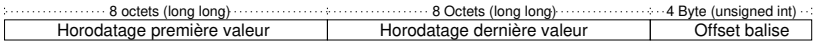

FIGURE 3.2. – Enregistrement du fichier d'index

Fichier d'index globaux Les fichiers d'index globaux permettent de déterminer plus facilement les intervalles de temps des fichiers individuels de données pour un méta-type particulier. Leur convention de nommage est :

data_<METATYPE>.idx

Un enregistrement dans un fichier d'index global correspond toujours à un fichier de donnée du même méta-type. La structure d'un enregistrement est expliquée dans Figure 3.3.

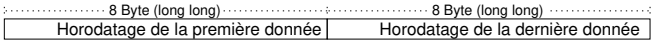

FIGURE 3.3. – Enregistrement d'un fichier d'index global

Un enregistrement dans un fichier d'index global fait toujours 16 octets de long. Les 8 premiers octets correspondent à l'horodatage en microsecondes (type *long long*) de la première valeur dans le fichier de données, les 8 derniers octets à l'horodatage de la dernière valeur dans le fichier de données.

Si une donnée vient juste d'être ajoutée dans un fichier de donnée, l'horodatage de l'enregistrement correspondant (le dernier) dans le fichier d'index global sera 0. Dans ce cas l'horodatage recherché doit être vérifié en lisant le dernier enregistrement dans le fichier d'index des données. Dès que l'acquisition des données est terminé dans le fichier de donnée, le second horodatage donnera la valeur "correcte".

3.6. Répertoire de messages

Les messages en provenance de la source de données sont stockés dans le répertoire de messages (*messages*) dans des tronçons séparés. Tous les répertoires de tronçons sont désignés par

chunk<HORODATAGE>

où l'horodatage correspond au premier message enregistré dans ce répertoire. Au sein des répertoires, il y a toujours seulement deux fichiers : le fichier des messages et le fichier d'index associé.

Fichiers de messages Un fichier avec des messages venant de la source de données est toujours nommé *messages*. Les messages de la source de données sont enregistrés sans transformation dans ce fichier, aussi il contient de simples balises XML qui correspondent aux messages respectifs (voir sous-section 2.2.5)).

Fichiers d'index de messages Le fichier d'index *messages.idx* appartient au fichier des messages et sert à activer le chargement rapide des messages pour un intevalle de temps donné sans devoir lire l'intégralité des messages.

Un enregistrement dans le fichier d'index correspond à un message dans le fichier de message. La structure d'un enregistrement est décrite dans Figure 3.4.

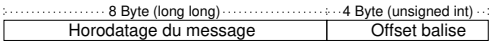

FIGURE 3.4. – Enregistrement d'index de message

4. DLS Manager

DLS Manager est une interface graphique utilisateur pour configurer les tâche d'acquisition dans un répertoire de données DLS. En outre, il sert au contrôle et à la surveillance des processus d'acquisition en cours d'éxécution.

DLS Manager est démarré par la commande `dls_ctl`.

De la même manière que dlsd, le paramètre `-d` dans la ligne de commande permet de spécifier le répertoire de données DLS dans lequel le programme doit travailler (voir section D.4).

Quand le programme DLS Manager démarre, il effectue automatiquement des vérifications :

— Si le répertoire de donnée DLS est vide, il demande à l'utilisateur si une structure valide de répertoires doit être créée à l'intérieur de celui-ci.
— S'il n'y a pas d'instance du démon dlsd en cours d'exécution, il demande à l'utilisateur s'il veut en démarrer une nouvelle.

4.1. Fenêtre principale

Figure 4.1 présente la fenêtre principale de DLS Manager, qui s'affiche lorsque le programme est démarré.

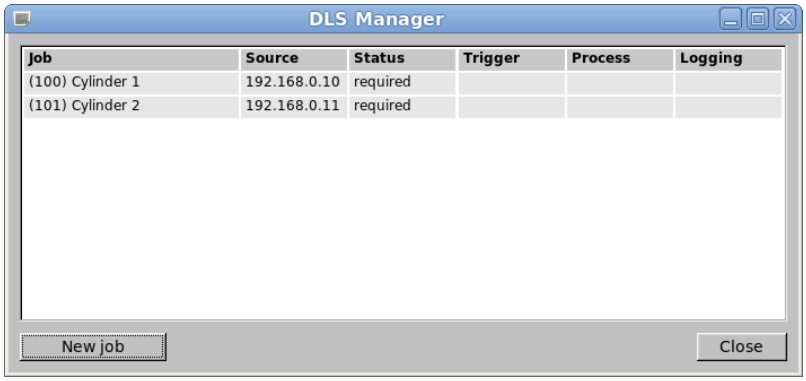

FIGURE 4.1. – Fenêtre principale de DLS Manager

4.1.1. Description

Dans la fenêtre principale, les tâches d'acquisition individuelles sont affichées sous forme de lignes dans un tableau. Les informations suivantes sont disponibles dans les colonnes :

Description La description de la tâche est arbitraire et peut être modifiée à tout moment. L'ID est un nombre fixe automatiquement attribué au moment de la création de la tâche. Toutes les données de la tâche sont enregistrées dans le répertoire de données DLS dans le sous-répertoire *job<ID>*.

Source Le nom d'hôte ou l'adresse IP du serveur qui sert de source de données. Sur ce dernier un serveur de contrôle et commandes doit être accessible via le port TCP 2345. La source doit être obligatoirement choisie au moment de la création de la tâche et elle ne peut plus être modifiée par la suite.

Status Le status de la tâche selectionnée par l'utilisateur : "started", quand l'acquisition de données est démarée, sinon "stopped".

Trigger . Le paramètre de la source de donnée qui servira de gâchette pour le processus d'acquisition de données. Lorsque qu'un paramètre de gâchette est selectionné, les données sont acquises seulement lorsque la gâchette vaut 1.

Process Indique si un processus d'acquisition est en cours d'exécution pour cette tâche. Rien n'est affiché quand la tâche est arrêtée.

Acquisition Si une gâchette a été configurée, vous pouvez voir ici si elle est activée. Sans gâchette, l'acquisition de données est toujours activée quand le processus d'acquisision est en cours d'exécution.

4.1.2. Utilisation

— Le lignes contenant les tâches individuelles peut être selectionnées avec le curseur de la souris.
— Quand une tâche est sélectionnée, un bouton *start* (démarrer) ou *stop* est affiché pour contrôler l'acquisition des données.
— Un double-click sur une tâche ouvre un dialogue pour éditer la tâche correspondante (voir section 4.3).
— Le bouton "Close" termine le programme.

4.2. Les dialogues "Create job" et "Change job"

Figure 4.2 montre le dialogue pour créer ou modifier une tâche d'acquisition. Elle s'affiche après avoir cliqué sur le bouton "New job" de la fenêtre principale ou le bouton "Change" dans le dialogue "Edit job". La seule différence entre les deux est que la source de données ne peut être modifiée que lors de la création de la tâche.

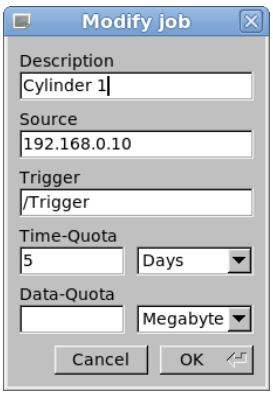

FIGURE 4.2. – Dialogue pour créer ou modifier une tâche d’acquisition

4.2.1. Description

Les différents champs du dialogue permettent à l’utilisateur d’indiquer les particularités de la tâche d’acquisition :

Description C’est un nom arbitraire pour la tâche d’aquisition qui sert seulement pour simplifier l’identification.

Source L’adresse de la source de données. Ça peut être un nom d’hôte ou une adresse IP. Si un nom d’hôte est utilisé, il faut qu’il puisse être résolu par dlsd au moment de l’exécution. L’hôte spécifié doit fournir une source de donnée pour l’acquisition et pour les canaux additionnels via le protocole correspondant (see section 4.4).

Trigger Le nom du paramètre qui servira de gâchette. Si un paramètre de gâchette a été sélectionné ici, les données ne seront acquises que si ce paramètre a la valeur 1. Si le champ d’entrée est vide, aucune gâchette n’est utilisée.

Time quota Le quota de temps (la durée maximale de rétention des données acquises voir sous-section 2.2.4) peut être indiqué ici sous la forme d’un entier et d’une unité de temps (jours, heures, minutes ou secondes). Si le champ d’entrée est vide, alors aucun quota de temps n’est appliqué.

Data quota Le quota de stockage (le volume maximal de stockage réservé pour les données acquises) peut être indiqué sous la forme d’un entier et d’une unité de taille. Si le champ d’entrée est vide, alors aucun quota de stockage n’est appliqué.

4.2.2. Utilisation

— Les champs sont vérifiées en cliquant sur le bouton “OK” (ou en pressant la touche *Entrée*. S’ils contiennent des erreurs, une fenêtre sera affichée avec les

messages exactes des erreurs. Autrement, les champs d'entrées sont acceptés et le dialogue est fermé. Si un processus d'acquisition dlsd est en cours d'exécution, il lui sera demandé de tenir compte des nouvelles spécifications.

— Si le bouton "Cancel" (Annuler) est cliqué, les champs d'entrées sont annulés et le dialogue est fermé.

4.3. Le dialogue "Edit job"

Figure 4.3 montre le dialogue pour éditer une tâche. Elle s'affiche lors d'un double-clic sur une tâche de la fenêtre principale.

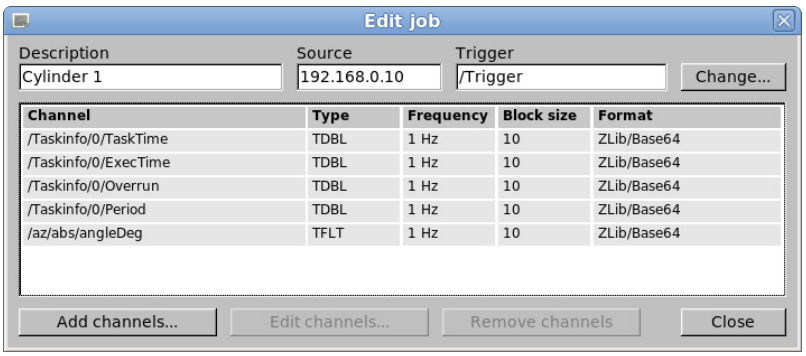

FIGURE 4.3. – Dialogue pour éditer une tâche

4.3.1. Description

La source des données est affichée en haut. En dessous, se trouve la liste des canaux à inclure et leurs paramètres clés. En particulier :

Channel Le nom du canal à inclure

Type Le type de canal. Un canal peut être de type entier ou virgule flottante (voir Appendice B).

Scanning frequency La fréquence d'échantillonnage, c'est-à-dire, la fréquence à laquelle les données sont enregistrées.

Block size Le nombre de données acquises qui doivent être compressées et sauvegardés ensemble pour former une unité.

Format La méthode de compression (voir chapitre 6).

4.3.2. Utilisation

— le dialogue pour éditer les caractéristiques principales d’une tâche peut être affiché en cliquant sur le bouton “Change”.
— Les lignes du tableau des canaux peut être sélectionnées avec le curseur de la souris. Il est alors possible d’utiliser les boutons “Edit channels” et “Delete channels”.
— En pressant les touches *Shift* ou *Ctrl*, vous pouvez sélectionner plusieurs canaux à la fois, qui peuvent alors être édités ou supprimés simultanément.
— Les spécifications pour un ou plusieurs canaux sélectionnés peuvent être éditées en cliquant sur les boutons “Edit channels.” du dialogue suivant. Cependant, des conditions spéciales s’appliquent pour éditer plusieurs canaux en parallèle (voir sous-section 4.5.3).
— Un double clic sur une ligne d’un canal ouvre aussi le dialogue pour éditer les spécifications du canal correspondant.
— Si le bouton “Delete channels” est cliqué, tous les canaux sélectionnés sont retirés des spécifications. Cependant les données acquises restent disponibles.
— Lorsque le bouton “Add channels” est cliqué, le dialogue pour ajouter des canaux aux spécifications s’ouvre (see section 4.4).
— Le bouton “Close” ferme le dialogue et retourne à la fenêtre principale.

4.4. Le dialogue “Add channels”

En cliquant le bouton “Add channels” dans le dialogue pour éditer une tâche, une fenêtre s’ouvre comme montré dans Figure 4.4. Au même instant, le programme essaye d’établir une connexion TCP vers la source de données pour récupérer la liste des canaux.

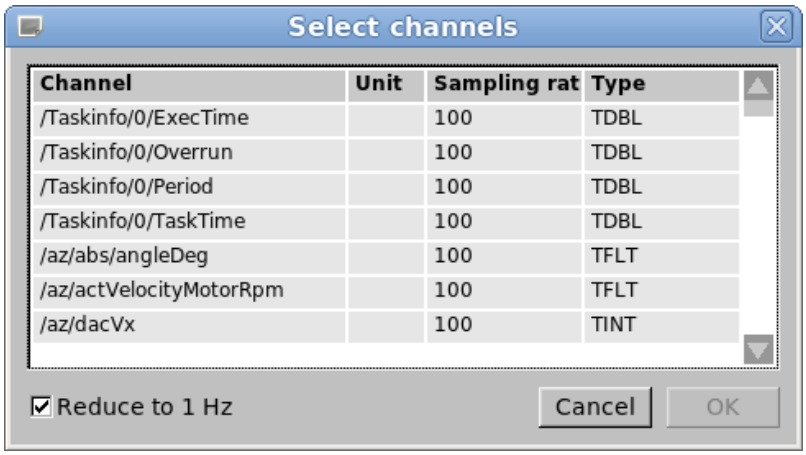

FIGURE 4.4. – Dialogue pour ajouter des spécifications de canaux

4.4.1. Description de l'affichage

— Tandis qu'il essaye d'établir la connexion avec la source de donnée, le message "Receiving channels. . ." s'affiche. Si après un temps d'attente pré-défini la connexion ne parvient toujours pas à être établie, une fenêtre avec le message d'erreur correspondant apparaît et le dialogue se ferme.
— Si la liste des canaux a été récupérée avec succès, les canaux individuels sont affichés dans un tableau. Le nom du canal, l'unité, la fréquence maximale d'échantillonnage et le type de donnée sont affichés.

4.4.2. Utilisation

— L'utilisateur peut utiliser le curseur de la souris pour sélectionner les canaux individuels. En pressant les touches *Ctrl* ou *Shift*, il est possible de sélectionner plusieurs canaux à la fois.
— Lorsque le bouton "OK" est cliqué, tous les canaux sélectionnés sont ajoutés à la liste des spécifications de canaux de la tâche. Si un canal particulier est déjà disponible, il sera affiché dans une fenêtre avec le message d'avertissement correspondant. Les autres canaux seront néanmoins ajoutés. Si des modifications ont été faites, le processus d'acquisition dlsd en cours d'exécution adoptera les nouvelles spécifications des canaux.
— Un clic sur le bouton "Cancel" ferme le dialogue sans ajouter de nouvelles spécifications de canaux au tâche.

4.5. Le dialogue "Edit channels"

Le dialogue pour éditer les spécifications d'un canal, tel que montré dans Figure 4.5 apparaît lors d'un double-clic sur la spécification d'un canal dans le dialogue pour éditer une tâche, ou bien après un clic sur le bouton "Edit channels" après avoir sélectionné un ou plusieurs canaux.

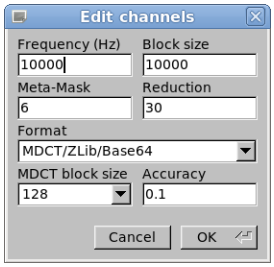

FIGURE 4.5. – Dialogue pour éditer les spécifications d'un canal

4.5.1. Description de l’affichage

Les champs d’entrée suivants sont disponibles pour l’utilisateur pour paramétrer les spécifications d’un canal.

Sample Rate Fréquence d’échantillonnage, c’est-à-dire le nombre de valeur stockées par seconde. Cette fréquence doit être un diviseur entier de la fréquence maximale d’échantillonnage du canal concerné.

Block Size Le nombre de valeurs compressées et stockées dans un bloc. Plus le bloc est grand, meilleurs sont les résultats de la compression, mais plus celle-ci dure longtemps. Pour une compression MDCT, la taille de bloc doit être un multiple entier de la taille du bloc MDCT.

Meta mask *Cette valeur n’est actuellement pas éditable*
Le méta-masque est un masque binaire qui indique les méta-données qui sont enregistrées. Voir sous-section 2.2.2.

Reduction ratio Le raport de méta-réduction est le nombre de valeurs d’un méta-niveau. Voir aussi sous-section 2.2.2. Cette valeur ne requière habituellement aucun ajustement.

Format Le format de compression dans lequel les données acquises sont compressées avant l’enregistrement. En fonction de la méthode de compression, des paramètres additionnels ont besoin d’être spécifiés.

MDCT block size Ce paramètre doit être spécifié pour les méthodes de compression MDCT et se comporte de manière similaire à la taille de bloc. Voir section 6.2.

Accuracy Certaines méthodes de compression avec pertes autorisent la spécification d’une erreur absolue maximale. L’erreur doit toujours être indiquée dans l’unité du canal correspondant.

4.5.2. Utilisation

— Lorsque le bouton “OK” est cliqué, le programme vérifie tout d’abord la plausibilité des paramètres. Si cette vérification échoue, une fenêtre s’affiche avec le message d’erreur adéquat. Autremenent, tous les paramètres spécifiés sont appliqués aux canaux préalablement sélectionnés, et le processus d’acquisision dlsd en cours d’exécution adopte les nouvelles spécifications des canaux, puis le dialogue se ferme.
— Si le bouton “Cancel” est cliqué, les paramètres sont ignorés et le dialogue se ferme.

4.5.3. Édition simultanée de plusieurs spécifications de canaux

Le dialogue pour éditer les spécifications des canaux (Figure 4.5) peut aussi être utilisé pour éditer simultanément plusieurs canaux. Dans ce cas, les conditions suivantes s'appliquent :

- Lors de l'ouverture du dialogue, tous les paramètres qui sont identiques pour tous les canaux à éditer seront affichés dans les champs d'entrées du dialogue. En revanche, les paramètres qui **ne sont pas** identiques pour tous les canaux auront des champs d'entrées vides.
- Après avoir changé ou ajouté une valeur dans le dialogue , un clic sur le bouton "OK" affectera **tous** les canaux préalablement sélectionnés.
- Si un champ d'entrée est vide au moment de cliquer sur "OK", ce valeur ne sera changé dans **aucune** spécification de canaux. Dans ce cas, les canaux préalablement sélectionnés conververont leurs valeurs respectives. Ainsi, il est possible, par exemple, de ne modifier que la fréquence d'échantillonnage pour un certain nombre de spécification de canaux.
- Les paramètres de compression (*format*, *taille de bloc MDC* et *precision*) sont traités comme une seule entité. Cela signifie que les paramètres de compression seront initiallement affichés uniquement s'ils sont exactement les mêmes pour **toutes** les spécifications des canaux. Mais ils peuvent aussi être édités collectivement.

5. DLS View

Le programme *DLS View* permet d'afficher une vue simple des données acquises par une tâche d'acquisition. C'est pourquoi il est composé de seulement deux fenêtres (voir Figure 5.1 et Figure 5.3).

5.1. Fenêtre principale

FIGURE 5.1. – Fenêtre principale de DLS View

5.1.1. Description

— Dans le coin supérieur gauche de l'écran, il y a un champ pour sélectionner la tâche d'acquisition dont les données doivent être affichées.

— Sur le coté droit est affiché la liste des canaux contenus dans la tâche qui a été sélectionnée.
— La majeure partie de la fenêtre est utilisée pour afficher les données. Son architecture permet d'afficher les données de plusieurs canaux les uns au dessous des autres en partageant le même axe temporel, affiché en haut du graphique. Les graduations apparaissent sous la forme de ligne verticales grises dans un système de coordonnées.
— L'intervalle de temps ainsi représenté est indiquée dans une petite ligne de texte tout en bas du graphique.
— Un petit entête au dessus de chaque canal, contient le nom du canal, la plage des valeurs affichées, le nombre de blocs de données chargées et le méta niveau utilisé (voir sous-section 2.2.2). Sous l'entête, les données sont représentées sous forme de courbes. Une courbe blue indique que les données affichées sont des données génériques (méta-niveau 0), si un méta-niveau supérieur est utilisé, alors la courbe correspondante est verte.
— Si aucune donnée n'est disponible pour l'intervalle de temps à afficher, alors la rangée du canal est affiché avec un fond jaune (voir Figure 5.2).
— Puisque toutes les canaux doivent partager la hauteur globale de l'affichage, la hauteur de chaque rangée diminue avec le nombre de canaux à afficher. Si cette hauteur devient inférieure à une valeur définie, une barre de défilement apparaît à droite.

5.1.2. Utilisation

— Dans la liste de sélection "Job", l'utilisateur peut choisir la tâche d'acquisition qu'il veut afficher. La liste des canaux acquis est alors mise à jour.
— L'utilisateur peut afficher ou cacher les canaux individuels dans la zone d'affichage en cochant les cases situées devant leurs noms respectifs dans la liste des canaux.
— Cliquer sur le bouton "All" détermine et affiche l'intervalle de temps complet dans lequel les données ont été acquises pour les canaux selectionnés.
Si des canaux sont ajoutés ou supprimés immédiatement après, le nouvel intervalle de temps est recalculé à nouveau. Cependant, si l'utilisateur choisit un autre intervalle de temps à afficher ce dernier est constamment appliqué même si des canaux sont ajoutés ou supprimés.
— Cliquer sur le bouton "Update" recharge et réaffiche toutes les données de l'intervalle de temps sélectionné.
— Cliquer sur le bouton "Export..." ouvre le dialogue d'exportation (voir section 5.2).
— En appuyant et tenant le bouton gauche de la souris dans la zone de donnée de l'affichage, l'utilisateur peut sélectionner un nouvel intervalle de temps qui sera indiqué par deux lignes rouges verticales tant que le bouton de la souris est maintenu enfoncé. Les valeurs temporelles exactes sont indiquées sur le bord

supérieur du graphique. Lorsque le bouton gauche de la souris est relâché, le nouvel intervalle de temps est accepté et les données correspondantes sont chargées.
De cette façon, le relâchement du bouton de la souris peut être en dehors de la zone d'affichage, ce qui permet d'étendre légèrement l'intervalle de temps affiché.
— De manière similaire, en appuyant et tenant le bouton droit de la souris dans la zone d'affichage, l'intervalle de temps présenté peut être déplacé. Lorsque le bouton de la souris est relâché, le nouvel intervalle de temps est accepté.
— Un double clic dans la zone d'affichage étend d'un facteur deux l'intervalle de temps présenté. Au préalable, il est centré autour du point temporel qui a été cliqué. Si la touche *Shift* est maintenue enfoncée pendant le double clic, un facteur 10 est utilisé pour l'extension.
— Si la zone d'affichage des données a le focus du clavier, l'appui sur la touche *Ctrl* dessine une ligne verticale de balayage qui passe par le point temporel pointé par la souris (voir Figure 5.2). Si cette ligne croise une courbe affichée, la valeur de la donnée au point d'intersection sera affichée.
Comme la ligne de balayage n'est pas infiniment étroite, il est possible que plusieurs valeurs de données soient concernées dans la zone couverte par celle-ci. Dans ce cas, la zone entière de valeur des valeurs qui sont présentes “sous“ la ligne de balayage sera marqu'ee par deux lignes horizontales correspondant au minimum et maximum (voir le troisième canal dans Figure 5.2).

5.2. Le dialogue “Export. . .”

5.2.1. Description

— La partie supérieure du dialogue (voir Figure 5.3) montre le nombre de canaux sélectionnés et l'intervalle de temps choisi. L'exportation inclut toujours les données affichées dans la vue actuellement selectionné dans la fenêtre principale.
— La barre de progression dans la partie inférieure montrera par la suite la progression de l' exportation.

5.2.2. Utilisation

— Dans la partie centrale du dialogue (voir Figure 5.3) cocher les cases pour sélectionner les formats d'exportation.
— Un clic sur le bouton “Export” démarre l'exportation des données. Si la variable d'environnement $DLS_EXPORT est définie, les données seront écrites dans le répertoire correspondant. Sinon, le répertoire courant sera utilisé. Pour

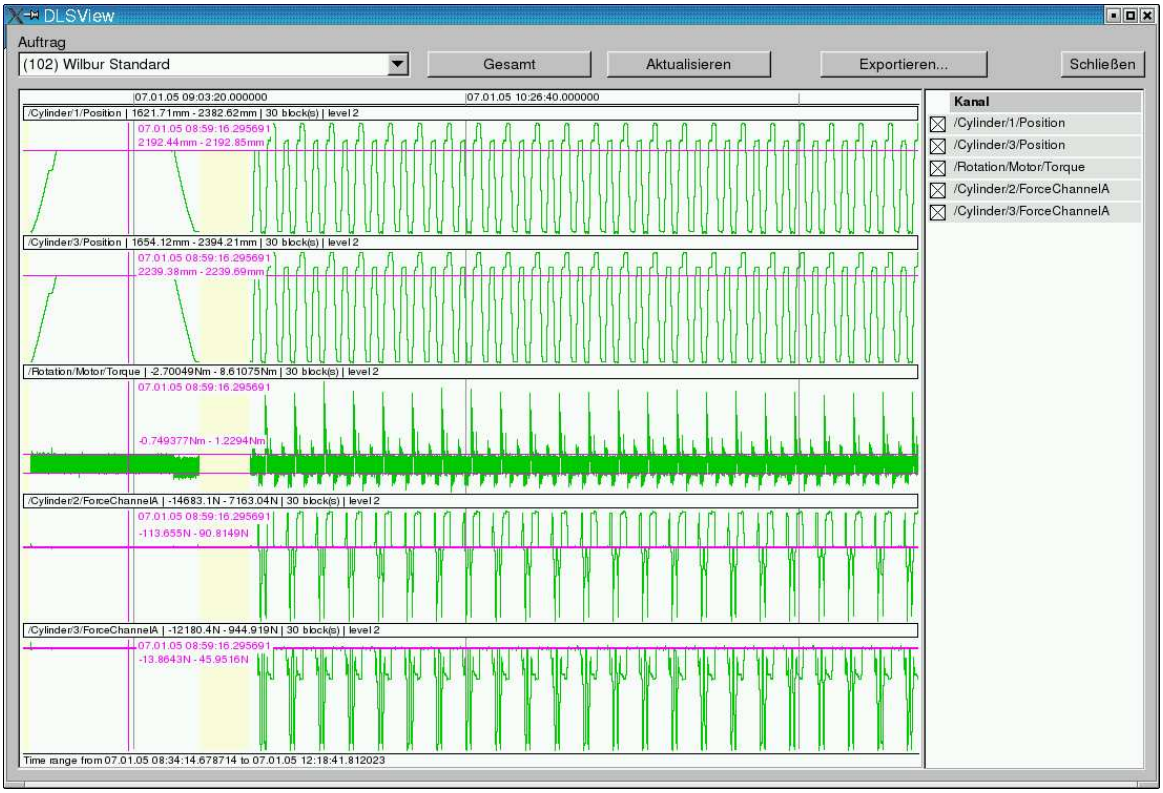

FIGURE 5.2. – Ligne de balayage avec le bouton *Ctrl* maintenu enfoncé

éviter d'écraser des données lors de l'exportation, un sous-répertoire sera toujours créé pour accueillr les fichiers de données. Le nom de ce sous-répertoire est déterminé par la variable d'environnemnent $DLS_EXPORT_FMT, qui accepte les caractères génériques conformément aux conventions de la fonction c strftime(). Voir `man 3 strftime` pour la liste. Si la variable d'environnement n'a pas été définie, le nom de répertoire par défaut *dls-export-%Y-%m-%d-%H-%M-%S* est utilisé.

— Le bouton "Cancel" interrompt l'exportation et ferme le dialogue.

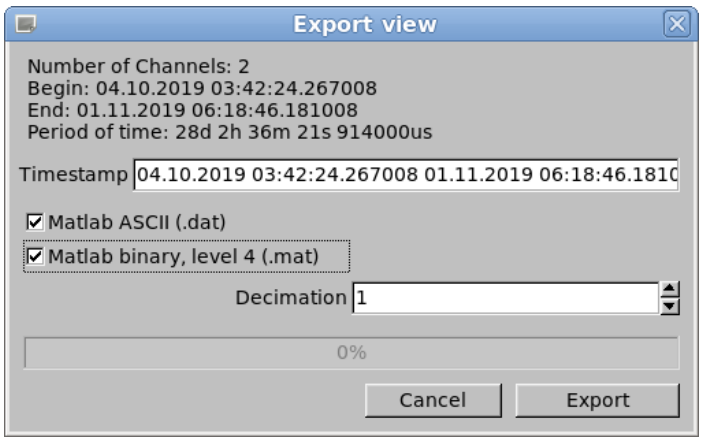

FIGURE 5.3. – Le dialogue d’exportation de DLS View

6. Méthodes de compression

La compression des données acquises reçues depuis les sources de données sert à réduire l'espace de stockage requis dans le système de fichier. Il s'applique toujours par bloc (c'est-à-dire un certain nombre de valeurs qui sont toujours compressées ensemble) dont l'utilisateur peut ajuster la taille dans les spécifications du canal. Une distinction fondamentale est faite entre la compression sans-perte et celle avec perte.

Le système DLS supporte différents algorithmes de compression. Puisque la plupart des algorithmes produisent une sortie au format binaire, elles seront converties en Base64 avant d'être enregistrées dans les fichiers de données. Cependant, cela augmente l'espace de stockage d'un tiers, mais avec l'avantage que les données compressées sont disponibles sous forme de caractères "imprimables" et par conséquent codables en XML. C'est pourquoi, toutes les méthodes de compression de DLS ont le suffixe */Base64*.

Les méthodes de compression suivantes sont supportées par DLS :

ZLib/Base64 Une méthode de compression simple mais efficace offrant une compression sans pertes. Voir section 6.1.

MDCT/ZLib/Base64 Une méthode de compression améliorée qui traite les données par une transformation et une quantification, puis enfin les compresse. Voir section 6.2.

6.1. Compression avec ZLib

Méthode de compression : ZLib/Base64
Type de données compressibles : toutes.

La bibliothèque "ZLib" (`http://www.gzip.org/zlib`) fournit des fonctions pour la compression sans perte de données. Le processus d'acquisition dlsd utilise ces fonctions dans la méthode de compression ZLib/Base64. Par ailleurs, l'algorithme ZLib est utilisé pour prendre en charge d'autres processus ultérieurs pour compresser davantage les données déjà traitées.

Comme ZLib produit des sorties binaires, elles sont enregistrées en Base64 pour tous les processus.

6.2. Compression avec MDCT

Méthode de compression : MDCT/ZLib/Base64
Type de données compressibles : *TFLT* (float), *TDBL* (double)

La méthode de compression MDCT ("modified, discrete cosinus transformation") de DLS est un processus hybride dans lequel les données sont premièrement transformées avec MDCT, puis quantifiées et finalement transposées par bit, dans le but de rendre plus efficace leur compression avec ZLib. Le but ici, est de compresser les données acquises avec une erreur limitée.

6.2.1. MDCT

MDCT est une sorte d'équivalent discret de la transformée de Fourier qui transforme un signal dans son équivalent dans le domaine fréquentiel qui à sont tour permet de retrouver le signal original.

Alors que la transformation "normale" DCT est fondé sur le principe que n valeurs sont transformées en n coefficients à partir desquels le signal original peut être complètement récupéré, la transformation "modifiée" DCT transforme toujours n valeurs en $\frac{n}{2}$ coefficients qui forme une représentation incomplète du signal original. Cependant, comme la transformation est effectuée avec un recouvrement de 50%, le signal original peut être recupéré en re-chevauchant et re-transformant deux séquences successives de coefficients. Cette méthode est expliquée dans Figure 6.1.

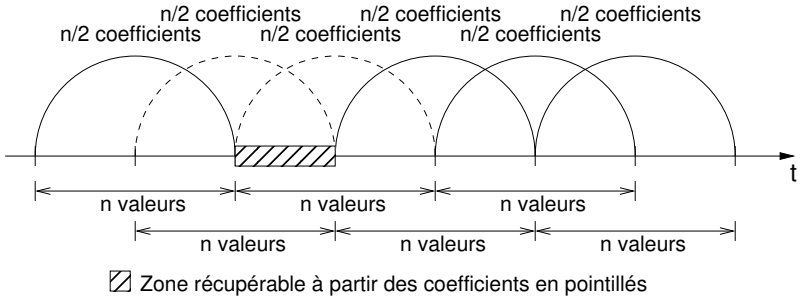

FIGURE 6.1. – MDCT : Modified, discrete cosinus transformation

Ce DCT modifiée est complétée par le recouvrement des valeurs à transformer au moyen d'une fonction fenêtre qui donne un poids plus faible aux valeurs à la marge. Ceci empêche que différentes erreurs sur les zones marginales des transformations ne conduisent à des "artefacts" afin que le signal original puisse être récupé de manière transparente.

6.2.2. Quantification

Les coefficients constatés par MDCT sont soumis à une quantification entière. Ici une méthode de bisection permet de décider combien de bits au minimum la quantification doit conserver pour éviter que l'erreur maximum commise ne devienne trop importante lors de la transformation inverse. La méthode de quantification fournit non seulement des coefficients quantifiés sur n bits mais aussi le facteur d'échelle flottant requis pour récupérer les coefficients originaux.

Comme l'erreur maximale commise ne peut pas être déterminée avec précision pour une transformation MDCT inverse individuelle, elle est estimée comme la moitié de l'erreur donnée. Quand deux séquences de retransformations des coefficients sont superposées pour récupérer le signal original, l'erreur donnée ne peux être dépassée aussi longtemps que l'erreur absolue dans les deux signaux partiels ne dépasse pas la moitié de l'erreur maximale totale.

Les signaux des processus techniques sont souvent les plus adaptés à une transformation par MDCT car, dans la plupart des cas, ils résultents de vibrations harmoniques superposées. Cela signifie que les portions hautes fréquences des coefficients pertinents sont souvent peu prononcés et peuvent être largement adaptés par la quantification. Cela conduira ultérieurement à une bonne compressibilité.

6.2.3. Transposition

La transposition est nécessaire parce que la méthode de compression ZLib fonctionne par octet et dans la plupart des cas elle ne parvient pas à reconnaître des modèles de bits similaires. Aussi, les bits individuels des coefficients quantifiés sont retriés en mémoire. Dans ce but, les coefficients sont premièrement séparés de leurs signes algébriques. Les bits de signes sont enregistrés séparément avant les bits des coefficients. Ils sont premièrement suivis par tous les *MSB*s ("Most Significant Bits") (bits de poids forts) des coefficients et enfin par tous les *LSB*s ("Least Significant Bits") (bits de poids faibles). En raison des coefficients généralement très faibles des fréquences les plus élevées, un grand nombre d'octets sont à zéro et peuvent être facilement compressés.

Par conséquent, un bloc MDCT compressé se compose du facteur d'échelle des coefficients quantifiés (4 ou 8 octets), la quantité q de bits de quantification utilisés (1 octet), de n bits de signe et enfin de $q \cdot (n-1)$ bits de coefficients.

6.2.4. MDCT via FFT (Fast Fourier Transform)

Marios Athineos[1] a développé une méthode pour réduire un MDCT portant sur n valeurs à une transformée de Fourier portant sur $\frac{n}{4}$ valeurs. DLS utilise cette

1. marios@ee.columbia.edu, http://www.ee.columbia.edu/~marios, Columbia University

méthode en combinaison avec la bibliothèque *FFTW3* (voir Appendice A) pour réduire considérablement la quantité de calcul nécessaire. Cette bibliothèque combine des algorithmes efficaces pour le calcul de la transformée de Fourier avec l'utilisation des extentions processeurs telles que MMX ou SSE.

6.3. Compression au travers de la quantification

Méthode de compression : Quant/ZLib/Base64
Types de données compressibles : *TFLT* (float), *TDBL* (double)

Cette méthode compression affaiblie soumet les valeurs de données à compresser à une quantification absolue, les différencie et enfin les stocke sous forme transposée. Cette méthode prépare les "données brutes" afin que la compression ultérieure avec ZLib soit encore plus efficace.

6.3.1. Quantification

Durant la quantification, les valeurs (en virgule flottante) sont distribuées via un facteur d'échelle sur un intervalle limité d'entiers naturels. Une compression est obtenue en essayant de garder cet intervalle aussi petit que possible afin de pouvoir coder les valeurs quantifiées avec quelques bits. Cependant, cela n'est fait que si l'erreur produite reste sous une certaine limite spécifiée par l'utilisateur.

6.3.2. Différentiation

De plus, les valeurs quantifiées sont différenciées pour que les formes d'onde du signal linéaire deviennent plus harmonisées dans le codage afin que l'algorithme ZLib puisse mieux compresser les données. À cette fin, au début de l'ensemble de données compressés, le décalage (entier) est stocké, et à partir de là, seule la différence de valeur à valeur est stockée.

6.3.3. Transposition

La transposition est faite pour les même raisons que pour **MDCT/ZLib/Base64** method. Voir sous-section 6.2.3 à ce sujet.

A. Installation de DLS

A.1. Configuration requise

DLS est principalement implémenté avec le langage de programmation C++. Pour sa compilation et son fonctionnement DLS a besoin du système d'exploitation Linux.

Les logiciels suivants doivent être installés pour la compilation et l'exécution :

- *syslogd*, qui fait habituellement partie de chaque distribution Linux, est utilisé pour enregistré les messages émits pendant l'exécution.
- Pour la compilation de l'interface graphique utilisateur de DLS Manager et DLS View, il est nécessaire d'avoir la bibliothèque graphique *FLTK* en version 1.1. Celle ci peut être téléchargée depuis le site web FLTK `http://www.fltk.org`. La bibliothèque doit être compilée avec le support pour le multithreading [1] (option de configuration `--enable-threads`).
- La bibliothèque *ZLib* est requise pour la compression. Elle est incluse dans quasiment toutes les distributions Linux. En cas de besoin, elle peut être téléchargée depuis `http://www.gzip.org/zlib` et installée.
- La bibliothèque *FFTW3* est aussi requise pour la compression. Elle permet à DLS de calculer la transformée de Fourier pour la compression MDCT. La bibliothèque peut être téléchargée depuis `http://www.fftw.org/download.html`.
- Pour DLS Manager et FLTK, vous aurez besoin de la bibliothèque *pthreads*.

A.2. Installation

Après avoir copié l'archive DLS depuis le CD ® ou bien depuis le site web EtherLab® `http://etherlab.org`, vous pouvez l'extraire :

```
$ tar xjf dls-1.0-rXXX.tar.bz2
$ cd dls-1.0-rXXX.tar.bz2
```

1. Malheureusement, certaines distributions ne fournissent qu'un paquet sans multithreadind, dans ce cas la bibliothèque FLTK doit être recompilée.

Maintenant le code source peut être configuré et compilé avec les commande mentionnées ci-dessous. La commande `configure` reconnaît les paramètres `--with-fltk-dir` et `--with-fftw3-dir` pour spécifier le répertoire d'installation des bibliothèques correspondantes. Le répertoire d'installation par défaut de DLS est */opt/etherlab*. Un autre répertoire d'installation peut être spécifié avec le paramètre `--prefix`.

```
$ ./configure
$ make
```

L'appel suivant (sous *root*) installera tous les exécutables, scripts et modèles de fichiers de configurations nécessaires.

```
# make install
```

A.3. Configurer DLS en tant que service

Si DLS doit être configuré en tant que service, les scripts *init*, *sysconfig* et *profile* doivent être copiés dans les répertoires appropriés de la distribution Linux. Les commandes suivantes sont adaptées pour la distribution Linux SUSE mais seront légèrement différentes pour d'autres distributions :

```
# cd /opt/etherlab
# cp etc/init.d/dls /etc/init.d/dls
# cp etc/sysconfig/dls /etc/sysconfig/dls
# cp etc/profile.d/dls /etc/profile.d/dls
# insserv dls
```

La configuration se fait en ajustant le fichier */etc/sysconfig/dls*. Les variables de configurations pertinentes sont documentées dans le fichier. Elles seront ensuite exportées comme variables d'environnement par le script *profile* et mise à disposition de tous les utilisateurs.

Le répertoire de données DLS sera automatiquement créé au démarrage de DLS manager. Pour cela, la variable d'environnement $DLS_DIR doit être définie ou bien le répertoire à utiliser doit être spécifié avec le paramètre `-d`. Si le répertoire indiqué n'est pas encore un répetoire de donnée DLS, le programme demande à l'utilisateur s'il veut l'initialiser ainsi.

B. Type de données

Tableau B.1 montre tous les types de données supportés pour les canaux et les méthodes de compressions possibles.

TABLE B.1. – Supported channel data types

Type	**Description**	**Compression**
TCHAR	Entier 1 octet (avec signe)	ZLib/Base64
TUCHAR	Entier 1 octet (sans signe)	ZLib/Base64
TINT	Entier 4 octets (avec signe)	ZLib/Base64
TUINT	Entier 4 octets (sans signe)	ZLib/Base64
TLINT	Entier 8 octets (avec signe)	ZLib/Base64
TULINT	Entier 8 octets (sans signe)	ZLib/Base64
TFLT	Virgule flottante 4 octets	ZLib/Base64, MDCT/ZLib/Base64, Quant/ZLib/Base64
TDBL	Virgule flottante 8 octets	ZLib/Base64, MDCT/ZLib/Base64, Quant/ZLib/Base64

C. Fichiers PID

À plusieurs endroits, le système DLS utilise des fichiers *PID*, un mécanisme pour éviter que plusieurs processus n'essayent d'exécuter une tâche unique. Un fichier *PID* contient, sous forme *ASCII*, le processus ID (*PID*) du processus en cours d'exécution. Après le démarrage de chaque processus, celui-ci vérifie si le fichier PID et le processus associé PID existent. Si les deux existent déjà, un nouveau processus ne doit pas être démarré et donc le nouveau processus doit s'arreter immédiatement. Si aucune autre instance n'existe, le nouveau processus doit continuer son exécution et créer un nouveau fichier PID. Avant cela, le fichier *PID* obsolète (c'est-à-dire que le processus spécifié n'existe plus) peut être supprimé.

D. Paramètre de ligne de commande

D.1. dlsd

```
dlsd 1.4.0-rc2 revision de0a3e76b9ae
Usage: dlsd [OPTIONS]
  -d <dir>       Set DLS data directory.
  -u <user>      Switch to <user>.
  -n <number>    Set maximal number of open files.
  -k             Do not detach from console.
  -w <seconds>   Wait time before restarting logging
                   process after an error. Default is 30.
  -b             Do not bind to network socket.
  -p <port>      Listen port or service name. Default is 53584.
  -r             Read-only mode (no data logging).
  -h             Show this help.
```

D.2. Script d'initialisation

(Le chemin peut varier suivant les distributions Linux.)

```
USAGE: /etc/init.d/dls {start|stop|restart|status}
```

D.3. dls_status

```
Call: dls_status [OPTIONS]
Options:
        -d [directory]     DLS data directory
        -h                 Show this help
```

D.4. dls_ctl

DLS Manager est décrit dans chapitre 4.

```
dls_ctl 1.4.0-rc2 revision de0a3e76b9ae
Call: dls_ctl [OPTIONS]
        -d [directory]     DLS data directory
        -u [user]          DLS user
        -h                 Show this help
```

D.5. dls_view

```
dls_view 1.4.0-rc2 revision de0a3e76b9ae
Call: dls_view [OPTIONS]
        -d [directory]   DLS data directory
        -h               Show this help
```

D.6. dls

```
  Usage: dls COMMAND [OPTIONS]
  Commands:
      list - List available chunks.
    export - Export collected data.
      help - Print this help.
  Enter "dls COMMAND -h" for command-specific help.
```

D.6.1. dls list

```
  Usage: 1. dls list [OPTIONS]
         2. dls list -j JOB [OPTIONS]

  Description:
         1. Lists all available jobs.
```

```
        2. Lists chunks in the specified job.

  Options:
          -d DIR   Specify DLS data directory.
          -j JOB   Specify job ID.
          -h       Print this help.
```

D.6.2. dls export

```
dls 1.4.0-rc2 revision de0a3e76b9ae
Usage: dls export [OPTIONS]
Options:
   -d DIR          DLS data directory. Default: $DLS_DIR
   -o DIR          Output directory. Default: $DLS_EXPORT_DIR or "."
   -f NAMEFMT      Naming format for export directory.
                   See strftime(3).
                   Default: $DLS_EXPORT_FMT or "dls-export-%Y-%m-%d-%H-%M-%S"
   -a              Enable ASCII exporter
   -m              Enable MATLAB4 exporter
   -j ID           Job to export (MANDATORY)
   -c CHANNELS     Indices of channels to export (see below).
                   Default: All channels
   -p CHANNEL      Path of one channel to export (see
                   below). This option may appear
                   multiple times. Default: All channels.
   -s TIMESTAMP    Start time (see below). Default: Start of recording
   -e TIMESTAMP    End time (see below). Default: End of recording
   -n DECIMATION   Export every n'th value.
   -g              Export messages.
   -l LANGUAGE     2-character language code for messages.
   -q              Be quiet (no progress bar)
   -h              Print this help
CHANNELS is a comma-separated list of channel indices.
   Use the minus sign to specify ranges.
   Examples: "2,4,9", "1-20", "2,4,13-15,42".
CHANNEL is a signal name, optionally prefixed with
   'FILE:', where FILE is the name of the exported
   channel data file. If FILE is empty, or there is no
   colon found, files are named according to the channel
   indices.
TIMESTAMP is a broken-down time with microsecond resolution:
   YYYY[-MM[-DD[-HH[-MM[-SS[-UUUUUU]]]]]] or
   YYYY[-MM[-SS[ HH[:MM[:SS[.UUUUUU]]]]]].
```

```
   Examples: "2006-08", "2005-08-15 13:14:58.896366"
```

D.7. dls_quota

```
Call: dls_quota [OPTIONS]
        -d [directory]     DLS data directory
        -i [seconds]       Verification interval (0 = single check)
        -k                 Not a daemon
        -h                 Show this help
```

Index