

Le Protocole CTCP

par [Mathieu Lemoine](#)

Date de publication : 09/08/2006

Dernière mise à jour : 09/10/2006

Le protocole CTCP a été ajouté au protocole IRC afin de permettre au clients d'interagir. Il permet ainsi d'envoyer d'ajouter certaines fonctionnalités au protocole IRC.

Notes

Remerciements

Introduction

I - Généralités

I-A - Présentation

I-B - Structure

I-C - Le "Low Level Quoting"

I-D - Le "CTCP Level Quoting"

II - Les commandes et messages CTCP

Voir aussi

Notes

Remerciements

Merci à [hansaplast](#) et à [Yogui](#) pour leur relecture.

Introduction

Cet article présente le protocole CTCP (Client-to-Client Protocol) d'une manière simple et, normalement, compréhensible pour n'importe qui ayant un minimum de connaissances sur le protocole IRC.

Au besoin, vous trouverez un lien vers un tutoriel sur le protocole IRC en fin d'article.

Le protocole CTCP a été ajouté au protocole IRC afin de permettre aux clients d'interagir. Il permet ainsi d'ajouter certaines fonctionnalités au protocole IRC.

I - Généralités

I-A - Présentation

Le protocole CTCP est encapsulé dans le protocole IRC et ne le modifie pas, ce qui permet de conserver une compatibilité avec les clients ne l'implémentant pas.

Il met en scène les messages IRC *PRIVMSG* et *NOTICE*.

I-B - Structure

Le protocole CTCP se présente sous forme de messages. Chaque message commence et se termine par un caractère ^A (code ASCII 1).

De fait, les caractères NUL (ASCII 0), ^A, CR (0x0D) et LF (0x0A) sont interdits dans les messages CTCP.

Les messages CTCP se divisent en deux types : requêtes et réponses.

La différence se situe au niveau du message IRC utilisé pour l'encapsulation du message CTCP.

Les requêtes sont encapsulées dans des messages PRIVMSG, tandis que les réponses le sont dans des NOTICE.

La syntaxe est ensuite celle du protocole IRC à peu de choses près :

```
{COMMANDE} [ {SP} {PARAMETRES} ]
```

Le séparateur *SP* est un espace (code ASCII : 0x20).

Les paramètres sont séparés par des *deux-point* (:).

Il peut tout à fait y avoir plusieurs messages CTCP dans un seul message IRC.

I-C - Le "Low Level Quoting"

Même si les caractères NUL, CR et LF sont interdits dans les messages CTCP, il peut être nécessaire de les utiliser dans un message.

Dans ce cas, il faut utiliser le "Low Level Quoting".

Cela consiste simplement en une séquence d'échappement, pour chaque caractère (il y en a quatre en tout) et les séquences d'échappement invalides doivent tout simplement être ignorées et considérées comme si le second caractère était le seul présent.

Le caractère d'échappement est M-QUOTE (0x10).

- M-QUOTE '0' <=> NUL
- M-QUOTE 'r' <=> CR
- M-QUOTE 'n' <=> LF

- M-QUOTE M-QUOTE <=> M-QUOTE

Ainsi, avant d'être envoyé, chaque message devra être échappé s'il contient un ou plusieurs caractères spéciaux.

Cette opération doit être faite par le client et non par l'utilisateur.

I-D - Le "CTCP Level Quoting"

Le LLQ ne permettant pas d'insérer des caractères ^A dans les messages CTCP, un autre type d'échappement a été ajouté.

Il n'y a que deux séquences d'échappement, comme pour le LLQ, toute séquence invalide doit être traitée comme si seul le second caractère était présent.

Le caractère d'échappement se nomme X-QUOTE dans la RFC, il s'agit en fait de l'anti-slash (0x5C)

- X-QUOTE 'a' <=> ^a
- X-QUOTE X-QUOTE <=> X-QUOTE

Ainsi, pendant sa rédaction, chaque message devra être échappé s'il contient un ou plusieurs caractères spéciaux.

Cette opération doit être faite par l'utilisateur et non par le client (comment pourrait-il faire la différence entre un ^A faisant partie d'un message et un ^A servant de délimiteur...).

II - Les commandes et messages CTCP

Comme nous l'avons vu ci-dessus, le protocole CTCP s'organise en commandes. Elles sont au nombre de 11. Mais d'autres peuvent être ajoutées par chaque client...

Nous allons maintenant voir ce qu'elles sont et à quoi elles servent.

- **ACTION**

Ce message CTCP ne prend qu'un paramètre (obligatoire) : le texte de l'action. De plus, il ne peut être utilisé qu'en tant que requête, pas en tant que réponse. Il permet d'indiquer ce qu'on fait...

Exemple :

```
PRIVMSG #ZeChAn :^AACTION :se marre^A
```

Sera affiché par la plupart des clients IRC supportant le CTCP comme : "* Machin se marre", où Machin est le nick du client ayant envoyé le message.

- **DCC**

Les messages CTCP DCC représentent la partie du protocole DCC qui est encapsulée dans le protocole CTCP.

Nous n'en parlerons pas ici, sachez simplement de ces messages existent.

- **SED**

Cette commande n'est pas réellement documentée dans la RFC.

Il s'agit certainement de l'abréviation de "Simple Encryption Decryption".

Tout comme ACTION, elle ne peut être utilisée que dans une requête.

Son seul paramètre est le texte crypté.

Les clients la supportant doivent afficher le texte en clair, cependant l'algorithme de cryptage en lui-même est laissé à la discrétion du client.

- **FINGER**

Cette commande permet d'échanger des informations sur le client (nom réel et login enregistrés dans le client, ainsi que le temps d'idle (d'inactivité)).

La requête ne possède pas de paramètre.

La réponse en possède un unique qui contient les informations demandées, dont la syntaxe est laissé à la discrétion du client.

- **VERSION**

Cette commande permet d'échanger certaines informations sur le client et son environnement.

La requête ne possède pas de paramètre.

La réponse en possède trois : le nom du client, la version du client, l'environnement d'exécution (système d'exploitation, etc.).

- **SOURCE**

Cette commande permet d'échanger des informations sur les points de téléchargement possibles du client.

La requête ne possède pas de paramètre.

La réponse en possède trois : un nom de domaine où l'on peut se connecter de manière anonyme pour récupérer le client, le dossier contenant les fichiers sur ce serveur, la liste des fichiers à télécharger (chaque nom de fichier étant séparé du suivant par un espace)

Il est possible qu'il y ait plusieurs réponses pour une seule requête.

- **USERINFO**

Cette commande permet d'échanger des informations mises à disposition par l'utilisateur (le client ne doit jamais définir ces informations). La requête ne possède pas de paramètre.

La réponse en a un unique, qui est le texte en question.

- **CLIENTINFO**

Cette commande permet d'indiquer quelles commandes du protocole CTCP sont supportées par le client.

La requête possède 0 ou 1 paramètre.

La réponse n'en possède qu'un.

Si la requête ne possède pas de paramètre, il s'agit de la liste des commandes CTCP supportées (séparées par des espaces).

Si la requête a un paramètre, il s'agit de la syntaxe de la commande passée en paramètre.

- **ERRMSG**

Ce message ne peut être utilisé qu'en tant que réponse, il permet de signaler un message d'erreur (qui sera transmis en paramètre de la réponse).

- **PING**

Cette commande permet de trouver le temps nécessaire à un message pour parcourir le réseau d'un client à l'autre.

La requête comme la réponse prennent comme paramètre un timestamp de l'heure à laquelle le message est envoyé.

- **TIME**

Cette commande permet d'échanger l'heure qu'il est pour un autre client.

La requête ne possède pas de paramètre.

La réponse en a un unique, qui est le timestamp sur le client répondant la requête.

Voir aussi

[Le Protocole IRC](#) : Ce qu'est le protocole IRC, comment il fonctionne, etc. Bref, tout ce qu'il est nécessaire de savoir pour comprendre le protocole CTCP, et même plus...

[La RFC du protocole CTCP](#) : Il s'agit en fait d'une "note" de doc, mais c'est la seule norme que j'ai réellement trouvée...