Applications I-Terminal
Spécifications Techniques
Protocoles communs

Abstract

Ce document décrit les messages et constantes communs à toutes les applications i-Terminal à venir dans la future plate-forme INGENICO d'applications internet. Tous les aspects abordés dans ces spécifications techniques ont été conçus en tenant compte des capacités de calcul et les tailles mémoire restreintes d'un terminal de paiement électronique. Ils constituent une base de travail pour toutes les applications utilisant internet comme moyen de communication entre le terminal et un serveur applicatif, qui soit standard, stable et évolutive.

Version

Nom fichier	Titre	Version	Auteur
AIT-ST-PC-1-1-nov01.doc	Applications i-Terminal - Spécifications Techniques – Protocoles communs	1.1 21/11/01	E. PASQUIER

Annule

Nom fichier	Titre	Version	Auteur
TS-01-2000-0001.doc	T-MARKETING - Spécifications Techniques - Le terminal	Toutes	S. GIANGRECO
TM-ProtocoleCom.doc	T-MARKETING - Spécification du protocole de communication	Toutes	G. BOISSIER

SOMMAIRE

Encapsulation des messages : Taille et checksum

Tout message est encapsulé entre deux valeurs 16 bits. L'entête est la taille du message, tandis que les deux derniers octets sont deux sommes de contrôle.
Comme tous les messages commencent par un octet de code, et que ce code est aussi pris en compte dans le calcul des sommes de contrôle, nous avons ainsi une certaine preuve que le message n'a pas été piraté et modifié pendant sa transmission, et que nous pouvons interpréter ce message.

Les sommes de contrôles sont calculées comme suit :
Le premier octet est la somme arithmétique 8 bits de tous les octets du message = $\sum$ (Message[i]) $_{\forall\ i\ \in[0..N-1]}$
Le deuxième octet est le complement à 1 de la parité bit à bit de chaque octet du message = NOT (XOR (Message[i])$_{\forall\ i\ \in[0..N-1]}$)

N_OCTETS_MSG	MESSAGE	CHK_ADD	CHK_XOR
xxxx.xxxx . xxxx.xxxx		xxxx.xxxx	xxxx.xxxx

Dans le but de simplifier les spécifications techniques des protocoles des toutes les i-applications, nous considérons que chaque message, code et données, est encapsulé de cette manière sans que cela soit précisé à chaque description de message. Il est cependant recommandé de spécifier avant toute description de protocole que l'application suit les standards décrits dans ce document – ref AIT-ST-PC-1-1-nov01.doc.

Les messages définissant le protocole de dialogue entre le terminal et le serveur sont TOUS précédés par un octet de code. Les deux bits de poids fort de cet octet spécifie le type du message, tandis que les 6 bits de poids faible codent la classe du message (tous les messages d'une seule classe sont les parties d'un même sous-dialogue).
Types de message :
00 : Requête
10 : ACK
01 : NACK
11 : Erreur d'exécution (ERROR)

Certains messages ont la même structure de données qu'ils soient émis par le terminal ou par le serveur, mais ne revêtent pas exactement la même signification suivant le sens de l'émission.
La description des messages commence donc par préciser le sens d'émission/réception du message et la signification du message dans ce sens, sous la forme « Source ➔ Destination : signification ». Il peut y avoir deux descriptions différentes, une pour chaque sens d'émission.

Messages standard

Connection

Terminal ➔ Serveur : demande d'ouverture de session. Mode = 1 si connexion automatique, 0 si manuelle. Mode = 3 si connexion automatique forcée pour éviter les problèmes de stockage de l'historique des transactions.

MES_COD	EPOS_ID	MODE
0000.0001	xxxx.xxxx . xxxx.xxxx . xxxx.xxxx . xxxx.xxxx	0000.00xx

Authentication

Serveur ➔ Terminal : Challenge avec une valeur aléatoire
Terminal ➔ Serveur : Réponse au challenge

MES_COD	AUTHENTICATION_VALUE
0000.0010	xxxx.xxxx . xxxx.xxxx . xxxx.xxxx . xxxx.xxxx

Authentication Acknowledgement

Acknowledged

Serveur ➔ Terminal : Session acceptée par le serveur

MES_COD
1000.0010

Not Acknowledged

Serveur ➔ Terminal : Session refusée par le serveur

MES_COD	ERROR
0100.0010	xxxx.xxxx

EPOS Program Version

Terminal ➔ Serveur : Version actuelle du programme sur le Terminal
Serveur ➔ Terminal : Version du dernier programme mis à disposition sur un serveur de téléchargement

MES_COD	MAJOR	MINOR
0000.0011	xxxx.xxxx	xxxx.xxxx

End of Session

Terminal ➔ Serveur : Fermeture de la session

MES_COD
0011.1111

Codes d'erreurs

Les erreurs ayant des codes supérieurs ou égal à 0x80 (bit de poids fort marqué à 1) sont irrécupérables et conduisent à la fermeture de la session des deux côtés.

Erreurs de connexion

Erreurs dues aux couches physique, liaison et réseau de la communication (Modem et Pile IP). Ce sont des erreurs irrécupérables.

CODE	Hexa	Signification
1111.1111	F.F	Initialisation Port Communication échouée

Signification : Le port COM du terminal n'a pas pu être initialisé
Causes possibles : Problème physique sur le terminal

1111.1110	F.E	Numérotation échouée

Signification : Impossible de composer le numéro de téléphone du provider
Causes possibles : Terminal non relié à la prise de téléphone ; Numéro du provider invalide

1111.1101	F.D	Connexion Provider échouée

Signification : Impossible de se connecter au provider
Causes possibles : Le numéro de téléphone ne correspond pas à un modem d'un provider ; le protocole de connexion a échoué (Time-Out)

1111.1100	F.C	Ouverture de session échouée

Signification : Impossible de s'authentifier auprès du provider
Causes possibles : Login et / ou mot de passe du compte invalide

1111.1011	F.B	Connexion au serveur échouée

Signification : Impossible de se connecter au server
Causes possibles : Le serveur est inaccessible.

1111.1010	F.A	Déconnexion serveur échouée

Signification : La déconnexion au serveur n'a pas été effectuée proprement
Causes possibles : ??

1111.1001	F.9	Envoi du message échoué

Signification : Le message n'a pas été envoyé correctement
Cause : Renvoyé si net_tx a échoué. Probablement, la connexion avec le provider ou le serveur a été rompue.

1111.1000	F.8	Réception du message échoué

Signification : Le message n'a pas été reçu correctement
Cause : Renvoyé si net_rx a échoué. La connexion avec le provider ou le serveur a été rompue.

Erreurs protocole générales (NACK)

Ces erreurs irrécupérables précisent la raison du rejet d'un message. Elles correspondent aux couches Transport et Session (TCP/IP ne gérant qu'une partie de la couche transport).

CODE	Hexa	Signification	Classe de message
1110.1111	E.F	Time Out	Toutes

Signification : Données attendues non reçues dans le temps imparti.
Causes possibles : une coupure réseau ; un paquet IP perdu et non réexpédié à temps, empêchant la reconstruction par TCP du flux de données ; des données envoyées d'une taille inférieure à ce qui était attendu.

1110.1110	E.E	Erreur checksum	Toutes

Signification : Checksum calculé différent du checksum envoyé en fin de message.
Causes possibles : Préfixe de taille incorrect, d'ou checksum lu au mauvais endroit du flux ; modification d'un ou plusieurs octets du flux.
Un double checksum calculé par addition et NOT(XOR) permet une détection des erreurs beaucoup plus efficace. En effet, pour qu'une modification ne soit pas détectée, elle doit porter sur plusieurs octets afin que les erreurs se compensent mutuellement. Seulement, il faut qu'elle se compensent selon 2 modes de vérification radicalement opposées : la modification d'un bit peut être compensée par la méthode XOR en modifiant le même bit sur un autre octet. La modification d'un bit peut être compensée par la méthode ADD en ajoutant à un autre octet la négation + 1de la valeur du bit modifié. Or, les deux modifications compensatrices A et (\A +1) ne sont égales que si A = 0 (soit aucune modification) et A=80h (seul le bit de poids le plus fort est changé). Les possibilités sont donc très limitées pour un éventuel hackeur, de modifier un message sans que l'on s'en aperçoive (a moins de savoir quand, où et comment se calcule le checksum).

1110.1101	E.D	Message inconnu	Toutes

Signification : Le code de message correspond à un type non reconnu par le receveur du message
Causes possibles : L'émetteur et le receveur ne correspondent pas à des versions compatibles : certaines fonctions ne sont pas encore implémentées ; l'émetteur n'est pas habilité à dialoguer avec le serveur (application pirate…)

1110.1100	E.C	Message non attendu	Toutes

Signification : Un message a été reçu correctement, mais son code ne correspond pas à un message valide dans la continuité de la session.
Causes possibles : Message Time-Out renvoyé par l'émetteur alors que le récepteur pensait avoir bien envoyé le message précédent

1110.1011	E.B	Message taille incorrecte	Toutes

Signification : La taille totale du message ne correspond pas à la taille attendue pour les messages de ce type. Dans le cas des commandes, cette erreur ne survient que si le passage des paramètres n'est pas valide, mais pas si les paramètres ne sont pas compatibles avec la commande.
Causes possibles : L'émetteur et le receveur ne correspondent pas à des versions compatibles : certains paramètres diffèrent ; l'émetteur n'est pas habilité à dialoguer avec le serveur (application pirate qui a déjà eu de la chance d'arriver là…)

1110.1010	E.A	Message Erreur Mémoire	Toutes

Signification : La mémoire temporaire ne permets pas le stockage du message dans sa totalité.
Causes possibles : Le message est trop gros et ne peut pas être traité en plusieurs fois.

1110.0000	E.0	Demande rejetée	2 (Autenticate)

Signification : La demande précédente, bien que formulée correctement, a été rejetée.
Cause : Authentification échouée

Erreurs Applicatives (ERROR)

Ces erreurs précisent la raison du non-traitement d'une requête valide. Les codes sont tous inférieurs à 0x80 (bit de poids fort marqué à 0). Les codes sont librement définissables par chaque application.

Librairie IPUtils – Manuel d'utilisation

Cette librairie définit les structures de données, constantes et procédures permettant de mettre en œuvre les concepts spécifiés dans ce document.

Elle nécessite pour fonctionner une librairie TCP/IP suivant les déclarations de prototypes de fonctions décrits en annexes.

Mécanisme de Callback pour les gros messages

Pour la gestion des messages longs (en réception), c'est à dire plus gros que la taille du buffer de réception désigné lors de l'initialisation de cette librairie, un mécanisme de callback d'une fonction permet à l'application de gérer partiellement le message reçu, pour permettre à la librairie de continuer la réception des données sans problèmes.

Attention: Les sommes de contrôle en fin de message n'ont pas encore été validées, de même que la taille du message qui n'est pas encore totalement reçu. Le traitement partiel du message doit donc pouvoir être annulé si le message devait s'avérer invalide.

Valeurs de retour :

PR_NOERR : si aucune erreur n'a été rencontrée lors du traitement partiel du message
PR_APPERR_VALUERANGE : si une valeur reçue dans le message est invalide
PR_APPERR_MEMORY : si la mémoire est insuffisante pour le traitement partiel du message
... d'autres codes d'erreur peuvent éventuellement être retournés par la fonction de callback.

Prototype d'une fonction de callback :

```
typedef char (*CallBackFct)(void);
```

Structures de données

Port d'écoute serveur

```
typedef struct
{
	UCHAR8	szLocation[16];  // Nom du service ou du serveur
	UCHAR8	IP[4];
	UINT16	Port;
} SERVER_SOCKET;
```

Décrit le port d'écoute (Socket) du serveur auquel il est possible de se connecter via TCP/IP.

Fournisseur d'accès

```
typedef struct
{
	CHAR8	szName[16];
	CHAR8	szPhoneNumber[21];
	CHAR8	szUserName[21];
	CHAR8	szPassword[21];
} PROVIDER;
```

Décrit un fournisseur d'accès et les paramètres pour ouvrir une session internet.

Paramètres de connexion

```
typedef struct
{
        UCHAR8      ucNbRetry;
        UINT16      uiTimeWait;
        UINT16      uiTimeoutFC;
        UINT16      uiTimeoutC;
} CONNECTPARAMS;
```

Décrit les paramètres généraux de dialogue via la pile IP.

Gestionnaire de message

```
typedef struct
{
        short Size;
        char Code;
        char BufORStat; // Statut de Buffer OverRun
} MESSAGEHANDLE;
```

Une structure particulière est utilisée dans la gestion des messages. Elle permet de stocker la taille du message suivant le code, son code et son statut de dépassement de capacité (pour la gestion des long messages – cf Mécanisme de CallBack).
Statut de dépassement de capacité de buffer
= 0 si aucun dépassement n'a eu lieu lors de la réception de message
= N si N dépassement ont eu lieu (avec appel de la fonction de callback)

Variables globales

Cette librairie ne peut gérer pour l'instant qu'une seule connexion à un serveur à la fois. Elle déclare en accès global une variable de type MESSAGEHANDLE :

```
MESSAGEHANDLE _MesHandle;
```

Via cette variable il est possible d'accéder aux informations concernant le dernier message reçu.

Procédures et fonctions

InitIPUtils

```
void InitIPUtils(CONNECTPARAMS *CP, char *Buffer, short MaxBuf, CallBackFct BufferOverrunFct);
```

Description :
Initialise les paramètres par défaut pour la pile IP, le buffer de réception (taille maximum et adresse) ainsi que l'adresse de la fonction de CallBack en cas de dépassement du buffer.

Paramètres :
- CP : Paramètres de connexion et de dialogue utilisés par la pile
- Buffer : Adresse du buffer de réception, dans la zone de mémoire de travail (XDATA)
- MaxBuf : Taille du buffer
- BufferOverrunFct : Fonction de CallBack

ReceiveMessage

char ReceiveMessage(char CodeANDMask, char Code, short Size);

Description :

Reçoit un message, le décapsule (valide), renseigne le handle de message et stocke les données après le code dans le buffer de réception

Paramètres :

- CodeANDMask : Masque AND binaire de validation du code attendu
- Code : Code de message attendu
- Size : Taille minimum du message attendu, 0 si imprévisible

Valeur de retour :

- PR_NOERR : Aucune erreur à la réception, un message parfaitement valide attends dans le buffer
- PR_GENERR_TIMEOUT : TimeOut
- PR_GENERR_CHECKSUM : Checksum error
- PR_GENERR_MESMEMORY : Buffer overrun et aucune fonction de CallBack spécifiée
- PR_GENERR_MESUNWAITED : Message non attendu (Code invalidé)
- PR_GENERR_MESBADSIZE : Message de taille incorrecte
- Toute autre code d'erreur renvoyé par la fonction de CallBack

Détails :

La validation du code message attendu se fait par la fonction logique suivante :

Validation = (CodeReçu AND Mask = CodeAttendu AND Mask)
ou encore (CodeReçu XOR CodeAttendu) AND Mask = 0

si le résultat de la fonction est PR_NOERR, _MesHandle est modifié afin de renseigner le code et la taille du message reçu. Dans les autres cas, _MesHandle peut ne pas être valide.

SendMessage

char SendMessage(char Code, short Size, char *Buffer);

Description :

Envoie un message encapsulé.

Paramètres :

- Code : Code du message
- Size : Taille du message code exclus
- Buffer : Adresse du buffer contenant le message a envoyer après le code

Valeur de retour :

- PR_CONERR_SEND : Une erreur s'est produite lors de l'envoi du message
- PR_NOERR : aucune erreur, message correctement envoyé au provider

ConnectProvider
char ConnectProvider(PROVIDER *Provider);

Description :
Se connecte au provider grâce aux paramètres dans la structure PROVIDER

Valeur de retour :
- PR_NOERR : Aucune erreur, la connexion au provider est établie et la session TCP/IP ouverte
- PR_CONERR_INITCOM : Problème lors de l'initialisation du port COM
- PR_CONERR_DIAL : Problème lors de l'appel du provider
- PR_CONERR_PROVIDER : Problème de connexion au provider
- PR_CONERR_LOGIN : Problème d'identification à l'ouverture de la session

ConnectServer
char ConnectServer(SERVER_SOCKET *Server);

Description :
Se connecte au serveur grâce aux paramètres dans la structure SERVER_SOCKET

Valeur de retour :
- PR_NOERR : Aucune erreur, la connexion TCP/IP au serveur est établie
- PR_CONERR_SERVER : Connexion au serveur impossible

DisconnectProvider
char DisconnectProvider(void);

Description :
Se déconnecte du provider, fermant ainsi la session TCP/IP et toutes les communication IP en cours.

Valeur de retour :
- PR_NOERR : Aucune erreur, la connexion au provider est coupée

DisconnectServer
char DisconnectServer(void);

Description :
Se déconnecte du serveur, fermant ainsi la session TCP/IP et toutes les communication IP en cours.

Valeur de retour :
- PR_NOERR : Aucune erreur, la session avec le serveur est fermée
- PR_CONERR_CLOSESERVER : Il y a eu un problème lors de la déconnexion avec le serveur

Annexes

Librairies nécessaires

La librairie IPUtils n'implémente qu'une partie des couches ISO Transport et Session. Elle s'appuie donc sur une librairie TCP/IP (couche réseau et une partie du transport) et sur l'OS du terminal (couche physique et liaison).

Dans la version actuelle de IPUtils, la librairie TCP/IP utilisée doit déclarer les structures de données et prototypes de fonctions décrites ci-dessous. Il est possible facilement de recompiler une version de la librairie IPUtils s'appuyant sur d'autres librairies de communication, du fait de son architecture interne en couche abstraite indépendante des couches réseau utilisées. Dans une version future, il pourrait même être envisagé de modifier cette bibliothèque pour qu'elle puisse utiliser à la demande (run-time) d'autres couches de communication que TCP/IP (basées sur une communication RS-232 directe par exemple).

Déclaration de structure de données

```
typedef struct netparam
{
        uchar IP_add[4];
        uchar net_type;
        COM_PARAM *eft_com_param;
        DIAL_PARAM *eft_dial_param;
}NET_PARAM;

typedef struct login_info_param
{
   uchar login_length;
   uchar *login;
   uchar pass_length;
   uchar *pass;
} LOGIN_INFO_PARAM;
```

Prototypes des fonctions

```
uchar   net_connect(NET_PARAM *net_con_param);
uchar   net_login_info(LOGIN_INFO_PARAM *lparam);
uchar   net_disconnect(void);
uint    net_open(uchar *IP_add, uint port_num);
uchar   net_close(uchar net_id);
uchar   net_tx(uchar net_id, uchar *msg, uint len);
uchar   net_rx(uchar net_id, uchar *msg, uint max, uint timeout, uint *len);
uchar   net_clr_rx(uchar net_id);
```

Révisions du document

Version 1.0	Commencé le : 07/11/2001	Fini le : 8/11/2001

- Mécanisme d'encapsulation des messages (Couche Transport ISO)
- Messages standard
- Messages d'erreur NACK (Session) et ERROR (Application)

Version 1.1	Commencé le : 20/11/2001	Fini le : 21/11/2001

- Messages d'erreur liaison/réseau
- Manuel d'utilisation de la librairie IPUtils
- Annexe : Librairie liaison