

Guide d'intégration



DINKEYPRO

Protection de logiciels LABVIEW avec DinkeyPRO/FD



Contact Commercial :

Tél. : 02 47 35 70 35

Email : com@aplika.fr

Contact Technique :

Tél. : 02 47 35 53 36

Email : support@aplika.fr

Version 1.20 du 24/02/2010



aplika

La Follière - 37270 AZAY SUR CHER
Tél. 33(0)2 47 35 70 35 - Fax 33(0)2 47 35 70 25 - e-mail : aplika@aplika.fr

Guide d'intégration DinkeyPRO/FD LABVIEW



1. Introduction	3
2. Le module de protection Dinkey pour LABVIEW	3
2.1. Module dpwin32.vi	3
3. Exemple d'appel de la protection.....	6
3.1. Intégration du code à votre application – Test simple de la protection.....	6
3.2. Protection du module DPWIN32.DLL	9
4. Déploiement de votre application	11

Guide d'intégration DinkeyPRO/FD LABVIEW

1. Introduction

Afin de vous aider dans les premiers pas de la pose de protection sur vos logiciels LABVIEW, vous trouverez dans ce guide de prise en main rapide l'essentiel des commandes et fonctionnalités pour la bonne utilisation des clés DinkeyPRO/FD.

Méthode utilisée : API avec appel de dpwin32.vi.

Le code LABVIEW de cet exemple a été testé avec la version 8.5 et 2009 de LABVIEW pour Windows. Pour l'utilisation sous d'autres versions et/ou d'autres plateformes, nous vous conseillons d'effectuer un test systématiquement car nous ne pouvons garantir une compatibilité. En cas de difficulté nous contacter en prenant soin de nous communiquer la version de LABVIEW et le système d'exploitation utilisé.

Le fichier sample.vi contient un exemple de code permettant l'appel à la protection Dinkey. Cet exemple a pour objectif de mettre en évidence le fonctionnement des APIs Dinkey afin de vous permettre une intégration facile dans vos projets LABVIEW.

ATTENTION :

Si vous distribuez du code LABVIEW sous forme de module LABVIEW réutilisable à partir d'autres fichiers LABVIEW de type .vi, n'oubliez pas de retirer le block diagramme VI afin que les utilisateurs ne puissent pas modifier votre code et ainsi retirer la protection. Pour plus d'informations à ce sujet. Consultez la documentation LABVIEW.

2. Le module de protection Dinkey pour LABVIEW

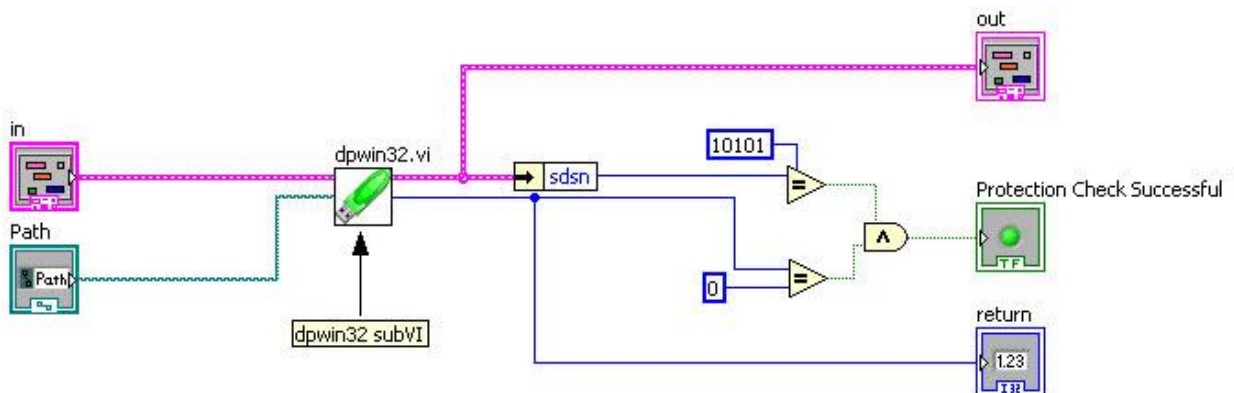
2.1. Module dpwin32.vi

Le test de protection LABVIEW est effectué en faisant appel à l'instrument virtuel dpwin32.vi. Ce dernier permet d'établir le dialogue entre la clé Dinkey et votre programme via l'utilisation de dpwin32.dll.

Vous devrez donc protéger cette DLL en utilisant la méthode de protection API comme décrit en section 3.2.

Le chemin de dpwin32.dll doit être indiqué comme une entrée de VI. Si vous n'indiquez pas de chemin à LabVIEW. Il ira alors par défaut chercher dpwin32.dll dans le répertoire système, couramment C:\Windows\system32.

Le diagramme de test pourra être par exemple :



Sur le diagramme on voit qu'en plus du chemin de dpwin32.vi, VI nécessite 1 cluster d'entrée et fournis 2 sorties (1 cluster – 1 entier 'integer')
 Comme décrit ci-après :

INPUTS	OUTPUTS	return
'in' cluster: function flags execs_dec rw_offset rw_length rw_data alt_prog_name var_a var_b var_c var_d var_e var_f var_g var_h alg_number	'out' cluster: ret_code ext_err type model sdsn prodcode dongle_number update_number data_area_size max_alg_num execs exp_day exp_month exp_year features net_users alg_answer fd_capacity fd_drive	'return' integer: Correspond à la valeur de retour De dpwin32.dll identique au ret_code de cette dll. Pour l'interprétation des valeurs se reporter à la documentation.

A l'exception de rw_data les noms et les types de données de variables utilisées sont en correspondance avec ceux de la structure DRIS, utilisée avec des langages de type Texte.

Les type et taille de ces variables sont décrit dans le manuel DinkeyPRO, chapitre STRUCTURE DRIS. En voici un extrait :

Champ	Taille	Type	Entrée/ Sortie	Description
header	4	char	Entrée	Doit être initialisée avec la valeur 'DRIS'.
size	4	int	Entrée	Taille de la structure DRIS.
seed1	4	int	Entrée	Clé d'encodage aléatoire utilisée pour le cryptage DRIS et pour le cryptage des données.
seed2	4	int	Entrée	Autre clé d'encodage (Voir description ci-dessus).
function	4	int	Entrée	Permet, en méthode API, d'indiquer la fonction que vous souhaitez utiliser lors de l'appel de DDProtCheck. Attention à n'utiliser qu'une seule fonction à la fois. Pour plus d'infos sur la liste des fonctions, voir la rubrique Fonctions
flags	4	int	Entrée	Le champ Flag vous permet d'indiquer des options pour chacune des fonctions. Vous pouvez utiliser une ou plusieurs valeurs en les associant avec OR. Pour la liste, consulter la section 'Flags'.
execs_dec	4	uint	Entrée	Décrémente le champ "exécution" de la valeur indiquée en entrée. La valeur doit être positive.
data_crypt_key_num	4	int	Entrée	Indique quelle sera la clé de cryptage/décryptage utilisée pour le traitement des données sécurisées.
rw_offset	4	int	Entrée	Offset de lecture /écriture de la zone de données (positionnement dans la zone).
rw_length	4	int	Entrée	Longueur de la zone de données à lire ou à écrire.
rw_data_ptr	4/8	Pointer*	Entrée	Adresse de la zone de données à lire ou à écrire.
alt_prog_name	256	Char	Entrée	Permet d'indiquer à la fonction DDProtcheck en méthode API un nom de programme appelant différent de celui du programme réellement utilisé pour l'appel de cette fonction. Doit être terminé par une chaîne ASCII Null.

var_a	4	int	Entrée	Variable à passer à l'algorithme.
var_b	4	int	Entrée	Variable à passer à l'algorithme.
var_c	4	int	Entrée	Variable à passer à l'algorithme.
var_d	4	int	Entrée	Variable à passer à l'algorithme.
var_e	4	int	Entrée	Variable à passer à l'algorithme.
var_f	4	int	Entrée	Variable à passer à l'algorithme.
var_g	4	int	Entrée	Variable à passer à l'algorithme.
var_h	4	int	Entrée	Variable à passer à l'algorithme.
alg_number	4	int	Entrée	Indique le numéro de l'algorithme utilisateur à exécuter.
ret_code	4	int	Sortie	Code d'erreur.
ext_err	4	int	Sortie	Code d'erreur étendu.
type	4	int	Sortie	Type de Dongle Dinkey (1 = Pro, 2 = FD)
model	4	int	Sortie	Modèle de Dongle (1 = Lite, 2 = Plus, 3 = Net 1, 4 = Net 5, 5 = Net 10, 6 = Net 50, 7 = Net illimité).
sdsn	4	int	Sortie	Numéro de série développeur (SDSN).
prodcode	12	char	Sortie	Code produit (Terminé par Null).
dongle_number	4	uint	Sortie	Numéro de série du dongle.
update_number	4	int	Sortie	Prochain numéro de mise à jour (Démarre à 1).
data_area_size	4	uint	Sortie	Taille allouée à la zone de données sécurisées (en octets).
max_alg_num	4	int	Sortie	Plus grand numéro d'algorithme existant.
execs	4	int	Sortie	Exécutions restantes.
exp_day	4	int	Sortie	Jour d'expiration (nombre).
exp_month	4	int	Sortie	Mois d'expiration (nombre).
exp_year	4	int	Sortie	Année d'expiration (nombre).
features	4	uint	Sortie	Mot de fonction.
net_users	4	int	Sortie	Nombre maximum d'utilisateurs réseaux simultanés.
alg_answer	4	int	Sortie	Résultat de l'exécution de votre algorithme en fonction de l'algorithme utilisé et des variables d'entrée du DRIS dans var_a...var_h.
fd_capacity	4	uint	Sortie	Capacité en octets de la zone de données Flash Disk du dongle détecté. Seulement disponible pour les clés de type FD. La valeur est actuellement de 1Go mais pourra être complétée par des capacités supérieures dans le futur.
fd_drive	4	Char	Sortie	Unité locale attribuée à l'espace Flash Disk du dongle exemple : f:\

*4 octets pour les codes 32 bits et 8 octets pour les codes 64 bits

NB si la valeur "no limit" ou "illimité" est utilisée, la valeur mémorisée est -1.

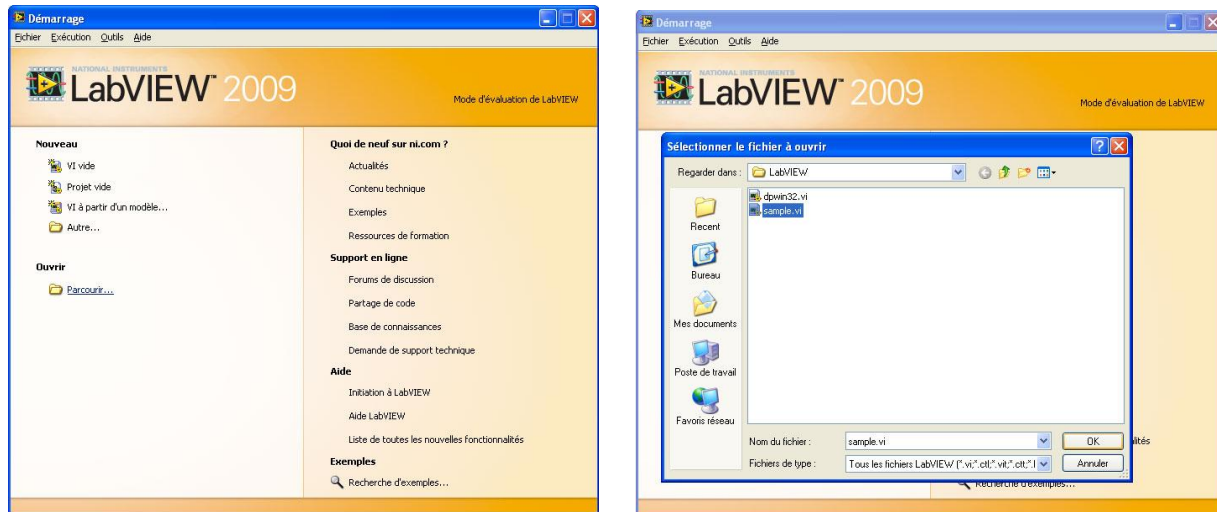
rw_data est un tableau d'octets utilisé pour la transmission des données à lire/écrire dans la zone de données sécurisée de la clé. (Modèles Net et Plus uniquement).

Inclus dans dpwin32 VI, ce tableau est converti en cluster. LabVIEW limitant le nombre d'éléments d'un cluster à 256, la taille maximum des données pouvant être lues ou écrites sera de 256 octets par lecture ou écriture. Pour des données plus grandes il faudra décomposer en plusieurs appels ces phases d'écriture ou de lecture.

3. Exemple d'appel de la protection

3.1. Intégration du code à votre application – Test simple de la protection

- Ouvrez l'éditeur LABVIEW.
- Puis ouvrir le fichier sample.vi qui contient un exemple simple d'appel à la protection via l'utilisation de dpwin32 VI.



- Notre exemple fait appel à la DLL et effectue alors une vérification de la protection. Un code 0 est retourné si la clé est présente et si le numéro SDSN de la clé est 10101 (code SDSN pour les clés de démo). Vous devrez alors adapter le numéro SDSN au votre, il vous sera attribué lors de votre première commande de clés.
- Attention à bien indiquer une valeur valide dans le champ fonction avant test de la protection. (les valeurs acceptées vont de 1 à 7 – 1 correspond à Protection_Check test simple de la protection). Pour les autres codes ce reportez au tableau ci-dessous :

PROTECTION_CHECK = 1 // Test de protection, test des paramètres programmes

EXECUTE_ALGORITHM = 2 // Test de protection + calcul de la réponse selon l'algorithme choisi et mes variables indiquées en entrée

WRITE_DATA_AREA = 3 // Test de protection + Ecriture dans la zone de données du dongle

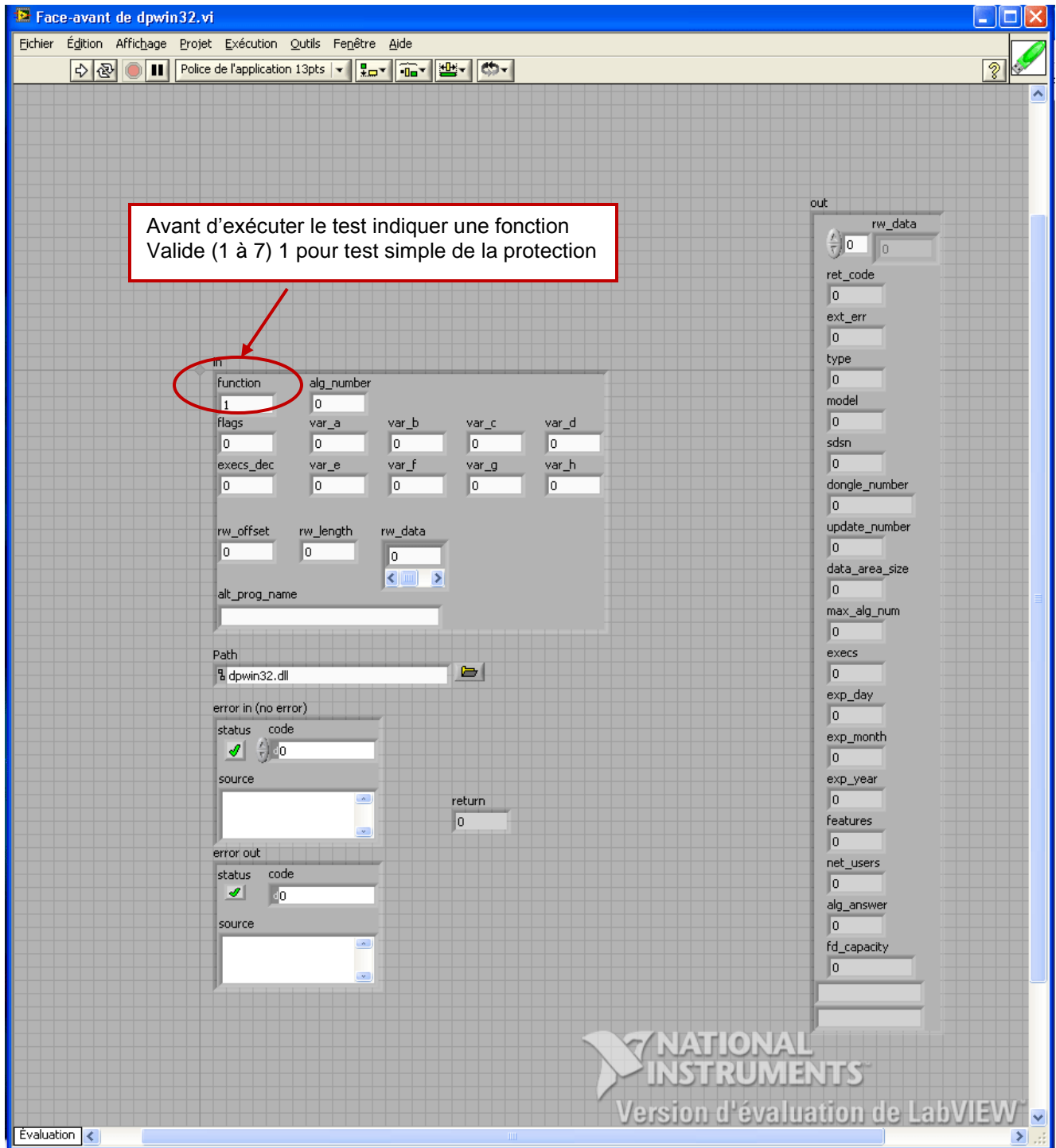
READ_DATA_AREA = 4 // Test de protection + Lecture de la zone de données du dongle

ENCRYPT_USER_DATA = 5 // Test de protection + Cryptage des données utilisateurs

DECRYPT_USER_DATA = 6 // Test de protection + Décryptage des données utilisateurs

FAST_PRESENCE_CHECK = 7 // Test de protection rapide de la clé. Sécurité réduite, pas d'options possibles

L'écran exemple apparaît



Si le code de retour est 0 et le numéro SDSN est valide. L'écran ci-dessous montre le résultat d'un test réalisé avec succès :

The screenshot displays a software interface with a grid background. It is divided into several sections:

- in:** A large input area containing fields for 'function' (1), 'alg_number' (0), 'flags' (0), 'execs_dec' (0), 'rw_offset' (0), 'rw_length' (0), 'rw_data' (0), and 'alt_prog_name' (empty). It also includes a 'Path' field with 'dpwin32.dll' and a folder icon.
- error in (no error):** A section with 'status' (green checkmark), 'code' (0), and 'source' (empty).
- error out:** A section with 'status' (green checkmark), 'code' (0), and 'source' (empty).
- return:** A single field containing the value 0.
- out:** A vertical list of output fields: 'rw_data' (0), 'ret_code' (0), 'ext_err' (0), 'type' (2), 'model' (2), 'sdsn' (10101), 'dongle_number' (2231755860), 'update_number' (1), 'data_area_size' (0), 'max_alg_num' (0), 'execs' (-1), 'exp_day' (-1), 'exp_month' (-1), 'exp_year' (-1), 'features' (0), 'net_users' (0), 'alg_answer' (0), 'fd_capacity' (7749), 'F:\', and 'DEMO'.

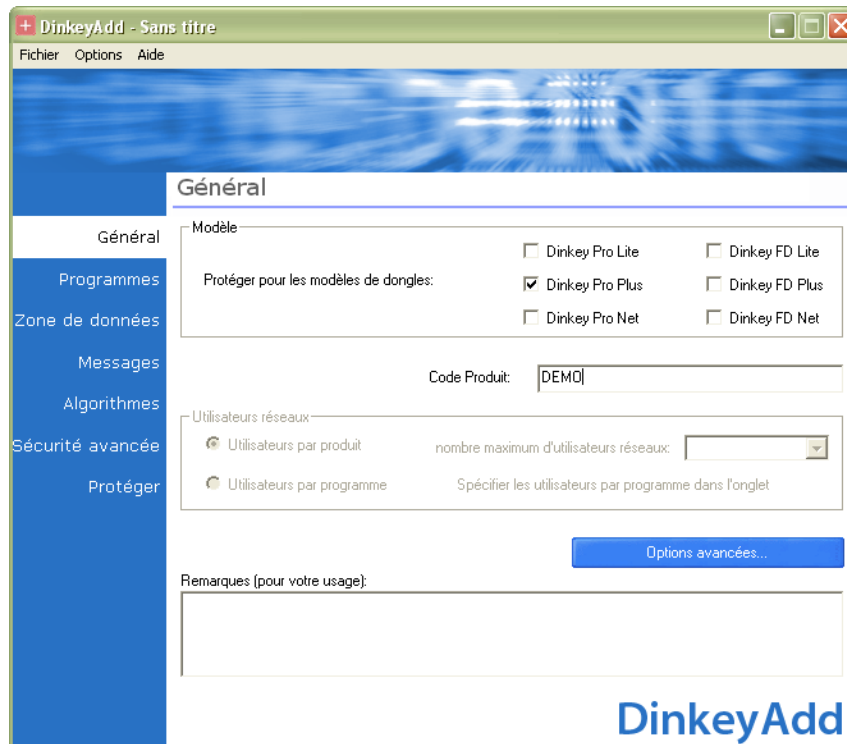
- Nous avons également placé des contrôles sur les variables INPUTS et OUTPUT, ainsi que sur le code de retour (ret_code). Cela afin de vous permettre une prise en main de la protection.
- Nous vous conseillons fortement de mettre en place les fonctions de cryptage et d'algorithme de la clé afin d'obtenir un niveau de sécurité optimal. Pour cela nous vous invitons à consulter le chapitre 'Comprendre la protection de logiciel et les méthodes de piratage'.

3.2. Protection du module DPWIN32.DLL

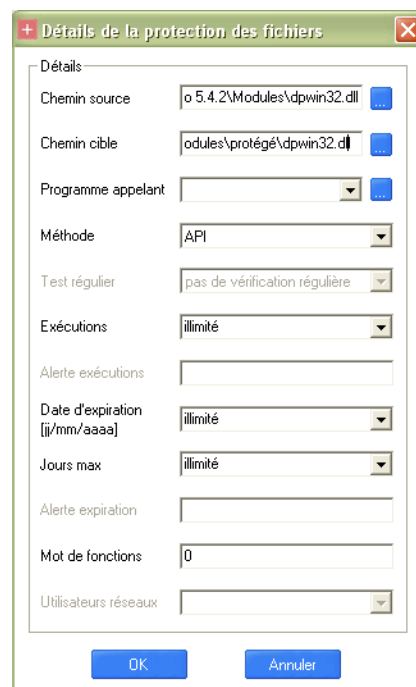
Ouvrez l'utilitaire DinkeyAdd.

Dans l'onglet "Général" :

- Précisez le type de clé que vous utilisez. En cas de doute, exécutez DinkeyLook afin d'afficher un diagnostic de la clé.
- Indiquez le Code Produit (pour les clés d'évaluation, celui-ci est "DEMO").

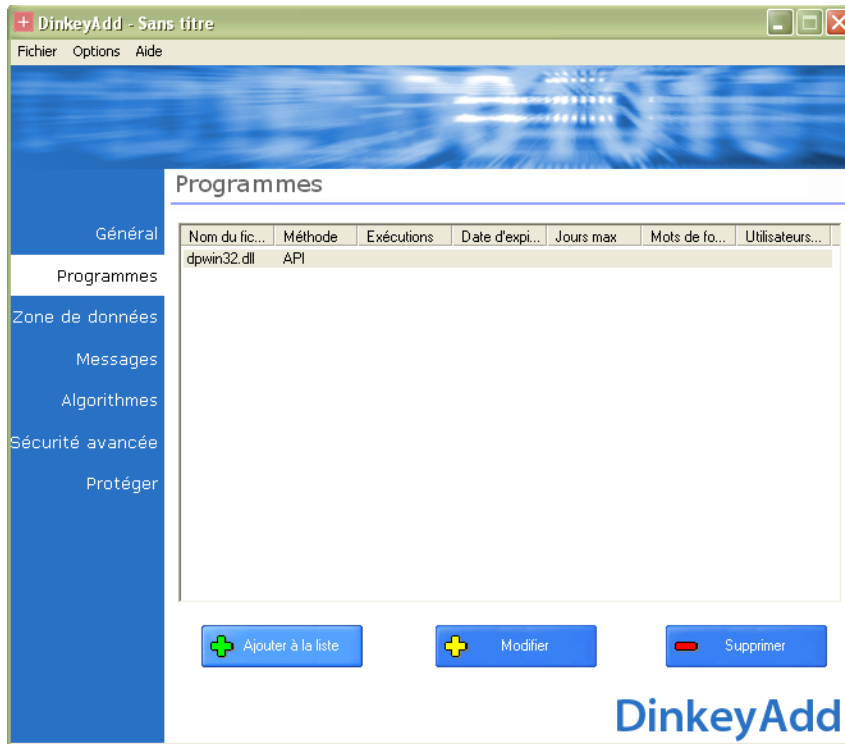


Dans l'onglet "Programmes" :

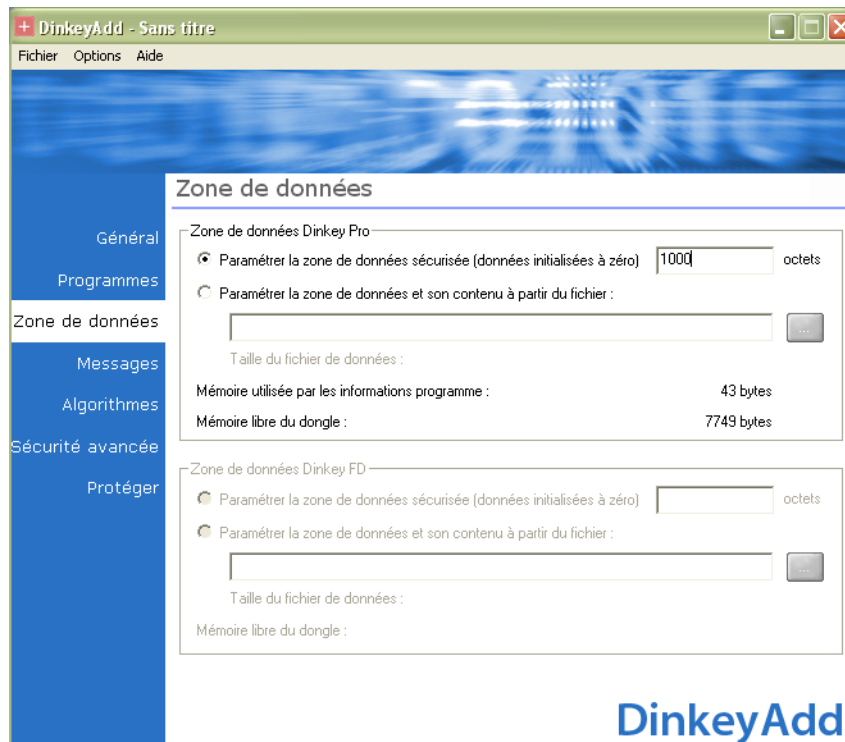


- Cliquez sur le bouton "Ajouter à la liste".
- Dans le champ "Chemin source", pointez sur le fichier dpwin32.dll (celui-ci se trouve par défaut dans le sous-dossier "Modules" du dossier d'installation de DinkeyPRO).
- Dans le champ "Chemin cible", indiquez l'emplacement vers lequel vous souhaitez qu'une copie protégée de dpwin32.dll soit placée.
- Vérifiez que la méthode "API" est bien sélectionnée.
- Laissez les valeurs par défaut pour les champs suivants.
- Validez en cliquant sur le bouton OK.

Le fichier dpwin32.dll est ajouté à la liste des programmes.



Dans l'onglet "Zone de données" :

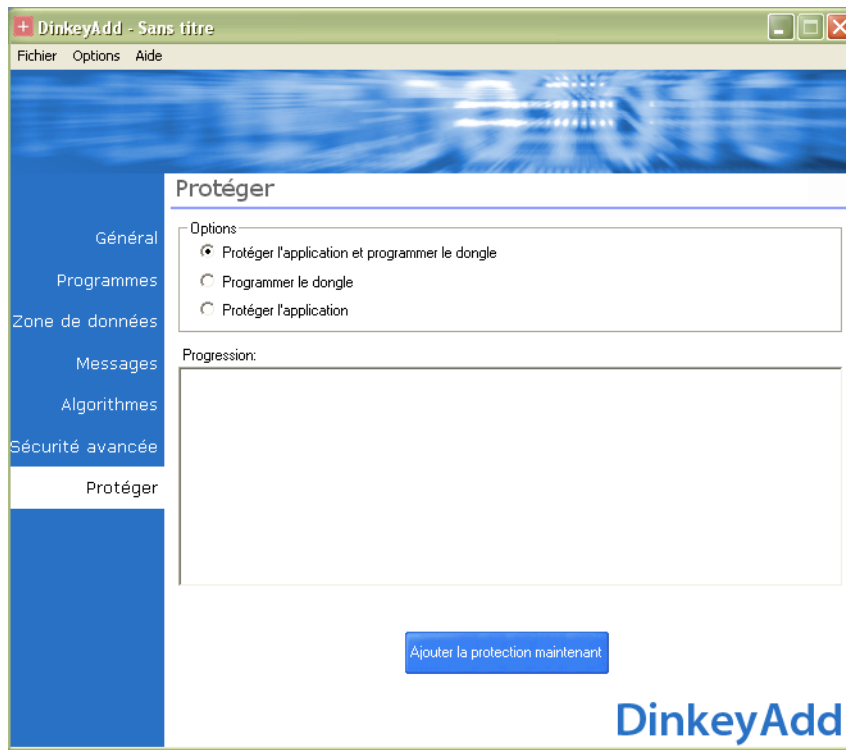


- Initialisez la taille de la zone de données sécurisée, par exemple 1000 octets. Le code exemple fourni permet de tester la lecture/écriture à vers/à partir de la zone de données sécurisée. Ceci ne fonctionnera que si cette zone est initialisée avec une taille suffisante pour recevoir les données de test.

Remarques :

- Selon le type de clé (DinkeyPRO ou DinkeyFD) choisi dans l'onglet "Général", les paramètres de la zone de données correspondante seront activés dans l'onglet "Zone de données".
- Seules les versions Plus et Net disposent d'une zone de données sécurisées.

Dans l'onglet "Protéger" :



- Vérifiez que l'option "Protéger l'application et programmer le dongle" est sélectionnée.
- Assurez-vous que le dongle est connecté.
- Cliquez sur le bouton "Ajouter la protection maintenant"
- Un message de confirmation apparaît.

Vous disposerez alors d'un dongle correctement programmé, ainsi que d'une version protégée de dpwin32.dll dans le dossier vers lequel vous aurez pointé dans le champ "Chemin cible" de la fenêtre "Détails de la protection des fichiers".

4. Déploiement de votre application

Vous devrez fournir le fichier '*dpwin32.dll*' avec votre logiciel. Ce fichier doit être situé dans le même dossier que votre application. Vous pouvez également le placer dans le dossier '*Windows\System32*' (ou '*Windows\System*' pour Win95/98/Me).

Si '*dpwin32.dll*' n'est pas trouvé, une erreur surviendra.



aplika

La Foltière - 37270 AZAY/CHER
Tél. 02 47 35 70 35 - Fax 02 47 35 70 25
e-mail : aplika@aplika.fr
www.aplika.fr