TIL TECHNOLOGIES

GUIDE de paramétrage d'une applet

01 Août 2017 version 1.6

TABLE DES MATIÈRES

Chapitre 1. Glossaire - Applet

Applet	Script qui est exécuté par le MLP2 à chaque passage de badge sur le lecteur. Ce script permetr d'effectuer des opérations sur le badge comme par exemple : récupérer l'UID du badge, lire un identifiant sécurisé... etc.
Authentification	L'authentification est une opération à effectuer sur le badge ou sur une application du badge, elle consiste à vérifier que l'utilisateur est bien légitime à effectuer une ou plusieurs actions sur le badge (accéder à un ou plusieurs informations, écrire des données, créer ou supprimer des applications ou des fichiers ....etc).
Diversification	La diversification est une opération qui consiste via un algorithme et différents paramètres à chiffrer les clés qui permettront de s'authentifier sur le badge.
HSM (Hardware Security Module)	Composant considéré comme inviolable. Le HSM consiste à générer, stocker et protéger des clefs cryptographiques.

Chapitre 2. Présentation

Une applet est un petit script qui est exécuté au sein du module à chaque passage d'un badge. Ce script décrit avec un langage simple les manipulations à réaliser avec le badge. Cela peut aller d'une opération simple telle que récupérer le numéro de série et le remonter à l'UTL, à une opération plus complexe telle que s'authentifier sur une application (si le badge est un Desfire), avec ou sans diversification, lire le contenu d'un fichier, traiter éventuellement les données lues, lire un autre fichier par exemple une signature, vérifier cette signature, ... Puis remonter sur le bus un identifiant optionnellement manipulé (tronqué, inversé, converti, ...).

Une ou plusieurs applets pourront être téléchargées dans les modules : soit par le chargement d'un fichier texte via la page web de configuration de l'UTL , soit en effectuant un paramétrage et un téléchargement dans l'UTL via MICRO-SESAME Le choix de l'applet à exécuter est déterminé par la sélection d'un index (de 1 à 255). Si aucune applet n'est sélectionnée et/ou aucun fichier n'est téléchargé, une applet par défaut est activée, elle est inscrite en dur dans la flash. Cette applet par défaut lit le numéro de série du badge et le remonte sur le bus UTL.

Chapitre 3. Prérequis

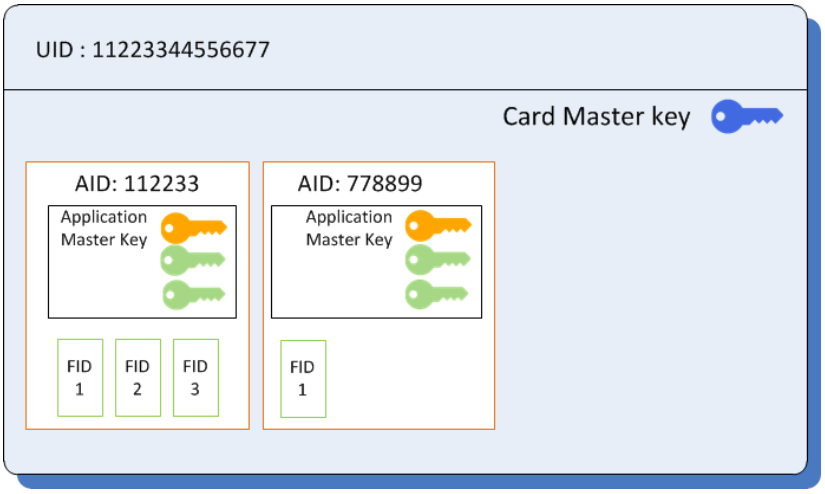
3.1. Structure des badges MIFARE DESFIRE

Un badge MIFARE DESFIRE contient :

- Un UID ou CSN (Card Serial Number) : code permettant d'identifier le badge, le CSN est une donnée accessible à tout le monde c'est à dire que cette donnée n'est pas sécurisée.

Si des données ont été encodées dans le badge, les éléments suivants sont aussi présents :

- Une ou plusieurs application(s) : élément identifiable par son AID (Application ID), qui contient un fichier de 14 clés indexés et de un ou plusieurs fichiers contenants des données
- Un ou plusieurs fichier(s) : élément se trouvant dans une application, les fichiers contiennent les données à récupérer.



3.2. Opérations pour la lecture d'une donnée sécurisée

Pour lire la donnée sécurisée d'un badge MIFARE DESFIRE, il est important de connaître les différentes opérations à réaliser dans le badge pour accéder à la donnée, ces opérations sont les suivantes :

1. Sélection de l'application contenant la donnée à lire
2. Dans le cas de l'utilisation de clés diversifiées : diversification de la clé de lecture
3. Authentification avec la clé de lecture (diversifiée si la diversification est utilisée) sur l'application
4. Lecture de la donnée dans le fichier contenant la donnée

Pour effectuer ces différents opérations, certains paramètres sont à connaître, ces derniers sont généralement rassemblés dans une documentation qui est créée pour l'encodage du badge, cette dernière se nomme "charte d'encodage". Ces différents paramètres sont les suivants:

- AID : Application ID
- FID : File ID
- Index de clé de lecture dans le badge Desfire
- Index de clé de lecture dans le HSM du module
- Type de clés (AES, 3DES ... etc)

- Type de diversification (dans le cas où diversification des clés utilisée)
- padding (dans le cas où diversification des clés utilisée)
- Utilisation ou non de la sous clé de diversification (bourrage ou K1/K2)
- Longueur de la donnée à lire
- Emplacement du début de la donnée

Chapitre 4. Écriture d'une applet

4.1. Syntaxe

La syntaxe utilisée pour "développer" une applet ainsi qu'une partie des commandes supportées sont fortement inspirées du langage TCL (Tool Command Language, se prononce Tickel). Voici l'ensemble des règles nécessaire à connaître pour écrire une applet :

- Il est fortement conseillé d'écrire une commande par ligne, toutefois, une ligne peut contenir plusieurs commandes si elles sont séparées par un point-virgule
- Toutes les commandes retournent un résultat (une chaîne de caractères vide ou non vide, ou "ERROR" dans le cas d'erreurs rencontrées lors de l'exécution de la commande)
- Les commandes peuvent nécessiter un ou plusieurs arguments
- Les arguments sont des mots (groupes de caractères séparés par des espaces)
- Tout ce qui suit le caractère # n'est pas interprété, et est donc vu comme un commentaire. Sauf pour la déclaration d'une nouvelle Applet où le "#" est suivi d'un "!" : *#! Nom de l'applet*
- Tout ce qui suit directement le caractère $ défini le nom d'une variable (caractères alphanumériques et souligné _), cette variable sera substituée par sa valeur avant l'exécution de la commande qui la contient comme argument
- Une commande peut contenir en guise d'argument une autre commande entourée des caractères [ ], cette dernière commande sera évaluée, et son résultat sera inscrit à sa place
- Si un argument nécessite de contenir des espaces ou retours à la ligne, il doit être entouré de guillemets "" ou d'accolades { }
- La différence entre accolades et guillemets est :
  - Les accolades protègent tout ce qu'elles contiennent de toute interprétation et substitution, ainsi on y peut écrire des noms de variable et des commandes, elles ne seront pas substituées
  - Les guillemets ne protègent pas ce qu'elles contiennent de substitutions et évaluations
- Une applet doit toujours commencer par une ligne contenant *#! Nom de l'applet*

4.2. commandes

4.2.1. set : déclaration d'une variable

La commande "set" permet de déclarer une variable et d'y affecter une valeur

Syntaxe : set VAR_NAME VALUE

Exemple

```
set ma_variable "123456"
set var1 123
set var2 456
```

Afin d'accéder à la valeur d'une variable la commande est la suivante: $VAR_NAME (le $ n'est pas utilisé dans les commandes "set" et "unset").

4.2.2. unset : Suppression d'une variable

Cette commande permet de libérer une ou plusieurs variables préalablement déclarées.

Syntaxe : unset VAR_NAME1 VAR_NAME2 VAR_NAME3 ...

Exemple

```
unset var1 var2
```

Dans cette commande le caractère "$" placé avant la le nom de la variable ne doit pas être utilisé.

4.2.3. string : manipulation de chaîne de caractère

Cette commande permet la manipulation de chaînes de caractères. A noter que dans une applet tout est considéré comme une chaîne de caractères : variables, retours de fonctions, arguments...etc.

Syntaxe : string SUBCOMMAND ARGS

SUBCOMMAND étant l'action qui va être effectué sur la chaîne de caractère. Les SUBCOMMAND pouvant être utilisés sont les suivants :

- length STRING: Permet de déterminer la longueur de la chaîne de caractères

Exemple

```
string length "123456"
La commande ci-dessus retournera la valeur 6.
```

- range STRING START STOP: Permet de Retourner une sous-chaîne d'une chaîne STRING depuis le caractère à la position START, jusqu'à la position STOP

  La valeur minimum de START est 0, STOP doit être obligatoirement plus grande que START.

Exemple

```
string range "abcde" 2 3
La commande ci-dessus retournera la valeur "cd".
```

- reverse STRING : permet d'inverser les caractères d'une chaîne

Exemple

```
string reverse "abcdef"
La commande ci-dessus retournera la valeur "fedcba"
```

- reverse_bytes STRING (ici STRING représente une chaîne de caractères hexadécimale) : permet d'inverser les octets de la chaîne de caractères. Cette commande est utilisée pour lire par exemple le numéro de série d'un badge en sens INVERSE ISO.

Exemple

```
string reverse_bytes "1234ABCD"
La commande ci-dessus retournera la valeur "CDAB3412"
```

4.2.4. if ... else : Conditions

Cette commande permet l'exécution de commandes conditionnées au résultat de l'évaluation d'une expression

Syntaxe : if { EXPRESSION } { BLOCK_1 } else { BLOCK_2 }

EXPRESSION est de la forme : OPERAND_1 OPERATOR OPERAND_2 , OPERATOR est actuellement soit "==" , soit "!="

Exemple

```
if { $result == "ERROR" } {
exit
} else {
# poursuite de l'applet # ...
}
```

La condition ci-dessus arrête l'exécution de l'applet dans le cas où la variable "result" retourne "ERROR".

4.2.5. exit : arrêt de l'exécution de l'applet

Cette commande permet l'arrêt prématuré de l'exécution de l'applet

Syntaxe : exit

4.2.6. get_uid : obtention numéro de série

Cette commande permet de récupérer le numéro de série du badge présenté.

Syntaxe : get_uid

4.2.7. desf_select_aid : Sélection d'un AID (Application ID)

Cette commande permet, lorsque un badge Desfire est présenté, de sélectionner une application identifiée par son AID. Si l'application n'est pas présente dans le badge, la valeur retournée est "ERROR".

Syntaxe : desf_select_aid NUM_AID

Exemple

```
desf_select_aid 112233
if { $? == ERROR } { exit }
```

Le code ci-dessus permet de sélectionner l'application ayant pour AID "112233", en cas de non présence de l'AID l'exécution de l'applet se termine.

4.2.8. diversify : la diversification

Cette commande permet de réaliser une diversification de clé suivant divers algorithmes. La clé à diversifier se trouve dans le module cryptographique (hsm), et n'est manipulable qu'à travers son index. La commande retourne "ERROR" si une erreur est rencontrée.

Syntaxe : diversify TYPE HSM_KEY_INDEX PARAMS

TYPE : diversification utilisée.

HSM_KEY_INDEX : est l'index de la clé à diversifier se trouvant dans le HSM

PARAMS : sont les paramètres utilisés pour effectuer la diversification. Pour le type AN10922 les différents paramètres sont :

- UID
- AID : Numéro de l'application en hexadécimal
- BADGE_KEY_INDEX : index de la clé point de vue du badge
- PADDING : valeur du padding utilisé

Exemple

```
set uid [get_uid]
set aid F02233
set desf_key_id 1
set hsm_key_id 20
set padding 80000000000000000000000000000000000000000000
diversify "AN10922" $hsm_key_id $uid $aid $desf_key_id $padding
```

Le code ci-dessus permet de mettre l'UID, l'AID, le BADGE_KEY_INDEX, le HSM_KEY_INDEX et la valeur du padding dans des variables, puis d'utiliser ces variables afin d'effectuer une diversification AN10922.

4.2.9. desf_authenticate : authentification

Cette commande s'utilise après la sélection d'une application, elle permet de s'authentifier sur l'application en indiquant le type de clé à utiliser (3DES, AES...Etc.), l'index de la clé à utiliser du badge Desfire ainsi que l'index de la clé du module crypto du mlp2.

Dans le cas de l'utilisation d'une clé diversifiée, la commande diversify doit être préalablement appelée (et vérifiée sans erreur), de plus, il faudra remplacer l'index de la clé dans le module crypto par le mot clé "KDIV"

Syntaxe : desf_authenticate ALGO DESF_KEY_INDEX HSM_KEY_INDEX

ALGO : correspond au type de clé (AES, 3 DES...etc)

Actuellement limitée à AES

DESF_KEY_INDEX : Représente le numéro d'index de la clé de lecture côté badge Desfire

HSM_KEY_INDEX : Représente le numéro d'index de la clé de lecture côté HSM (Module de crypto du MLP2).

Exemple : Authentification non diversifiée

```
desf_authenticate AES 1 20
```

La commande ci-dessus permet de s'authentifier avec une clé non diversifiée de type AES, L'index de clé utilisée dans le badge est l'index numéro 1 et côté hsm (module de crypto du MLP2) l'index est le numéro 20.

Exemple : Authentification diversifiée

```
desf_authenticate AES 1 KDIV
```

La commande ci-dessus permet de s'authentifier en avec une clé diversifiée de type AES, l'index utilisé est le numéro 1 du badge Desfire et côté MLP2, c'estla clé diversifiée calculé (KDIV) lors de la commande "diversify" qui est utilisée.

4.2.10. desf_read_data : lecture de la donnée

Cette commande permet de lire un fichier (ou extrait de fichier) d'un badge Desfire, après sélection de l'application et authentification. La commande retourne ERROR si une erreur est rencontrée.

Syntaxe : desf_read_data DESF_FILE_ID DESF_DATA_OFFSET DESF_DATA_LEN

DESF_FILE_ID : index de fichier Desfire à lire, valeur entière ou hexadécimale (si préfixée par 0x)

DESF_DATA_OFFSET : position à partir de laquelle lire, valeur entière ou hexadécimale (si préfixée par 0x)

DESF_DATA_LEN : nombre d'octets à lire (maximum 255), valeur entière ou hexadécimale (si préfixée par 0x)

la valeur 0 indique de tout lire, mais attention au débordement, il est plus prudent de maîtriser le nombre d'octets à lire

Exemple : commande de lecture d'une donnée de 7 octets dans le fichier 1

```
desf_read_data 1 0 7
```

4.3. Limitations

Pour que l'applet soit fonctionnelle, ci-dessous les limitations à respectées

- Le nombre maximum de récursion de la commande 'if' est de 8
- Le niveau de récursion de commandes est limité à 2 (La commande *"set uid [get_uid]*" utilise une récursion)
- Le nombre maximum d'applets permis par fichier est 255
- La taille maximum d'un fichier d'applets de doit pas excéder 20 Ko
- La taille maximum d'une applet, débarrassée de ses commentaires, ne doit pas dépasser 2 Ko
- Le nombre maximum de variables utilisables est de 16. Utiliser si nécessaire la commande unset sur les variables qui ne seront plus utilisées
- le nombre maximum de caractères pour le nom des variables (et des commandes) est de 64

- Le nombre maximum de caractères contenus dans une variable, est actuellement de 128. Ce nombre est également la limitation du nombre de caractères dans un argument, et par conséquent, toutes les commandes doivent retourner un résultat inférieur à cette limite.

Chapitre 5. Exemples d'applet

5.1. Lecture de numéro de série

L'exemple ci-dessous permet de lire l'uid d'un badge MIFARE CLASSIC ou MIFARE DESFIRE

Exemple :lecture du CSN (UID)

```
#! applet 1
export [get_uid]
```

5.2. Lecture d'un identifiant sécurisé d'un badge DESFIRE sans diversification

L'exemple ci-dessous permet de lire un identifiant se trouvant dans le fichier 1 de l'application F531AF. L'authentification à cette application n'utilise pas de clés diversifiées.

Exemple : Authentification non diversifiée

```
#! applet
# création des variables contenant les informations
# pour la lecture des données
set aid F531AF
set desf_file_id 1
set desf_key_id 1
set hsm_key_id 20
set id_size 10
# Sélection de l'application
desf_select_aid $aid
# arrêt de l'exécution de l'applet en cas d'erreur
if {$? == "ERROR"} {
exit
}
# Authentification sur l'application sélectionnée
desf_authenticate "AES" $desf_key_id $hsm_key_id
# arrêt de l'exécution de l'applet en cas d'erreur
if {$? == "ERROR"} {
exit
}
# Lecture de la donnée
desf_read_data $desf_file_id 0 $id_size
# écriture du résultat dans la variable badge_id
set badge_id $?
# arrêt de l'exécution de l'applet en cas d'erreur
if {$badge_id == "ERROR"} {
exit
}
# Envoie de l'identifiant lu à l'UTL
export $badge_id
```

5.3. Lecture d'un identifiant sécurisé d'un badge DESFIRE avec diversification

L'exemple ci-dessous permet de lire un identifiant se trouvant dans le fichier 0 de l'application F531AF. L'authentification à cette application utilise des clé diversifiées.

Exemple : Authentification diversifiée

```
#! applet
# création des variables contenant les informations
# pour la diversification et pour la lecture des données.
set uid [get_uid]
set aid F531AF
set desf_file_id 0
set desf_key_id 1
set hsm_key_id 21
set padding_type "00000000000000000000000000000000000000000000"
set id_size 7
# diversification de type AN10922 de la clé de lecture
diversify "AN10922" $hsm_key_id $uid $aid $desf_key_id $padding_type
# arrêt de l'applet en cas d'erreur
set diversify_status $?
if {$diversify_status == "ERROR"} {
exit
}
# Sélection de l'application
# arrêt de l'exécution de l'applet dans le cas d'erreur.
desf_select_aid $aid
if {$? == "ERROR"} {
exit
}
# Authentification avec la clé diversifiée
# arrêt de l'exécution de l'applet dans le cas d'une erreur.
desf_authenticate "AES" $desf_key_id "KDIV"
if {$? == "ERROR"} {
exit
}
# Lecture de la donnée
# arrêt de l'exécution de l'applet dans le cas d'une erreur
desf_read_data $desf_file_id 0 $id_size
set badge_id $?
if {$badge_id == "ERROR"} {
exit
}
# Envoie de l'identifiant à l'UTL
export badge_id
```

Chapitre 6. Téléchargement des applets dans les modules

6.1. Téléchargement des applets via la page web de la TILLYS NG

Il est possible de télécharger les applets dans un module via la page web de l'UTL. Pour cela :

1. Écrire l'applet ou les applets dans un fichier ".txt"
2. Se connecter sur la page web de la TILLYS NG en tapant son adresse IP dans la barre de recherche d'un navigateur et en entrant les login de connexion
3. Ouvrir la page "ML upload" du menu déroulant "Maintenance"
4. Dans la partie "Applet upload", sélectionner l'adresse du module à télécharger ainsi que le bus sur lequel le module se trouve.
5. Cliquer sur le bouton "Parcourir", sélectionner le fichier ".txt" puis cliquer sur le bouton "Upload and transmit to MLv3"

Une fois le fichier applet téléchargé, il est nécessaire d'indiquer à quel index se trouve l'applet à utiliser, dans la partie "Applet Index":

1. Sélectionner l'adresse du module
2. Sélectionner le bus sur lequel se trouve le module
3. Sélectionner l'index de l'applet à utiliser
4. cliquer sur le bouton "submit"

Afin que cette configuration ne soit pas écrasée par celle de MICRO-SESAME lors d'un téléchargement, vérifier dans le logiciel MICRO-SESAME que l'option "paramétrage lors du téléchargement" soit bien désactivé : Menu MICRO-SESAME > Paramétrage > Unités de Traitement Local > Configuration > Bouton "Paramétrage lors du téléchargement" doit être de couleur rouge

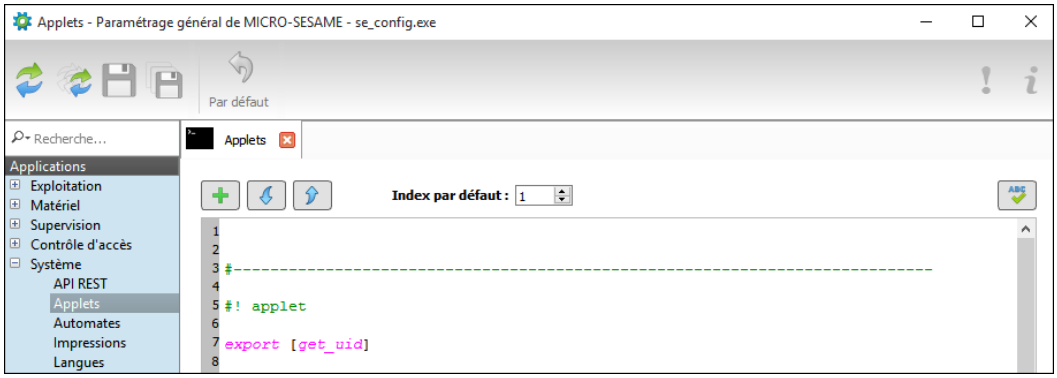
Avec cette option désactivée, l'UTL est à paramétrer entièrement via ses pages Web de configuration.

En cas d'erreur de téléchargement, des logs sont disponible : Menu déroulant "System information" > ML upload log > Partie "Applet" error log

6.2. Téléchargement des applets via MICRO-SESAME

Il est possible de télécharger les applets dans les modules via MICRO-SESAME. Pour cela il est nécessaire dans un premier temps d'enregistrer l'applet ou les applets dans la base de donnée . :

1. Ouvrir le paramétrage des applets de la fenêtre "Paramétrage général" de MICRO-SESAME : *Menu MICRO-SESAME > Paramétrage > Paramétrage général > Applet*
2. Inscrire l'applet ou les applets dans l'éditeur proposé, ou importer via le bouton "importer" le fichier texte contenant les applets. Le bouton "+" permet d'insérer le code pour d'ajouter une nouvelle applet dans l'éditeur.



3. Vérifier qu'il n'y ai pas d'erreur via le bouton "vérifier syntaxe" se trouvant au coin supérieur droit de l'éditeur.
4. Sélectionner l'index par défaut qui sera proposé lors de l'ajout d'un module dans le paramétrage de l'UTL
5. Sauvegarder la configuration

Dans le paramétrage des UTL, configurer les différents modules qui utiliseront les applets enregistrés précédemment. Pour cela :

1. Ouvrir la fenêtre Unités de Traitement Local : *Menu MICRO-SESAME > Paramétrage > Unités de Traitement Local*
2. Sélectionner l'UTL concerné
3. Dans l'onglet "Configuration", vérifier que la fonction "Paramétrage lors du téléchargement" soit bien active (bouton de couleur vert)
4. Dans l'onglet Matériel, si le module n'a pas été paramétré, ajouter le module concerné : *cliquer sur le bouton "+", sélectionner le module puis cliquer sur le bouton "ajouter et fermer"*
5. Dans le tableau, paramétrer le protocole utilisé par le module pour communiquer avec l'UTL (voir fiche technique du module concerné) puis le type de badge qui sera lu par le lecteur.
6. Dans la colonne "Paramétrage MLv3 Applet", sélectionner l'index de l'applet à utiliser.

Pour terminer, effectuer un téléchargement des UTL afin de télécharger les applets dans les modules :

1. Ouvrir la fenêtre de téléchargement : *Menu MICRO-SESAME > Paramétrage > Téléchargement*
2. Sélectionner les UTL concernés puis "Mise à jour complète"
3. Cliquer sur le bouton "Mise à jour"

Pour plus de renseignements sur le paramétrage du logiciel, se référer à la documentation MICRO-SESAME.