ChipMan
easy encoding and printing

2017
Technical Documentation

Version 1.4

MIFARE
Registered Partner

MP-Sys

Contact:

MP-Sys GmbH
Rahserstr. 58
41747 Viersen

Phone: +49 (0) 2162 22 96 4
Fax: +49 (0) 2162 22 62 7

Internet: http://www.mpsys.de
Email info@mpsys.de

Copyright and Trademarks

Content

General

Chip-Manager

Script Commands

Contact based Chips

Mifare Classic

Mifare Ultralight

Mifare Plus

Mifare Desfire EV1

Legic

Appendix A

Prerequisites

ChipMan is a 32 bit Windows application and can be run under:

Windows®XP, Windows®Vista (32/64 bit), Windows®7 (32/64 bit) as well as Windows®8 (32 / 64 bit) and within a virtual machine such as VMware®.

An installed PC/SC smart card reader is required for the coding of a chip card.

For the use of a camera and / or scanner, these devices need to be installed on Windows as imaging devices.

For the use of a scanner, a corresponding TWAIN driver must be installed.

For the use of Fargo and Zebra card printers an installed SDK could be necessary.

Versions

Available ChipMan versions:

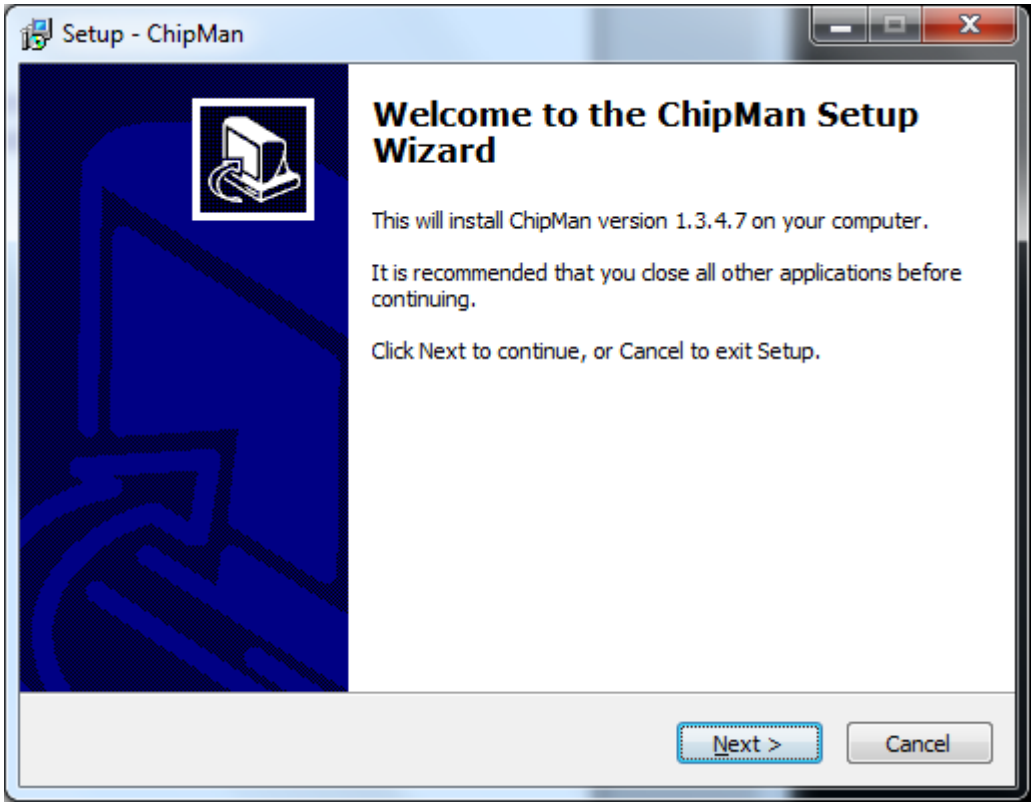
PrintBasic	Basic functions for printing cards with Text- and Image Objects
PrintPro	Card print with mail merge ADO / ODBC SQL database connectivity, text objects, image objects, Bar code objects, camera function, scanner, database write back functions
Contact	PrintPro with additional encoding functions of: Contact chip cards, i²c bus types such as Atmel, STMicroelectronics, GEMplus ST14C02C, ST14E32...M14256, GFM2K...GFM32K, AT24C01A...AT241024 2 wire bus types such as Siemens SLE4432, SLE4442 3 wire bus types such as Siemens SLE4418
Mifare	PrintPro with additional encoding functions of: MIFARE Classic 1k / 4k, Ultralight encoding
MifarePlus	PrintPro with additional encoding functions of: MIFARE Plus, Classic 1k / 4k, Ultralight, Ultralight C
Desfire	PrintPro with additional encoding functions of: Desfire, Desfire EV1
Professional	PrintPro with additional encoding functions of MIFARE Classic 1k/4k, MIFARE Plus, Ultralight C, Desfire EV1
Starter KIT	SCM SCL011 USB Reader CD with driver software & ChipMan software 6 RFID ISO cards included (2x MIFARE 2x DESFIRE EV1 2x ULTRALIGHT) Fully functional with the enclosed card. Card printing contains the fixed text "Printed with Chipman"
License Manager	Global network Licence, offers ChipMan clients to run within a network without a local license.

Installation

The ChipMan software is available on the following media:

- **USB memory stick**
  (contains chipman setup, also the license, a further licensing is not required)
- **CD media**
  (License required)
- **Download file**
  (License required)

Please start the Setup program (chipman_setup.exe) to install the chipman application. (Setup can be found on all media, or available as a download)



General

Licensing

A license key is required depending on the purchased version for the versions on CD or via download, which will be sent to you by email.

The corresponding product key of your installed ChipMan version is required for the provision of the license key.

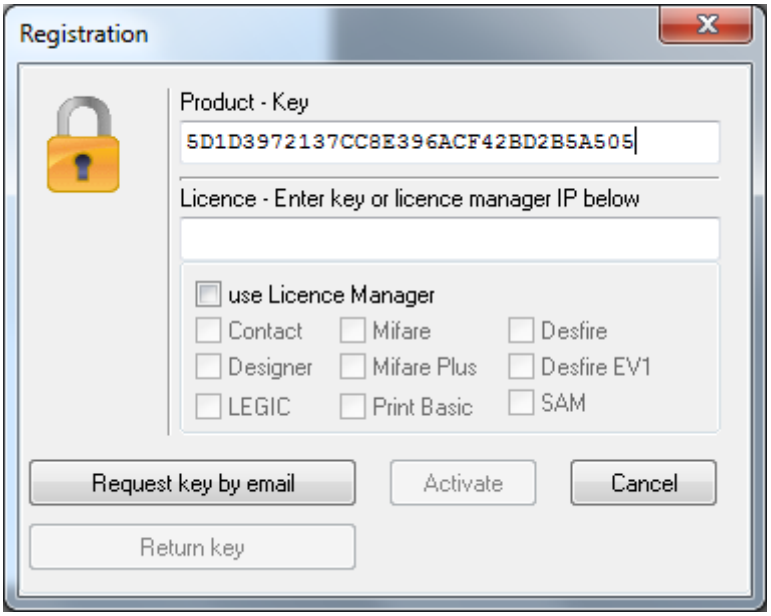
This product key is hardware-dependent. Running the setup at a later time or on another PC once again, this will change your product key and you will need a new license key.

Once purchased, you can return your old key when changing your PC and get a new key for free. Alternatively, we offer a license manager to use in your LAN, which passes the licenses. A local license is no longer necessary, ChipMan can run so often and from any PC, as there are licenses on the network.

Another alternative is to use a USB memory sticks, which already contains the appropriate license file for delivery. ChipMan license is activated automatically as soon as the stick in your system is detected.

To get your license key, proceed as follows:

Start the ChipMan application and click the menu info / registration:



Licensing

1. Licence request

Once you have purchased your ChipMan version, please send your product key by email to service@mpsys.de, you will receive your license key immediately.

2. Activate your ChipMan license

To unlock your ChipMan version, please enter your license key you received by email and activate the license with the "activate" button.

3. License return before new installation

If you want to return the installed key, in case of installing the software on another PC, click the "Return key" button and send us your “return key” by email. The email content will be automatically generated with your standard mail client.

4. Your purchased licence

The capabilities acquired with your license, are displayed with a check mark.

Note:

This manual describes the functions of the ChipMan professional version, possibly not all items available in your version.

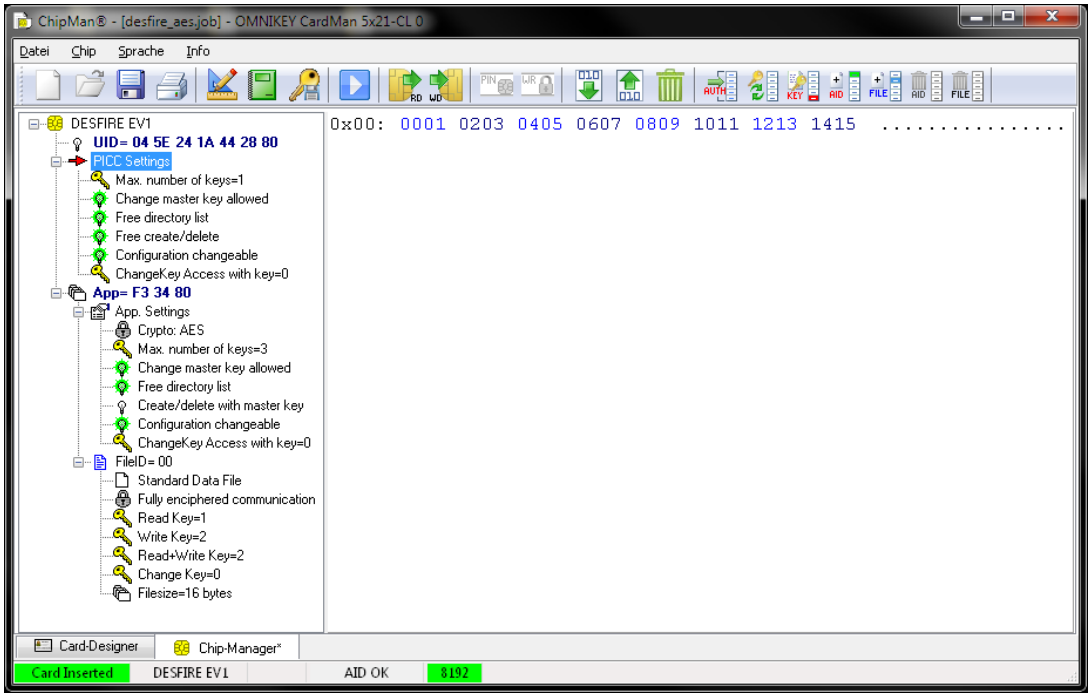
Password

The script editor, the key container (KeySafe), the user rights and the edit database button are protected with a password function, as well as the designer.

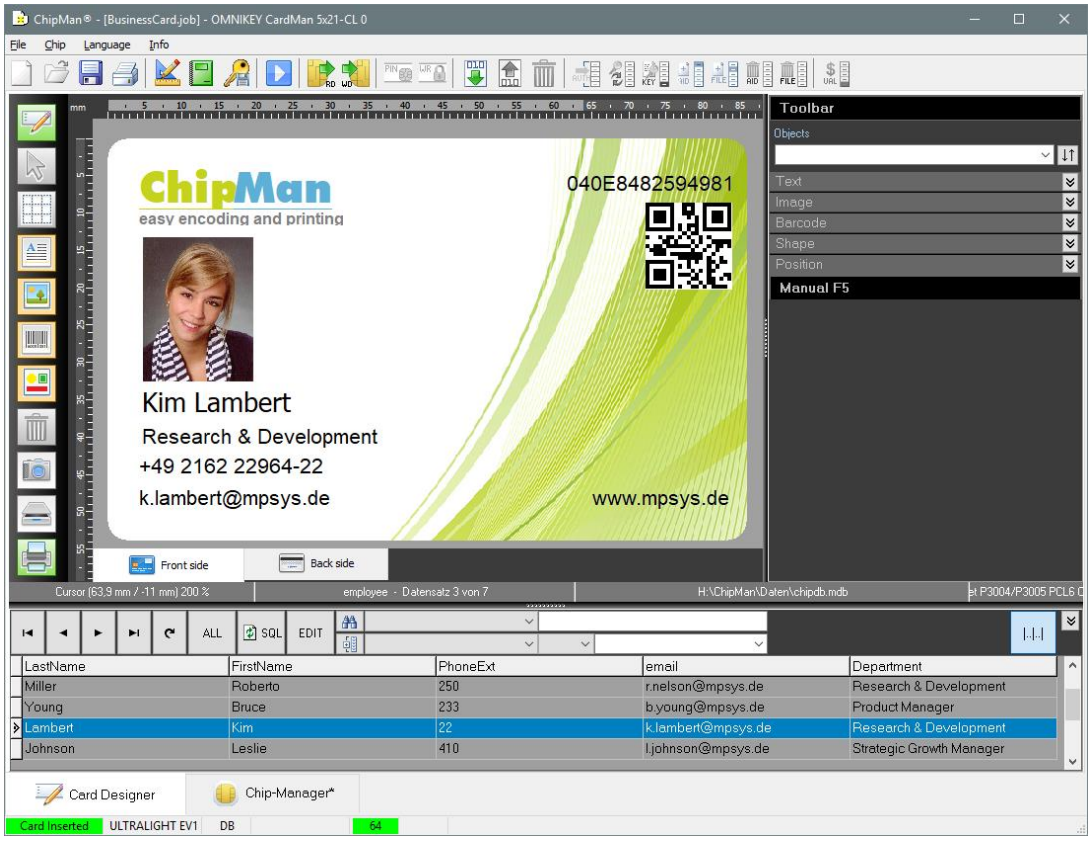
The default password is: **admin** and can be changed via "Change password" menu item.

Controls

Chip-Manager: The interface for encoding of smart cards:



Card-Designer : The interface for card design and printing

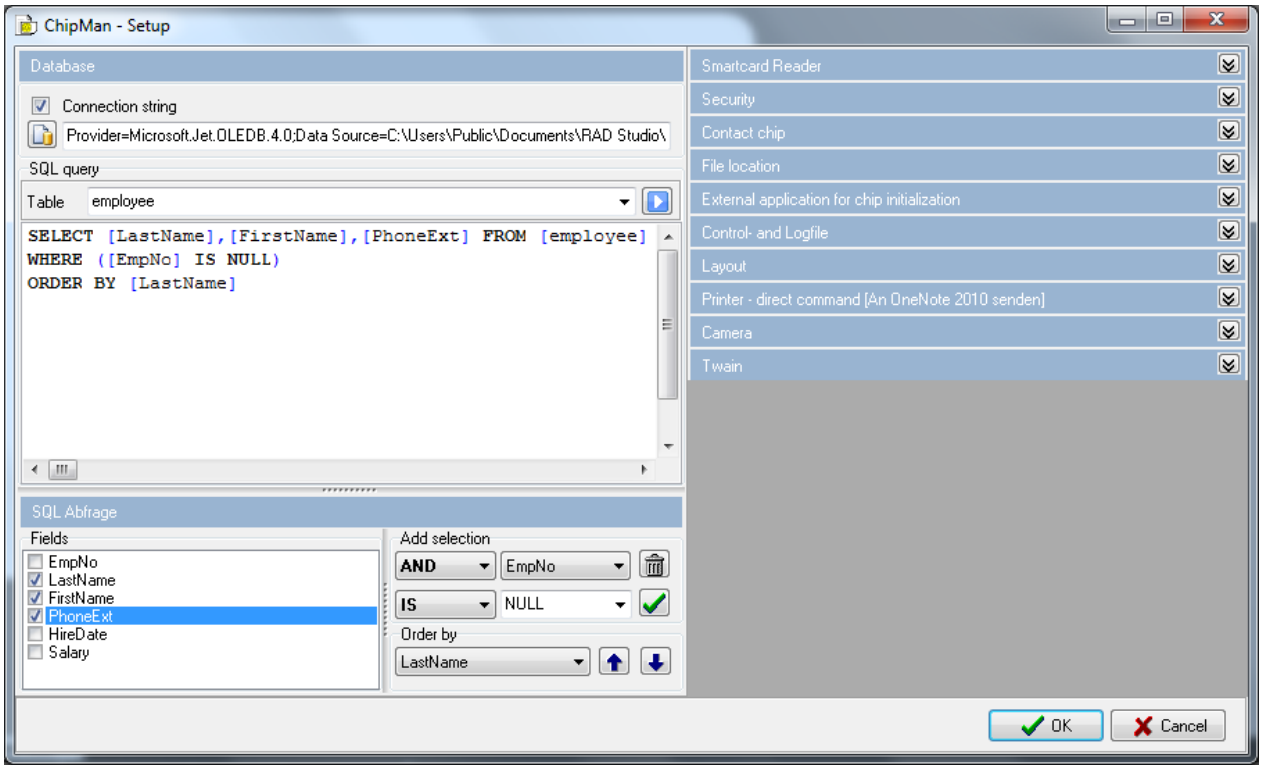


General

Settings

In ChipMan setup, all adjustments are made centrally and can be saved in a job file.

Setup:



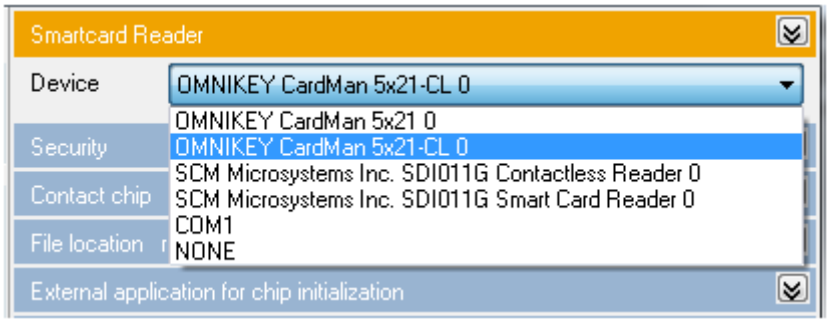
The possible settings are described below in details.

Setup

Smartcard Reader

To select a smart card reader, open the ChipMan setup with the menu item “settings"

Under the tab "smart card reader / device " all available readers will be listed:

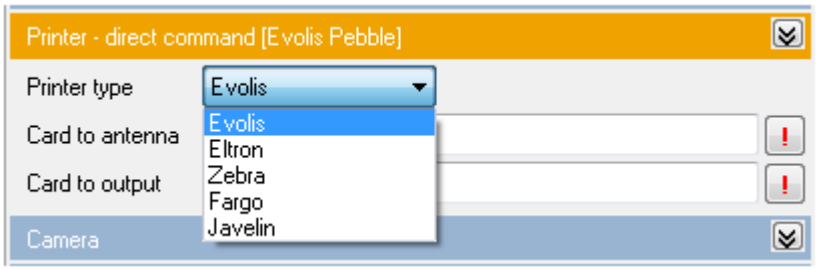


Note:
The reader must already be connected to your system before starting the program. Subsequently connected devices are not detected.

Card printers

1. Under the menu item file / printer settings, select the printer to be used.

2. Open the ChipMan Setup and select the desired printer type under the tab "Printer - direct command".



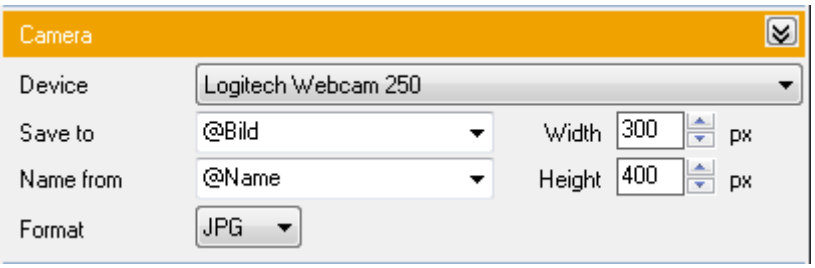
"Card to output" and "Card to antenna" transport commands are already filled and can be changed manually. To test this functionality, click on the [!] Button. The printer must be connected and ready for use and should now perform the desired functions of transport.

For available printer commands, please refer to the printer documentation. ChipMan supports in addition the SDK of the manufacturer Zebra and Fargo printer.

Camera

All connected cameras are available in register "Camera / device".

Hinweis:
Your camera must be connected to your system before starting the program. Subsequently connected devices are not detected.



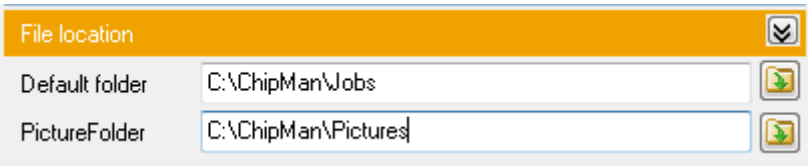
Save the image:
With a connected database, you can select the database field to hold the image filename. The image file name can be automatically generated from the database field selected from the pull down menu “name from”

File name of the image:
The file name can be composed of database fields and fixed texts in the field "File name". If this field is empty, the name is formed automatically from the current date and time.

Image type:
With the menu item "Format" select the image type to use (bmp, jpg) and also the file extension.

Image size:
The image size is set in "width" and "height" of pixels. This setting is taken over by the selection frame in the live camera image.

File location:
File location of the images are set under the menu item "Picture folder" in the settings “File location”. The database stores only the name of the image, the image directory can thus subsequently changed without changing database contents.



Scanner

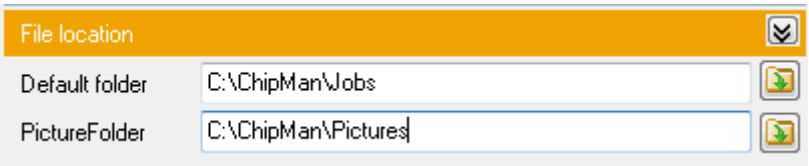
All connected Twain Scanners are available in register register „Twain / Device“

Save the scanned image:
With a connected database, you can select the database field to hold the image filename. The image file name can be automatically generated from the database field selected from the pull down menu “name from”

File name of the image:
The file name can be composed of database fields and fixed texts in the field "File name". If this field is empty, the name is formed automatically from the current date and time.

Image type:
With the menu item "Format" select the image type to use (bmp, jpg) and also the file extension.

File location:
File location of the images are set under the menu item "Picture folder" in the settings “File location”. The database stores only the name of the image, the image directory can thus subsequently changed without changing database contents.



Setup

Contact chip

Adjustments for contact chips are made under register "Contact chip"

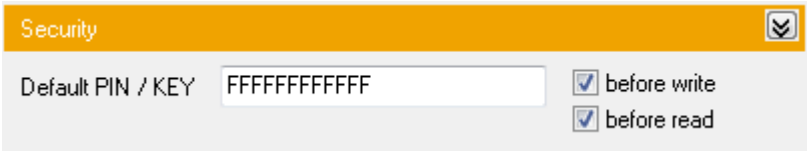
With enabled "show write protection" option, read-only memory areas are highlighted in the editor with a red background.
These memory areas can no longer be changed.

Enable the "auto" function to automatically determine the memory size of the chip.
This feature does not work with all types of chip.

The memory size, the starting address and the number of bytes will be used by the functions "chip data read" and "Write chip data" in the
manual mode of the editor.

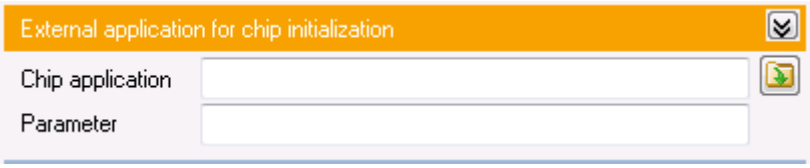
Security

Under the tab "Security", the default card pin or card key of the smartcard is set.
ChipMan automatically sets the default values of
the approved type. A read or write operation is possible in manual mode immediately after the contact of the smartcard (in the delivery State).
Prerequisite: The default PIN/KEYs of the card have not been changed yet.



External Application

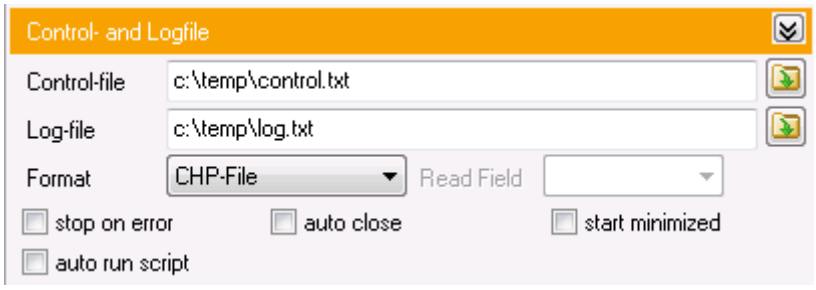
If an external application is to be used for Chip initialization, this encoding application and its necessary program parameters can be set under register "External Application ..."



The program parameters can contain database fields (@name..etc), fixed text or variables (%VAR1..etc). A transfer file (control file) to host is set in the register "Control and log file".

Control & Logfile

Settings for the communication with external applications are made in the register "Control- and Logfile". ChipMan can act both as a coding application (Plug-In mode) or as a host application for an external code module.



Control file
The control file contains the result of the last script execution. Error message are written as plain text (e.g. "WRITE ERR") or for a successful execution the text "OK" will be returned. The control file is re-created on every script execution, an existing control file will be overwritten.

Log file
The log file contains the current date and time, as well as tab separated the parameters %JOB, %ID,%TEXT, %FILE, %PIN and the result of the last script execution. The entries are appended to an existing log file.

In Plug-In mode,
ChipMan writes the control file, so it provides the result of encoding.
In host mode,
ChipMan evaluates the control file and responds to messages from the control file.

Control & Logfile

The log and control files are created automatically on any of the following conditions. A valid file name must be entered.

1. ChipMan was started with the parameter /Job=<jobname> /RUN
2. "auto run script" was selected and a smart card has been detected by the Reader.
3. A printing process with "RFID" coding was started and a smart card has been detected by the reader

Additional settings:

Format
The following formats are supported when creating the control file:

Format	Typ
CHP-File	Simple text file
Matica-Write	Transfer file for use with MATICA systems
Matica-Read	Transfer file for use with MATICA system and writeback in the database. A "Readfield" must be specified in addition. The Matica system uses this database field for further processing after the encoding.
Multicard	Transfer file for use with multicard systems

Auto Close
After the end of the script execution and the writing of control- & log files, the application exits automatically.

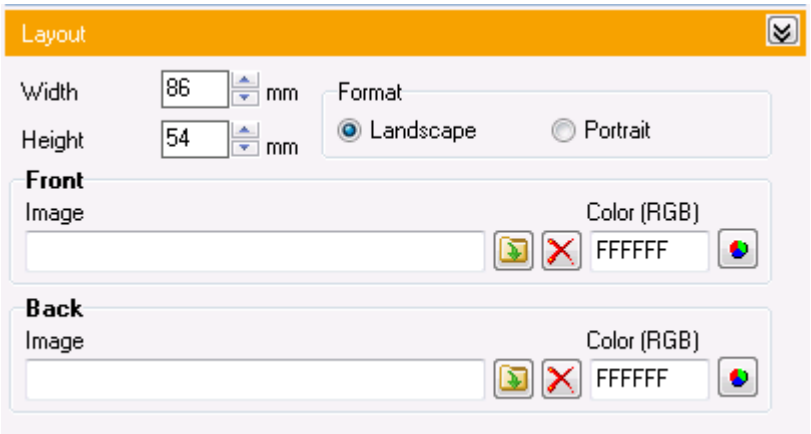
Start minimized
ChipMan will only be shown in the taskbar as an icon.

Stop on error
In the event of a fault and in conjunction with the function "Auto close" the application is **not** closed.

Auto run script
Das Script wird gestartet sobald der Smartcard Leser eine Karte erkannt hat oder wenn die Parameter /JOB=<jobname.job> /RUN angegeben wurden.

Layout

The card layout in width and height, portrait or landscape format and the background color for the front and back side of the card are set in the "Layout" tab.



The background can be both an image in jpg, bmp or wmf format, or a plain background defined by RGB values. (R = Red, G = Green, B = Blue)

The RGB value is expected in hex notation.

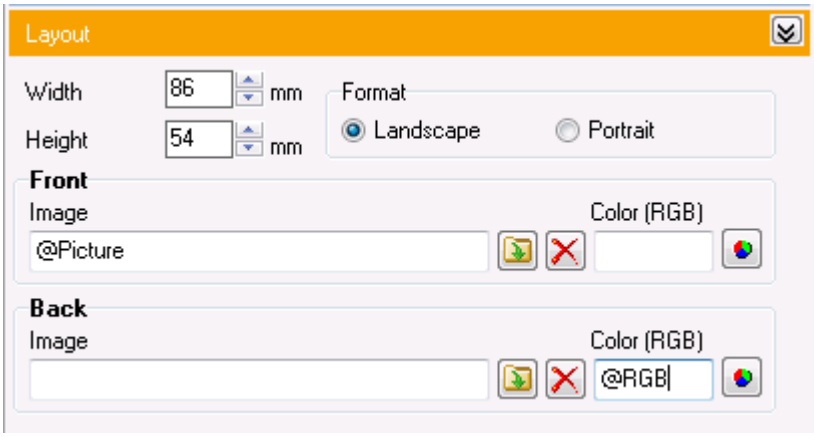
Example:

Red = FF0000
Green = 00FF00
Blue = 0000FF

The background image fills the whole card area, regardless of the actual aspect ratio of the graphics. In the printout the background image lies behind all other objects.

The background image or background color can also be controlled through a database field. To do this, insert the database field name instead of the image file name or the RGB value.

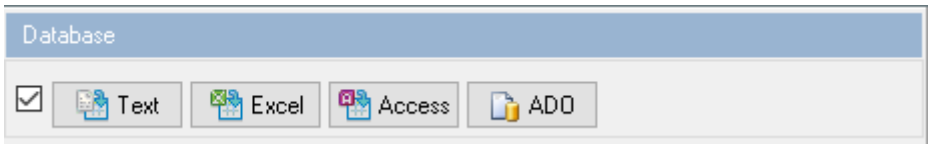
Example: (Database fields: @Picture, @RGB)



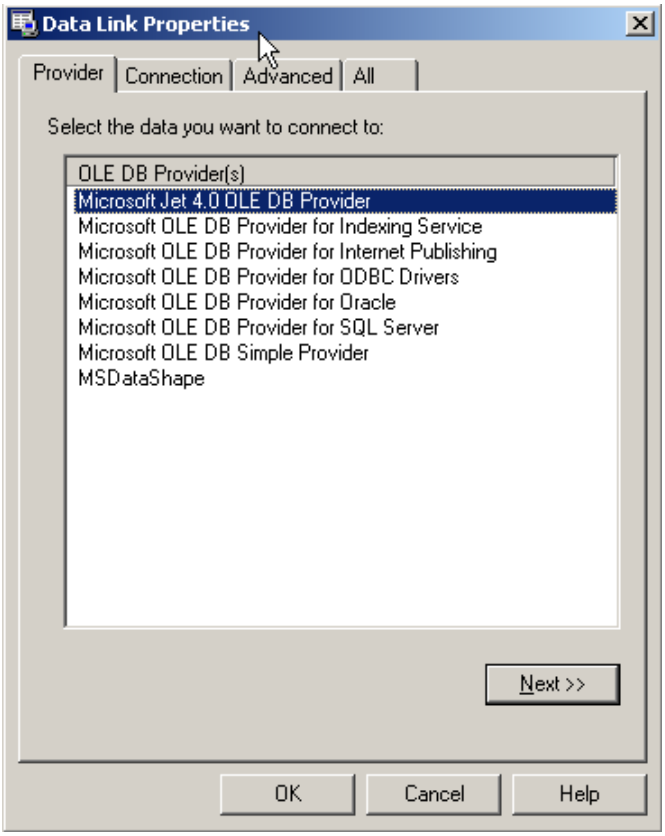
Setup

Database

Under the menu item "Database / connection string", you create a connection to the database menu-driven.



Menu-driven database connection:



Use the **Provider** tab for the type of database that you want to access, select the appropriate OLE DB provider.
Use the **Connection** tab to specify how a connection to the database is to be established.

The connection of the data link properties dialog box tab is provider-specific and displays only the connection properties that are required by the selected OLE DB provider.

With the connection properties, you can specify where your data is stored and how to connect to the data.

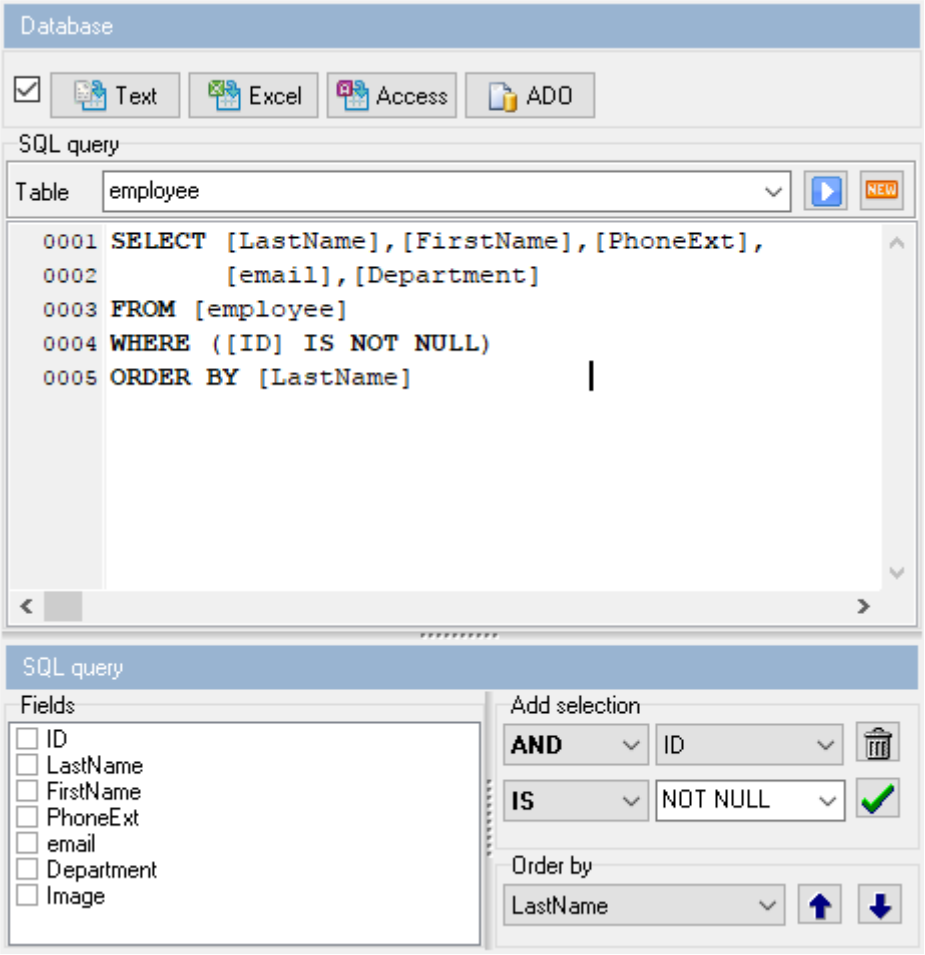
For more information, see the documentation that came with your OLE DB provider.

Database

After selecting the table to be used, a SELECT command is generated automatically, so all records are available in the DB browser.

To restrict the selection and / or set the sort order, an SQL Query Builder is integrated to quickly create simple SQL queries.

SQL Query example:



More extensive SQL queries can always be created or added manually, the possible SQL commands are not restricted and depends on the database command set.

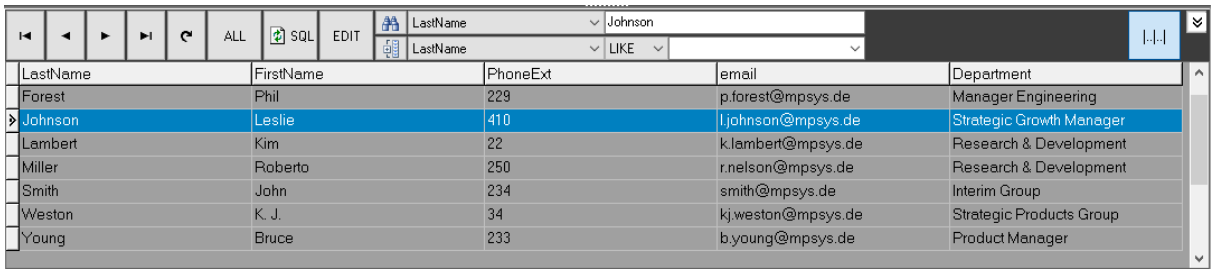
An SQL syntax check can be done with the button.

In case of success the first record is displayed, or in case of an error an appropriate message will be shown.

Database

The selected records from the SQL query are displayed in the Card Designer and are ready for production.

DB-Browser:



LastName	FirstName	PhoneExt	email	Department
Forest	Phil	229	p.forest@mpsys.de	Manager Engineering
Johnson	Leslie	410	l.johnson@mpsys.de	Strategic Growth Manager
Lambert	Kim	22	k.lambert@mpsys.de	Research & Development
Miller	Roberto	250	r.nelson@mpsys.de	Research & Development
Smith	John	234	smith@mpsys.de	Interim Group
Weston	K. J.	34	kj.weston@mpsys.de	Strategic Products Group
Young	Bruce	233	b.young@mpsys.de	Product Manager

Within the Database browser you can select individual or all records, navigate and search for records. Sorting by individual fields is possible by clicking on the field name. Another click on the field name change the sort order.

Note:
If possible, use a unique field with a primary key in a table. This speeds up database access as well as sorting and avoids errors in the selection of individual records.

Program parameters

By execute the Chipman application from host software (plugin mode) the following program parameters can / must be passed:

Parameter Name	ChipMan Variable	Usage	Example
/JOB	%JOB	Opens the job at program start. This parameter **must** be specified	/JOB="filename.job"
/PIN	%PIN	Data transfer freely usable e.g. as a Key	/PIN="12345"
/TEXT	%TEXT	Data transfer freely usable e.g. for chip encoding	/TEXT="Max Muster"
/ID	%ID	Data transfer freely usable e.g. for selection in the SQL Statement : *where Feld=%ID*	/ID="10023"
/FILE	%FILE	Data transfer freely usable e.g. for chip encoding. as well as a data file with a string list structure like: [Fields] Name=Smith Vorname=John	/FILE=''data.txt" /FILE="01020304FF"
/INI	%INI	Data transfer freely usable e.g. for chip encoding. as well as control file for Matica - systems	/INI="chip.ini" /INI="AA55AA"
/ROW		For use with Multi Card - systems as a control file.	/ROW="Row#.chk"
/PRINTCMD		Printer command. e.g. to insert a RFID card and move the card to the Antenna at program startup.	/PRINTCMD=Sic
/PRINT		Selects the specified records and starts the print dialog.	/PRINT=ALL /PRINT=n1,n2 n1=Start Record n2=End Record
/RUN		The script starts automatically after loading the job and successfully contacted a SmartCard	

Parameter

Variables

Variables are available within the ChipMan application for processing of data,. These variables are used to hold the data from the database, the smart card, a manual input or even program parameters.

Using these variables, data can be converted, checksums calculated, substrings are copied and many more processing steps are carried out.

For more information about data conversion, see the section "Tools"

The following variables are available:

Variable Name	**Usage**
%UID	Initialized with the UID of the recognized RFID card
%ATR	Initialized with the ATR of the recognized smart card
%NULL	Null string. Usage for delete other variable by using the copy command
%VAR1	Free variable not initialized
%VAR2	Free variable not initialized
%CAM	Holds the path and filename of the last camera image, Can an assigned to an Image Object.
%TWAIN	Holds the path and filename of the last scanner image, Can an assigned to an Image Object.
%DATE	Contains the current date in regional format settings
%EDITOR	Referenced data in the hex editor. Can be used as a source and destination.

The program parameters %JOB, %PIN, %TEXT, %ID, %FILE, %INI can be additionally used as variables and are initialized with the appropriate parameters.

Chip Folder

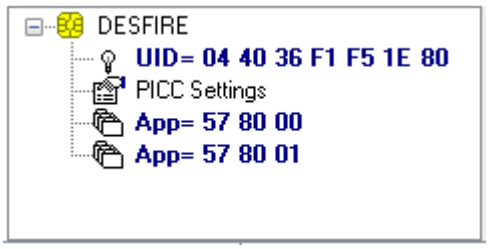
Once the reader contacts a card, ChipMan automatically detect the type of card and represents the view of these type of chip.

Depending on the chip type the UID or ATR and in case of Desfire card also the application directory are automatically loaded.

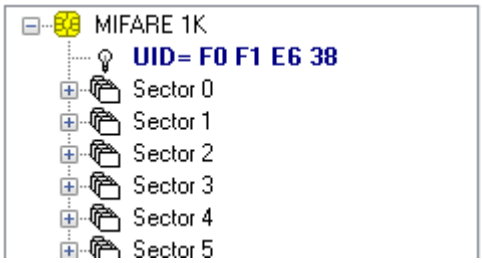
This information is shown in the Chip-Manager´s TreeView:

Examples:

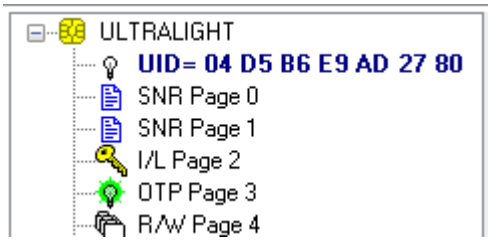
Desfire:



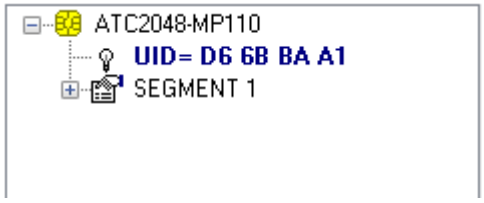
Mifare Classic:



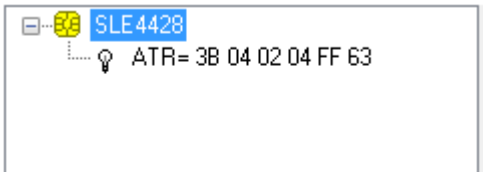
Ultralight:



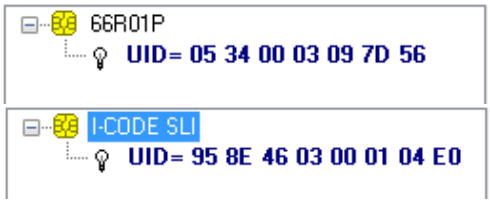
Legic:



SLE4428:



ISO14443



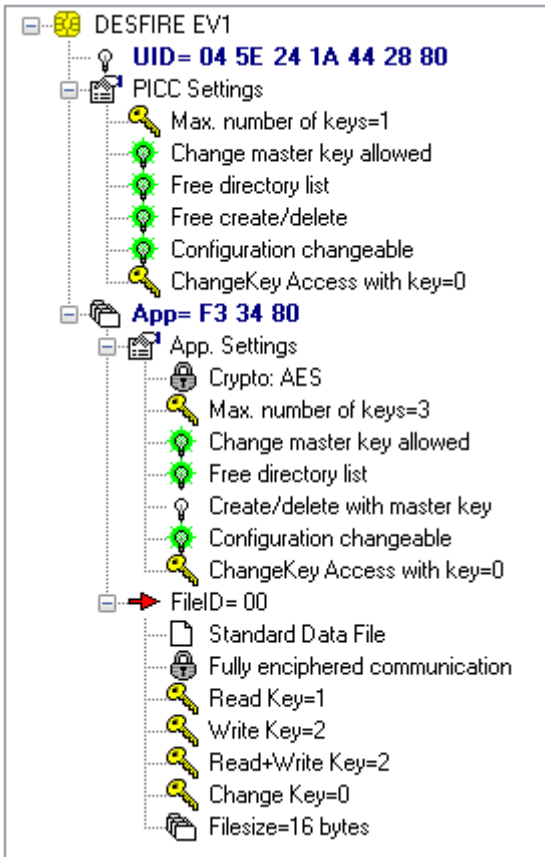
Chip Folder

Depending on the type of chip, a double-click on the directory entries will retrieved more information and further details are displayed.

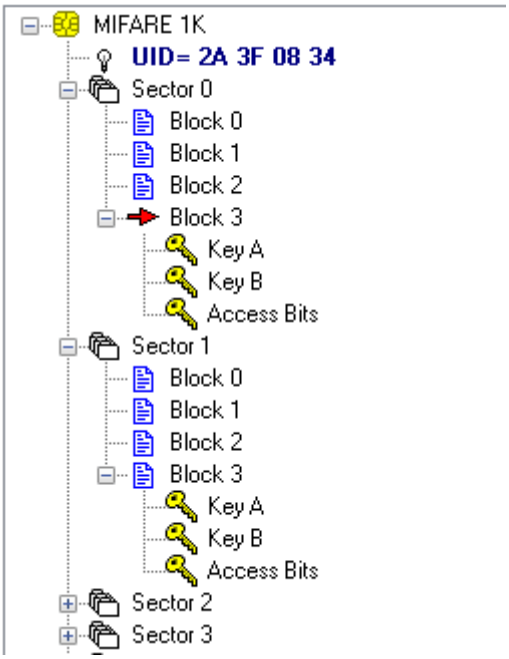
Depending on the configuration of the chip a previous authentication may be required to read the data. (see chip manufacturer documentation)

Example:

Desfire EV1:



Mifare 1k:



Chip-Manager

Hex Editor

The built-in hex editor provides chip data in hex & ansi, also provides the ability to edit data, store it in a binary file or load and print chip data.

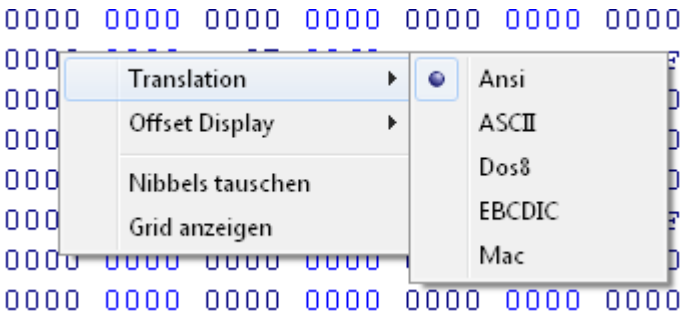
Hexeditor:

```
0x000: 2A3F 0834 2988 0400 47C1 2B57 5500 1607  *?.4)^..GÁ+WU...
0x010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x030: 0000 0000 0000 FF07 8069 FFFF FFFF FFFF  ......ÿ.€iÿÿÿÿÿÿ
0x040: 5465 7374 4131 4132 4133 0000 0000 0000  TestA1A2A3......
0x050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x070: 0000 0000 0000 FF07 8069 FFFF FFFF FFFF  ......ÿ.€iÿÿÿÿÿÿ
0x080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0A0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0B0: 0000 0000 0000 FF07 8069 FFFF FFFF FFFF  ......ÿ.€iÿÿÿÿÿÿ
0x0C0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0D0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0E0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0F0: 0000 0000 0000 FF07 8069 FFFF FFFF FFFF  ......ÿ.€iÿÿÿÿÿÿ
```

Use the TAB key to switch between the hex and text mode.

The editor is customizable via a context menu (right mouse button)

Context menu:



Hex - Editor

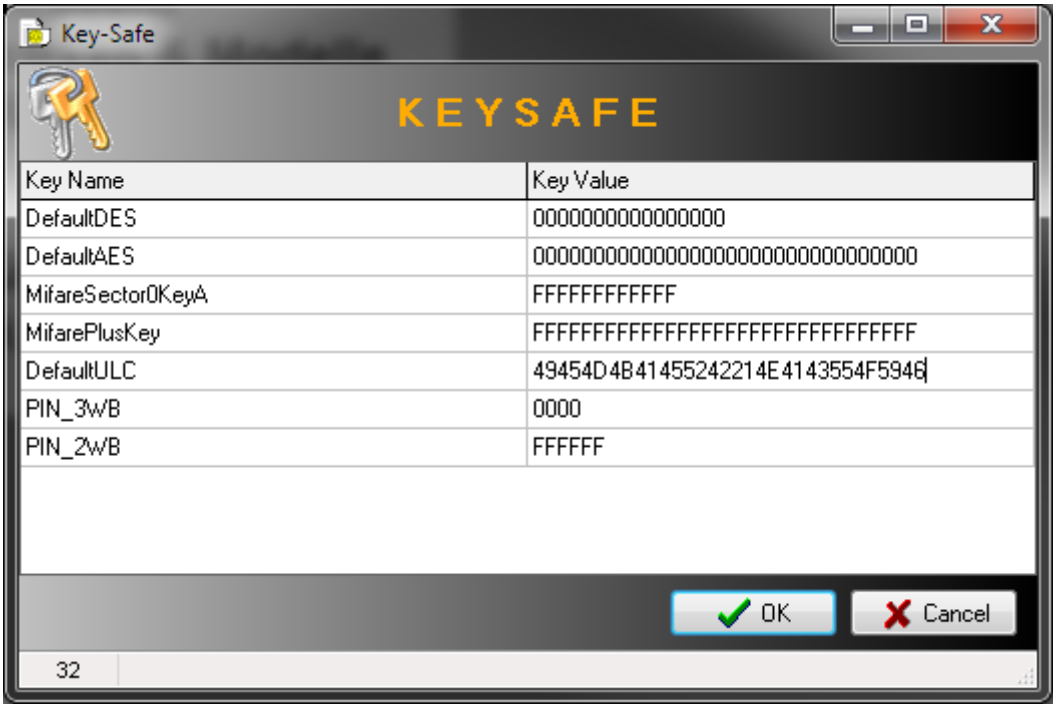
Keysafe - Editor

The KeySafe used to store and protect the keys used in the jobs and scripts.

Within the KeySafe each key is assigned to a name (alias). This name will be used in the script or for manual processing later. The script therefore never contains the actual key. This allows you to change the keys anytime in the KeySafe without also having to change the script. Also the script can be provided to any third party without the keys to announce.

Before opening the KeySafe, the system password is required.
(default: admin)

The KeySafe content is stored encrypted (AES) in the JOB file and is only readable within the KeySafe window.



The KeySafe contains key information in the form:

<Key Name> = <Key Value in Hex>

Example:
Key1=37362AF4E3A655EF

Wherever the specification of a key is required, the name of the key can now (here Key1) be used.

By default, the KeySafe contains the default keys of common chip types.

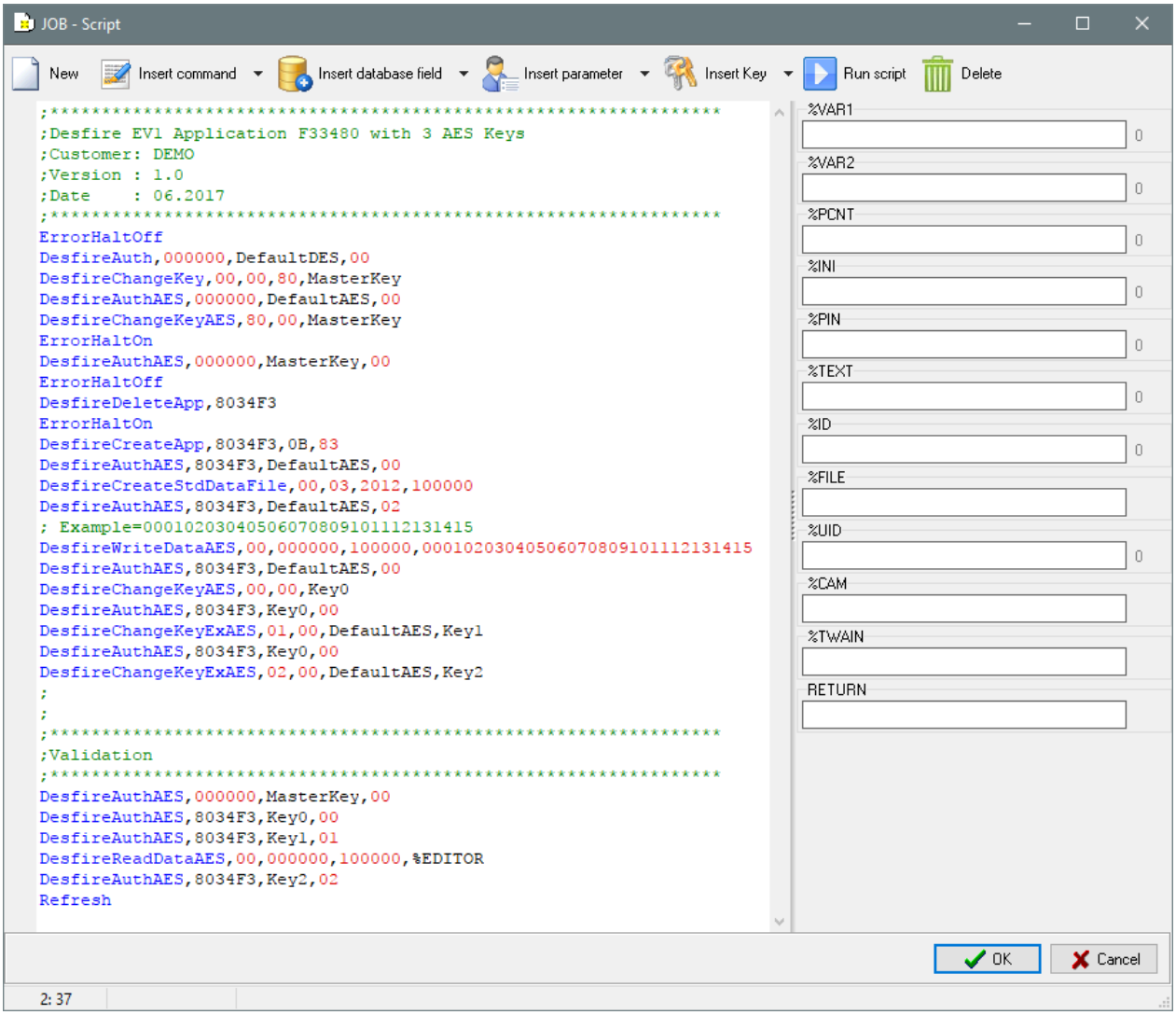
Script - Editor

The script editor is the "control center" of the job and is used for the automatic processing of chip encoding, database access and printer commands.

Within the script editor, all individual commands required for the job are grouped together and can then be processed automatically.

In case of an error, the script stopped on that line the error occurred and a error message will be shown or can be stored in a control or logfile.

Script example:



Script - Editor

Below you will find a list and a description of all script commands by category.
The desired command is selected by clicking on menu item "insert command" and is automatically entered into the script. The desired command parameters must only be entered on simple commands. More complex commands are supported by there appropriate wizards. You can therefore easily select the desired functions without having to know the exact syntax.

Command syntax in the script:

<commandl>, <Parameter 1>, <Parameter 2>, …, <Parameter n>

Parameter:

<target> is a placeholder for the storage destination to be written. This can be any variable (eg% VAR1,% EDITOR), or a database field (eg @ name).

<source> is a placeholder for an any data source (variable or database field) or even a fixed text (e.g. FF1245) to be read from.

Befehl	Kategorie	Syntax	Parameter
ReadChip **ReadChipHex**	Contact read data	ReadChip, <adr>,<len>,<target> ReadChipHex, <adr>,<len>,<target> ReadChip,CUR,<len>,<target> ReadChipHex,CUR,<len>,<target>	**adr=** start address in memory with adr = CUR (read from cursor position, the cursor pos is automatically incremented by len bytes) **len** = number of bytes with len = # nn (read until tag # nn) **target** = target field
WriteChip **WriteChipHex**	Contact write data	WriteChip, <adr>,<source> WriteChipHex, <adr>,<source> WriteChip, <adr>,%EDITOR,<len> WriteChipHex, <adr>,%EDITOR,<len> WriteChip,CUR,<source> WriteChipHex,CUR,<source> WriteChip,CUR,%EDITOR,<len> WriteChipHex,CUR,%EDITOR,<len>	**adr=** start address in memory with adr = CUR (write at cursor position, the cursor pos is automatically incremented by len bytes) **len** = number of bytes with len = # nn (read until tag # nn) **source** = data source
PresentPIN	Contact PIN	PresentPIN,<pin>	**pin=** Chip PIN
ChangePIN	Contact PIN	ChangePIN,<old>,<new>	**old=** oldr PIN **new=** new PIN
Protect	Contact Write protection	Protect,<adr>,<len>	**adr=**start address im Chip **len=**number of bytes
MifareAuth	Mifare Classic Sector authentification	MifareAuth,<AuthKeyA|B>,<sector>,<Key>	**AuthKeyA|B**=KeyA | KeyB **Sector**=Sector number **Key**=Key
MifareReadBlock **MifareReadBlockHex**	Mifare Classic Read Block	MifareReadBlock,<sector>,<block>,<target> MifareReadBlockHex,<sector>,<block>,<target>	**Sector**=Sector number **Block**=Block number **Target**=target field
MifareReadUID **MifareReadUIDHex**	Mifare UID	MifareReadUID,<target>	**Target**=target field

Chip-Manager

Befehl	Kategorie	Syntax	Parameter
MifareWriteBlock **MifareWriteBlockHex**	Mifare Classic Write Block	MifareWriteBlock,<sector>,<block>,<source> MifareWriteBlockHex,<sector>,<block>,<source>	**Sector**=Sector-No. **Block**=Block-No. **Source**=Datasource
MifareWriteKeyAccess	Mifare Classic KeyA, KeyB, AccessBits	MifareWriteKeyAccess,<sector>,<KeyA>,<ABits>,<KeyB> <KeyA>, <KeyB> = 6 Bytes <ABits> = 4 Bytes	**Sector**=Sector-No. KeyA= <Key> ABits= 4 Byte Access Bits KeyB= <Key> <Key> from KeySafe , Database field or Fixtext
UltralightReadPage	Mifare Ultralight Read page	UltralightReadPage,<page>,<target> UltralightReadPageHex,<page>,<target>	**Page**=Page - No. **Target**=target field
UltralightWritePage	Mifare Ultralight Write Page	UltralightWritePage,<page>,<source> UltralightWritePageHex,<page>,<source>	**Page**=Page - No. **Source**=data source
MifarePlusWritePerso	Mifare Plus Security Level 1 WritePerso	MifarePlusWritePerso,<Block>,<Data>	**Block**=Block address **Data**=Persodata / Key
MifarePlusCommitPerso	Mifare Plus Security Level 0 Commit	MifarePlusCommitPerso	
MifarePlusSL1Auth	Mifare Plus Security Level 1 Auth AES	MifarePlusSL1Auth, <Block>,<Data>	**Block**=Block address **Data=**Data / Key
MifarePlusSL1SwitchLevel	Mifare Plus Security Level 1 Switch Level	MifarePlusSL1SwitchLevel, <Block>,<Data>	**Block**=Block address **Data**=Data / Key
MifarePlusSL3Auth	Mifare Plus Security Level 3 Auth AES	MifarePlusSL3Auth, <Block>,<Data>	**Block**=Block address **Data**=Data / Key

Befehl	Kategorie	Syntax	Parameter
DesfireAuth	Desfire Auth DES	DesfireAuth,<AID>,<Key>,<KeyNo>	**AID**=Application ID **Key**=Key **KeyNo**=Key Number
DesfireAuthISO	Desfire Auth 3KDES	DesfireAuthISO,<AID>,<Key>,<KeyNo>	**AID**=Application ID **Key**=Key **KeyNo**= Key Number
DesfireAuthAES	Desfire Auth AES	DesfireAuthAES,<AID>,<Key>,<KeyNo>	**AID**=Application ID **Key**=Key **KeyNo**= Key Number
DesfireChangeKey	Desfire Change Key	DesfireChangeKey,<KeyNo>,<Ver>,<Crypt>,<Key>	**KeyNo**= Key Number **Ver**= Key version **Crypt**=Verschlüsselung **Key**= Key
DesfireChangeKeyAES	Desfire Change Key AES	DesfireChangeKeyAES,<KeyNo>,<Ver>,<Crypt>,<Key>	**KeyNo**= Key Number **Ver**=Schlüsselversion **Crypt**=Crypto mode **Key**= Key
DesfireChangeKeyEx	Desfire Change Key	DesfireChangeKeyEx,<KeyNo>,<Ver>,<OldKey>,<NewKey> (wenn der zu ändernde Schlüssel nicht der aktuelle Anmeldeschlüssel ist)	**KeyNo**= Key Number **Ver**= Key version **OldKey**=old key **NewKey**=new key
DesfireChangeKeySettings	Desfire Key Config	DesfireChangeKeySettings,<config>	**Config**=configuration byte in Hex
DesfireFormatPICC	Desfire Format	DesfireFormatPICC	
DesfireSelectApp	Desfire Application	DesfireSelectApp,<AID>	**AID**=Application ID
DesfireCreateApp	Desfire Application	DesfireCreateApp,<AID>,<config>,<KeyCnt>	**AID**=Application ID **Config**= configuration byte in Hex **KeyCnt**=number of keys
DesfireDeleteApp	Desfire Application	DesfireDeleteApp,<AID>	**AID**=Application ID
DesfireCreateStdDataFile	Desfire File	DesfireCreateStdDataFile,<FID>,<Comm>,<Access>,<Size>	**FID**=FileID **Comm**=communication crypt mode **Access**=Key No.: RD,WR,RD+WR,Change **Size**=File size
DesfireCreateBackupFile	Desfire File	DesfireCreateBackupFile,<FID>,<Comm>,<Access>,<Size>	**FID**=FileID **Comm**= communication crypt mode **Access**=Schlüssel-Nr.: RD,WR,RD+WR,Change **Size**=File size
DesfireCreateValueFile	Desfire File	DesfireCreateValueFile,<FID>,<Comm>,<Access>,<Low>,<Up>,<Val>,<LC>	**FID**=FileID **Comm**= communication crypt mode **Access**=Key No.: RD,WR,RD+WR,Change **Low**= Unteres Limit **Up**=Oberes Limit **Val**=Wert **LC**=limited credit
DesfireCreateLinear RecordFile	Desfire File	DesfireCreateLinearRecordFile,<FID>,<Comm>,<Access>,<RecSize>,<MaxRec>	**FID**=FileID **Comm**= communication crypt mode **Access**=Key No.: RD,WR,RD+WR,Change **RecSize**=Record size **MaxRec**=max. Records

Befehl	Kategorie	Syntax	Parameter
DesfireCreateCyclic RecordFile	Desfire File	DesfireCreateCyclicRecordFile,<FID>,<Comm>, <Access>,<RecSize>,<MaxRec>	**FID**=File ID **Comm**= communication crypt mode **Access**=Key No.: RD,WR,RD+WR,Change **RecSize**=Record size **MaxRec**=max. Records
DesfireDeleteFile	Desfire File	DesfireDeleteFile,<FID>	**FID**=File ID
DesfireClearRecordFile	Desfire File	DesfireClearRecordFile,<FID>	**FID**=File ID
DesfireReadData DES,3KDES,AES	Desfire File	DesfireReadData,<FID>,<Off>,<Len>,<target> DesfireReadDataDES,<FID>,<Off>,<Len>,<target> DesfireReadData3KDES,<FID>,<Off>,<Len>,<target> DesfireReadDataAES,<FID>,<Off>,<Len>,<target>	**FID**=File ID **Off**=Offset **Len**=Number of Bytes **Target**=target filed
DesfireReadRecords	Desfire File	DesfireReadRecords,<FID>,<Off>,<Len>,<target>	**FID**=File ID **Off**=Offset **Len**=Number of bytes **Target**=target field
DesfireGetValue	Desfire File	DesfireGetValue,<FID>,<target>	**FID**=File ID **Target**=target field
DesfireWriteData DES,3KDES,AES	Desfire File	DesfireWriteData,<FID>,<Off>,<Len>,<source> DesfireWriteDataDES,<FID>,<Off>,<Len>,<source> DesfireWriteData3KDES,<FID>,<Off>,<Len>,<source> DesfireWriteDataAES,<FID>,<Off>,<Len>,<source>	**FID**=File ID **Off**=Offset **Len**=Number of bytes **Source**=Data source
DesfireWriteRecord	Desfire File	DesfireWriteRecord,<FID>,<Off>,<Len>,<source>	**FID**=File ID **Off**=Offset **Len**=Number of bytes **Source**=Data source
DesfireCredit	Desfire File	DesfireCredit,<FID>,<source>	**FID**=File ID **Source**=Data source
DesfireLimitedCredit	Desfire File	DesfireLimitedCredit,<FID>,<source>	**FID**=File ID **Source**=Data source
DesfireDebit	Desfire File	DesfireDebit,<FID>,<source>	**FID**=File ID **Source**=Data source
DesfireCommitTransaction	Desfire File	DesfireCommitTransaction	
DesfireAbortTransaction	Desfire File	DesfireAbortTransaction	
DesfireRandomUID	Desfire Config	DesfireRandomUID	
DesfireDefaultKey	Desfire Config	DesfireDefaultKey,<VER>,<KEY>	**VER**=New Default Key Version **Key**=New Default Key
DesfireDisableFormat	Desfire Config	DesfireDisableFormat	

Befehl	Kategorie	Syntax	Parameter
LegicSearchTxp	LEGIC	LegicSearchTxp	
LegicSearchSegment	LEGIC	LegicSearchSegment,<SN>,<ST>,<STR>	**SN**=Segment No. **ST**=Segment Type **STR**=Search string
LegicAddSegment	LEGIC	LegicAddSegment,<SN>,<ST>,<Config> *Config = OL,WRP,WRC,DataSize,RD*	**SN**=Segment No. **ST**=Segment Type **STR**=Search string **Config=**Configuration bytes Hex
LegicRemoveSegment	LEGIC	LegicRemoveSegment,<SN>	**SN**=Segment No.
LegicAddMasterData	LEGIC	LegicAddMasterData,<Config>	**Config=**Configuration bytes Hex
LegicDeleteMasterData	LEGIC	LegicDeleteMasterData	
LegicRead	LEGIC	LegicRead,<Adr>,<Len>,<target> LegicReadHex,<Adr>,<Len>,<target>	**Adr**=Segment address **Len**=Number of Bytes **Target**=target field
LegicWrite	LEGIC	LegicWrite,<crc>,<crc_adr>,<adr>,<source> LegicWriteHex,<crc>,<crc_adr>,<adr>,<source>	**crc=**CRC Type **crc_adr**=CRC memory address **adr**=memory address **Source**=Data source
LegicMakeCRC	LEGIC	LegicMakeCRC,<CRC_TYPE>,<CRC_FLAGS>, <CRC_ADR>,<ADR>,<DATA_LEN>	
LegicDisablePolling	LEGIC	LegicDisablePolling Disable Auto SeachTxp mode	
LegicEnablePolling	LEGIC	LegicEnablePolling Enable Auto SeachTxp mode	
Befehl	**Kategorie**	**Syntax**	**Parameter**
APDU	ISO 7816 APDU	APDU,<cmd>,<target>	**Cmd=**APDU command Hex **Target**=target field
Befehl	**Kategorie**	**Syntax**	**Parameter**
COPY	Tools	COPY,<source>,<target>,<start>,<len> Copies data from source to target	**Source**=data source **Target**=target field **Start**=Source start address **Len**=Number of char.
INSERT	Tools	INSERT,<source>,<target>,<pos> Adds data from a <source> at position <pos> in <Target>	**Source**=data source **Target**=target field **Pos**=Target position
WRITE	Tools	WRITE,<source>,<target>,<start>,<len> Writes data from <source> starting at position <start> with the length <len> in <target>. In this case, the existing data will be overwritten.	**Source**=Data source **Target**=target field **Start**=Source start address **Len**=Number of char.
ROTATE	Tools	ROTATE,<source> Rotates the data from left to right. e.g. 123456 will become 654321	**Source**=Data source
LSBTOMSB	Tools	LSBTOMSB,<source> Swaps the high and low byte e.g. 123456 will become 563412	**Source**=Data source
FILL_LEFT	Tools	FILL_LEFT,<char>,<len>,<var> Fills up <var> from the left with character <char> til total length <len> is reached	**Char**=Fill character **Len**=nominal length **Var**=%VAR1, %VAR2
FILL_RIGHT	Tools	FILL_RIGHT,<char>,<len>,<var> Fills up <var> from the right with character <char> til total length <len> is reached	**Char**=Fill character **Len**=nominal length **Var**=%VAR1, %VAR2
COMPARE	Tools	COMPARE,%VAR1,%VAR2,<Errortext> Compares the variable% VAR1, VAR2% In case of NOT equal, the script is finished, the error text is given in the control file.	**ErrorText=** freely definable error text

Befehl	Kategorie	Syntax	Parameter
BIN toHEX	Tools	HEX,<var> Contents of the variable <var> are converted into hex	**Var**=%VAR1, %VAR2
BIN toBCD	Tools	BCD, <var> Contents of the variables <var> are converted to BCD	**Var**=%VAR1, %VAR2
BIN to DEZ	Tools	DEZ,<var> Contents of the variables <var> are converted to decimal	**Var**=%VAR1, %VAR2
HEX to BIN	Tools	HEXBIN,<var> Contents of the variables <var> are converted from HEX to binary	**Var**=%VAR1, %VAR2
ADDMOD10	Tools	ADDMOD10,<source>,<var> Luhn check digit on <source>. <var> = <source> + Check Digit	**Source**=Data source **Var**=%VAR1, %VAR2
ADDXOR	Tools	ADDXOR,<var> XOR calculation on <var> byte 1 xor byte 2 xor byte n The result is appended to <var>	**Var**=%VAR1, %VAR2
ADDINVXOR	Tools	ADDXOR,<var> XOR calculation on <var> byte 1 xor byte 2 xor byte n The result is **inverted** and appended to <var>	**Var**=%VAR1, %VAR2
CRC	Tools	CRC,<source>,<target> 8-bit CRC calculation on <source>, the CRC byte is passed to <target>	**Source**=Data source **Target**=Target field
INC	Tools	INC,<source>,<step> Increments <source> with step size <step>	**Source**=Data source **Step**=Step size
Befehl	**Kategorie**	**Syntax**	**Parameter**
PRINTCMD	Printer	PRINTCMD,<cmd> Es wird die ESC-Sequenz ESC+<cmd>+CR an den Drucker gesendet.	**Cmd**=Printer command
PRINTCARD	Printer	Druckvorgang wird gestartet	
KEYB	Input	KEYB,<target>,<name>[,<mask>]	**Target**=Target field **Name**=Display text **Mask**= optional input mask Defined in the Annex **mask definitions**

Chip-Manager

Befehl	Kategorie	Syntax	Parameter
SQL	SQL	SQL,<Statement> Executes a SQL statement. **Warning:** The current database cursor is changed.	**Statement**= Any SQL statement
SetDBField	SQL	SetDBField,<field>,<value> Writing back to the database in the current record	**Field**=Fieldname (without @ sign) **Value**=Data
Befehl	**Kategorie**	**Syntax**	**Parameter**
Refresh	Smartcard	Refresh chip tree view in chip manager window	
Wait	Script	wait,<time> The execution of the next script command is delayed by <time> ms	**Time**=Wait Time in ms
Exit	Script	The execution of the script is terminated	
ErrorHaltOn	Script	In case of an error, the execution of the script is stopped (default)	
ErrorHaltOff	Script	The execution of the script will be continued in casse of an error.	

Card-Designer

The card designer is used to create the card layout and can contain a unlimited number of textobjects, imageobjects and barcodeobjects.

The individual objects can get their data from any source.

This may be data from the database, data read from the chip (e.g., the UID of an RFID card) calculated, or composite data from the script (for example, a checksum), camera (such as an employee photograph) or scanned images (for example, a signature) or manual keystrokes.

The following describes the individual control elements to create a card layout.

Tools in the Card Designer

	Release of the editing function, may be password protected
	Mouse pointer to select objects
	Turns Grid on or off
	Create new text object
	Create new Image object
	Create new barcode object
	Create new shape object (Rectangle, RoundRect, Circle, Elipse)
	Delete the currently selected object
	Opens the camera dialog, to retrieve a new camera image
	Opens the scanner dialog, to retrieve a new scan
	Opens the Print dialog

To create a new layout or to edit an existing one, the tool palette has to be turned on with the button.

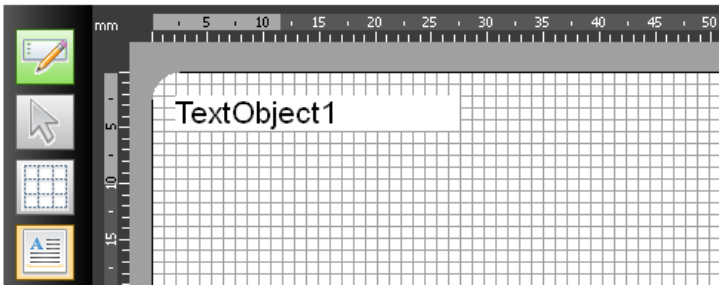
In case of a system password is set, it has to be entered to enable the edit function.
A further click on the button locks the edit function again. This is to prevent an accidental change to the layout.

The tools "get camera image" and "pick scanner image" are only active
In case of an camera device or scanner device is selected in the settings and under "File Locations" an image directory is specified.

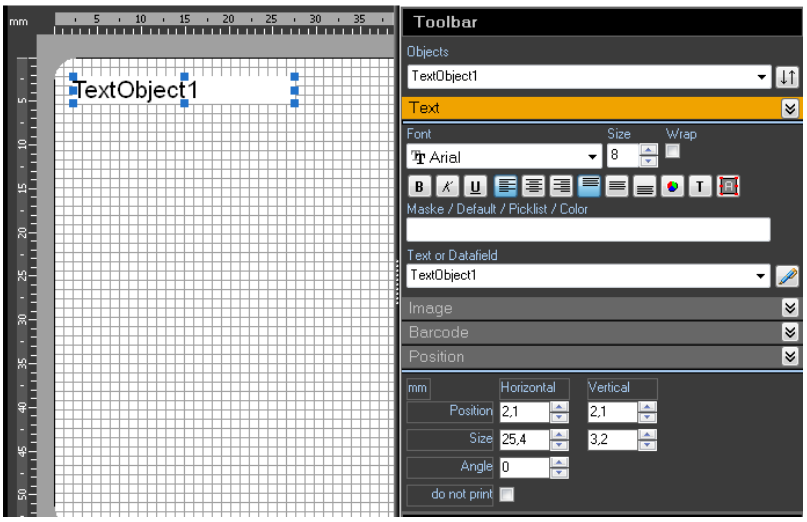
When the grid function is activated, all objects are aligned on a grid,
this facilitates the arrangement and orientation of objects.

Textobject

To create a text object, first click on the  button , then on a free position in the card layout. The text object is placed with preset properties (font, color, text) on the card layout.



To edit the text object, click on the text object. The text object is selected and the corresponding tool palette is displayed.



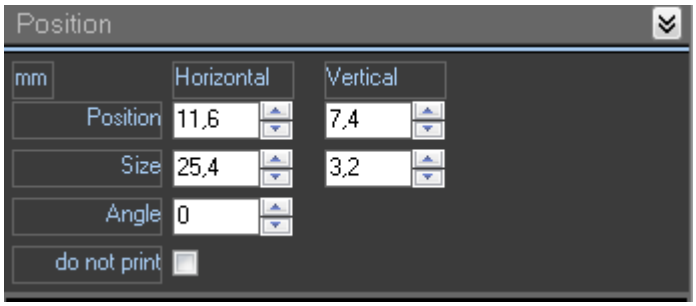
Editierbare Eigenschaften eines Textobjektes:

Property	Bemerkung
Font	Available fonts
Größe	Font size in points
Wrap	Automatic line break, can be fourced with the | (pipe) character
B Bold	Bold font
Italic	Italic font
U Underline	Underline font
Left Align	Left aligned
Center	Center aligned
Right Align	Right aligned
Color	Font color
T Transparent	Transparent, the background is visible in the text object.

Card-Designer

Textobjekt

Eigenschaft	Bemerkung
Autosize	The object is resized to fit the text content
	Maske / Default / Picklist / Color These properties can be specified in combination. The tilde "~" character is used as a separator between the definitions. Example: *Picklist=Red;Green;Blue~Default=Green*
Mask	Text template for manual input. **Mask definition:** Maske=<maske> (see mask definition in appendix) *Example: Maske=99/99/9999*
Default	Default Text by manual input **Default text definition:** Default=<default text> *Example: Default=ABCDEFG*
Picklist	Picklist by manual input **Picklist Definition:** Picklist=<Entry 1>;<Entry 2>;< Entry n>; *Example: Picklist=Red;Green;Blue*
Color	Font color, can also be defined via database field. **Color Definition:** Color=<RGB value> *Example fix: Color=00FF00* *Example Databasefield: Color=@RGB*
Text	**Printable text** Can be fixed text, variable or a database field. *Example:* *Fixtext: Max Mustermann* *Variable: %UID* *Databasefield: @Name*
	Position
Horizontal	In [mm] from the left margin
Vertical	In [mm] from the top
Horizontal size	Width of the text object in [mm]
Vertical size	Height of the text object in [mm]
Angle	Text angle in degrees 0..359
Do not print	Exclude from printing



Card-Designer

Appendix A

Example Desfire EV1

Create an application with AID = 8034F3, AES crypto method, create a data file, write data and change keys.

Keysafe Entries:

MasterKey=01020304050607080910111213141516
Key0=0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a
Key1=1a1a1a1a1a1a1a1a1a1a1a1a1a1a1a1a
Key2=2a2a2a2a2a2a2a2a2a2a2a2a2a2a2a2a

Script:

```
;*****************************************************************
;Desfire EV1 Application F33480 with 3 AES Keys
;Customer: Demo
;Version : 1.0
;Stand   : 05.2013
;*****************************************************************
ErrorHaltOff
DesfireAuth,000000,DefaultDES,00
DesfireChangeKey,00,00,80,MasterKey
DesfireAuthAES,000000,DefaultAES,00
DesfireChangeKeyAES,80,00,MasterKey
ErrorHaltOn
DesfireAuthAES,000000,MasterKey,00
ErrorHaltOff
DesfireDeleteApp,8034F3
ErrorHaltOn
DesfireCreateApp,8034F3,0B,83
DesfireAuthAES,8034F3,DefaultAES,00
DesfireCreateStdDataFile,00,03,2012,100000
DesfireAuthAES,8034F3,DefaultAES,02
;
; Data here=00010203040506070809101112131415
; for Database or Parameter use -> "@fieldname"; Parameter-> "%parameter"
;
DesfireWriteDataAES,00,000000,100000,00010203040506070809101112131415
DesfireAuthAES,8034F3,DefaultAES,00
DesfireChangeKeyAES,00,00,Key0
DesfireAuthAES,8034F3,Key0,00
DesfireChangeKeyExAES,01,00,DefaultAES,Key1
DesfireAuthAES,8034F3,Key0,00
DesfireChangeKeyExAES,02,00,DefaultAES,Key2
;
;
;*****************************************************************
;Test commands
;*****************************************************************
DesfireAuthAES,000000,MasterKey,00
DesfireAuthAES,8034F3,Key0,00
DesfireAuthAES,8034F3,Key1,01
DesfireReadDataAES,00,000000,100000,%EDITOR
DesfireAuthAES,8034F3,Key2,02
Refresh
```

Example

Appendix A

Mifare Classic

Sector auth, read and write blocks, change access bits

```
;*****************************************************************
;Mifare Classic
;Customer: Demo
;Version : 1.0
;Stand   : 05.2013
;*****************************************************************
;Auth Sector 0
MifareAuth,KeyA,0,MIFARESECTOR0KEYA
;Save UID in Database Fieldname <UID>
MifareReadUIDHex,@UID
;Write Mifare Sector 0, Block 1 from Database Fieldname <Data>
MifareWriteBlock,0,1,@Data
;Write KeyA,KeyB and Accessbits
;KeyA,KeyB from Keysafe
;Accessbits fix =FF078069
MifareWriteKeyAccess,0,KeyA,FF078069,KeyB
```

Example

Appendix A

Mifare Classic

Create a Application Directory (MAD)

```
;*****************************************************************
;Mifare Classic MAD
;Customer: Demo
;Version : 1.0
;Stand   : 05.2013
;*****************************************************************
;
;1. Read existing entry from  Sektor 0 Block 1 & Block 2
;
MifareAuth,KeyA,0,MIFARESECTOR0KEYA
MifareReadBlock,0,1,%EDITOR
MifareReadBlock,0,2,%EDITOR+
;
;write MAD Entry 2D48 for Sector 4 (Byte 8 and 9)
;
COPY,2D48,%VAR1,1,4
HEXBIN,%VAR1
WRITE,%VAR1,%EDITOR,9,2
;
;CRC calculation over address 1-32
;
COPY,%EDITOR,%VAR2,2,31
CRC,%VAR2,%VAR1
;
;write CRC on address 0
;
WRITE,%VAR1,%EDITOR,1,1
;
;write block 1
;
COPY,%EDITOR,%VAR1,1,16
HEX,%VAR1
MifareWriteBlockHex,0,1,%VAR1
;
;write block 2
;
COPY,%EDITOR,%VAR2,17,16
HEX,%VAR2
MifareWriteBlockHex,0,2,%VAR2
```

Example

Appendix A

Database Examples (Plain Text)

Plain text file as a database in directory „C:\temp“

The text file named "data.txt" containing records:

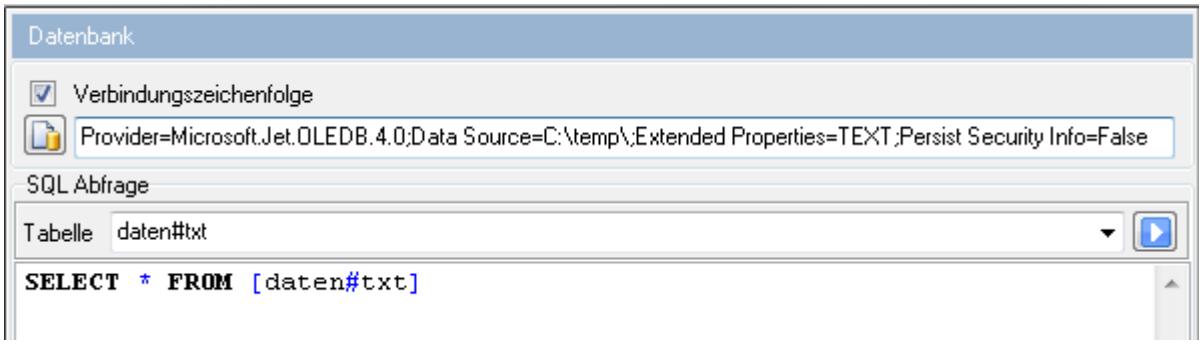
```
Vorname;Name
Max;Mustermann
Fritz;Schubert
```

The "schema.ini" file in the same directory contains the entries:

```
[daten.txt]
ColNameHeader=True
Format=Delimited(;)
```

Connection string to use:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\temp\;Extended
Properties=TEXT;Persist Security Info=False
```



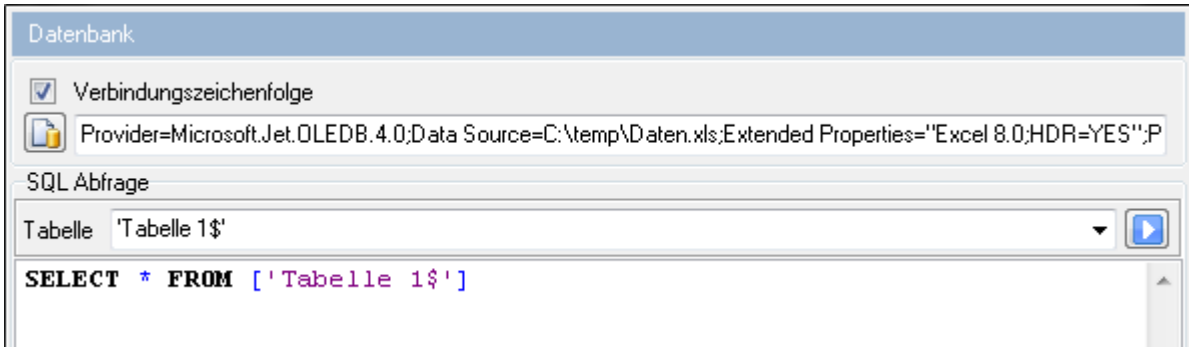
Example

Appendix A

Database Examples (Excel table)

To use the excel file “data.xls” from directory “c:\temp” use the following connection string:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\temp\Daten.xls;Extended
Properties="Excel 8.0;HDR=YES";Persist Security Info=False
```

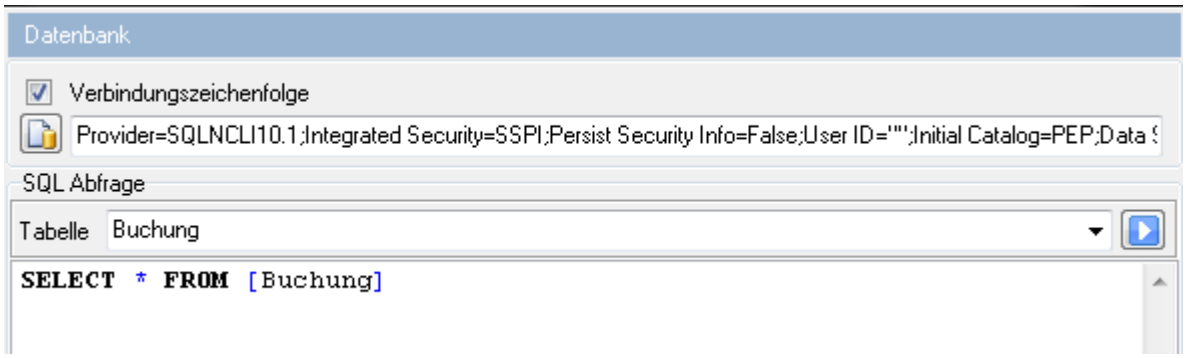


MS-SQL Server connection example

- MS-SQL Servername: (local)\SQLExpress
- Initial Catalog: PEP
- windows integrated security

Connection string to use here:

```
Provider=SQLNCLI10.1;Integrated Security=SSPI;Persist Security
Info=False;User ID="";Initial Catalog=PEP;Data
Source=(local)\SQLExpress;Initial File Name="";Server SPN=""
```



Example

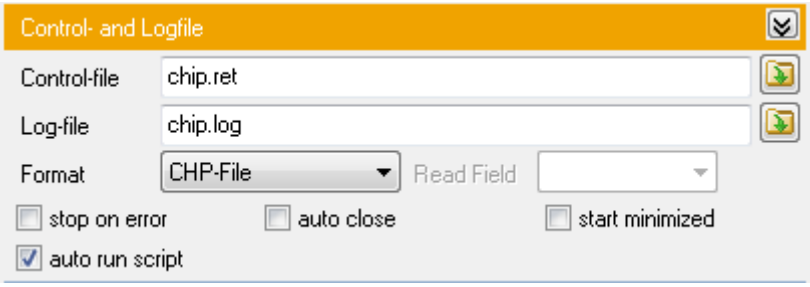
Appendix A

Database Example:

Write UIDs into the database

1. Task:
Save the UID of each card in the database.
The user places the card manually on the reader:

The option “auto run script” must be checked.



The script starts automatically, when the card is recognized by the reader.

Use the following script to append the new data with field name „UID“. Variable %UID contains the UID of the card.

```
SQL,insert into Data#txt (UID) VALUES ('%UID')
SQL,select * from Daten#txt
```

The database here is a simple text file (data.txt) with the data field "UID".

Example

2. Task

Code and print RFID cards from an existing database and write back the UID. Only records that have not been processed yet, should be selected. Print order should be sorted by field “UserNo“. The database contains the fields for the personalization of the card and also the fields „UID“ and „EXPORT“. The UID of the card should be saved in the data field „UID“. After processing the card, the field „EXPORT“ should be set to „false“, to avoid accessing the card a second time.

The SQL command for selecting the data might look now as follows:

SQL:

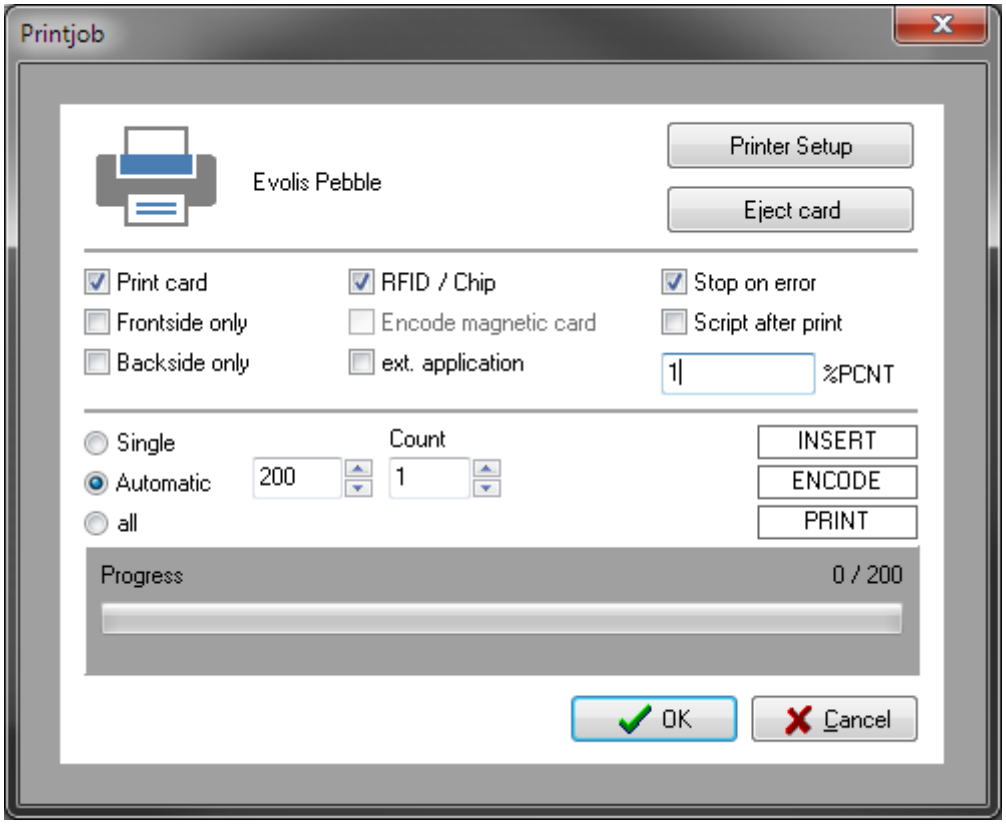
```
select * from data
where export=true
order by UserNo
```

Script:

```
; ... code commands for RFID here
;
; write back the UID, set Export to false
SetDBField,UID,%UID
SetDBField,Export,false
```

Open the Print Dialog:

The script will start automatically with each card that is recognized by the reader. In the print dialog the option "RFID" must be selected.



Example

Masks definitions

Zeichen	Bedeutung in der Maske
!	If a ! character appears in the mask, optional characters are represented in the text as leading blanks. If a ! character is not present, optional characters are represented in the text as trailing blanks.
>	if a > character appears in the mask, all characters that follow are in uppercase until the end of the mask or until a < character is encountered.
<	If a < character appears in the mask, all characters that follow are in lowercase until the end of the mask or until a > character is encountered.
<>	If these two characters appear together in a mask, no case checking is done and the data is formatted with the case the user uses to enter the data.
\	The character that follows a \ character is a literal character. Use this character to use any of the mask special characters as a literal in the data.
L	The L character requires an alphabetic character only in this position. For the US, this is A-Z, a-z.
l (lower L)	The l character permits only an alphabetic character in this position, but doesn't require it.
A	The A character requires an alphanumeric character only in this position. For the US, this is A-Z, a-z, 0-9.
a	The a character permits an alphanumeric character in this position, but doesn't require it.
C	The C character requires an arbitrary character in this position.
c	The c character permits an arbitrary character in this position, but doesn't require it.
0	The 0 character requires a numeric character only in this position.

Masks definitions

9	The 9 character permits a numeric character in this position, but doesn't require it.
#	The # character permits a numeric character or a plus or minus sign in this position, but doesn't require it..
:	The : character is used to separate hours, minutes, and seconds in times. If the character that separates hours, minutes, and seconds is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
/	The / character is used to separate months, days, and years in dates. If the character that separates months, days, and years is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
;	The ; character is used to separate the three fields of the mask.
_	The _ character automatically inserts spaces into the text. When the user enters characters in the field, the cursor skips the _ character.

Mask examples:

Usage	Mask	Results
Date	Maske=99/99/9999	18.06.2013
Phone No	Maske=(99999) 999999	(02162) 22964
VAT ID	Maske=>LL 000000000	DE 123456789
Bank Account	Maske=B\LZ 000 000 00	BLZ 312 500 00