# Smart DB®

# User Manual

# TABLE OF CONTENTS

# Figure

# 1 Introduction

SmartDesign® is the program which focuses on designing a card so you can print card data one by one. However when the data is huge, it takes too much time to print every data one by one.

For this reason, SmartDB® is developed and you can print several data at once using this program. This program is able to be linked with DBMS via ODBC function and have many useful functions such as “Multi-language Support” and “Print History”.

## 1.1 System Requirements

SmartDB can be operated with only IDP Corp,.Ltd’s SMART card printers.

The required software and hardware configuration for your computer system to execute SmartDB is as below.

- Windows XP, Vista, 7
- Pentium 1G Hz
- 256MB RAM

## 1.2 Program Structure

If you design a card in SmartDesign and save in your PC, the file will be saved as “*.CSD”. In this CSD file, there is “Field” section and it is for linking texts, images and barcodes to SmartDB so that SmartDB program can modify the items.

You don’t need to change data case by case in SmartDesign but you can modify different data using SmartDB at once. If you edit CSD “Field” in SmartDB, it will be the same effect as you modified the CSD file. But this changed data is valid until closing CSD file in SmartDB. So if you close CSD file or exit SmartDB program, the original design of CSD data would not be changed.

You can design and modify large numbers of cards through SmartDB quickly and safely.

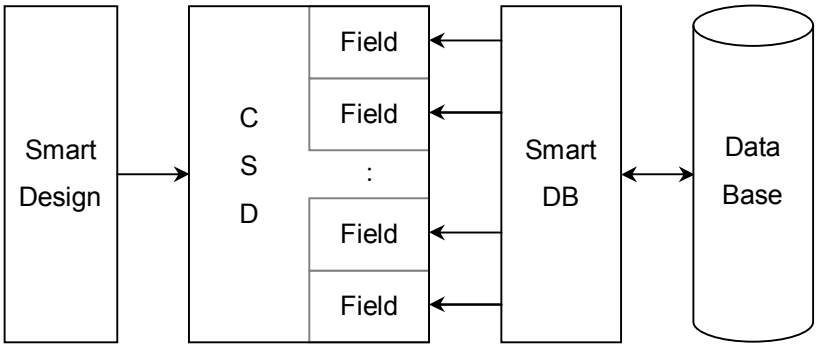


**Figure 1 SmartDB Operation Structure.**

# 2 Program Start

Usage of this program will be introduced in this chapter briefly.

## 2.1 Execute Program

From Window's Start menu

Click **Start** Menu > **All Programs** > **Smart** > **SmartDB**

### From the shortcut icon created on Desktop background

Double-click the **icon** which is created on Desktop background after installation of program.


When you run the program, it will be displayed as Fig. 2.

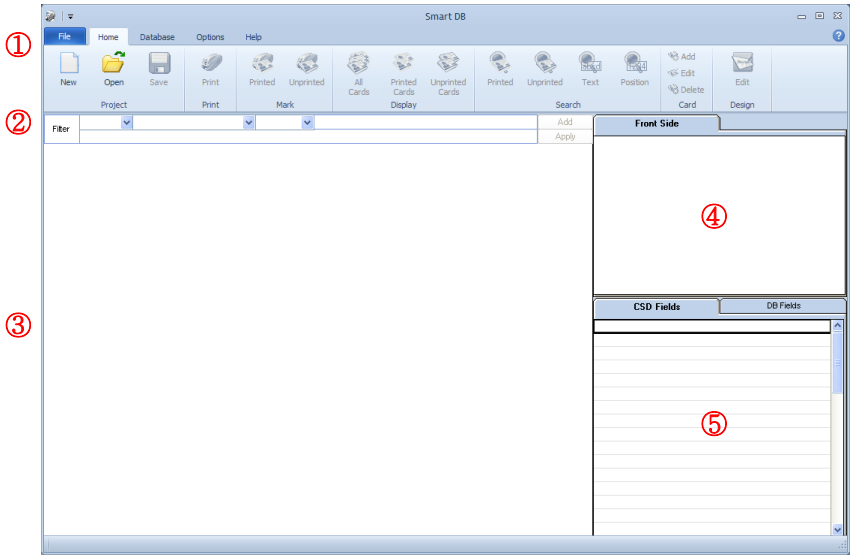


**Figure 2 SmartDB Window**

① Ribbon bar

Please refer to **chapter 4.Ribbon bar**

② Filter bar

Please refer to chapter **3.11 Use the Conditional Formula**

③ Data

It shows data which is connected to CSD field and DB field.

④ Preview

It shows preview image. If a document has single side, it shows only front side. If a document has both sides, it shows another tap for the back of the document. When select the tap, it will present preview image of back side.

⑤ CSD fields / DB fields

When you select a CSD field tap, it shows field information of the CSD file opened. When you select DB field tap, it shows recorded information of Database.

## 2.2 Create New Project

To connect a CSD file and Database, create new project.

① Create a CSD file which has fields section using SmartDesign.
For more information, please refer to user manual of SmartDesign.

② Click **File** tab and **New** button, or Click  **New** button on home tab.

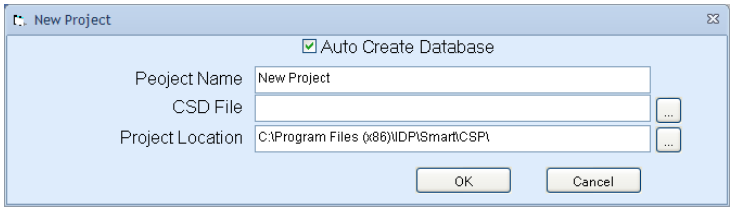


**Figure 3 New Project**

③ Click upper **"…"** button and open the saved CSD file. When you select a CSD file, Project Name will be changed as the CSD file name.

④ Click lower **"..."** button and select the path where the new project is saved in.

⑤ "Auto Create Database" is the function which creates and connects database automatically according to CSD fields when you make new project.
Select CSD file and mark **Auto Create Database** Check box. Click **OK**.

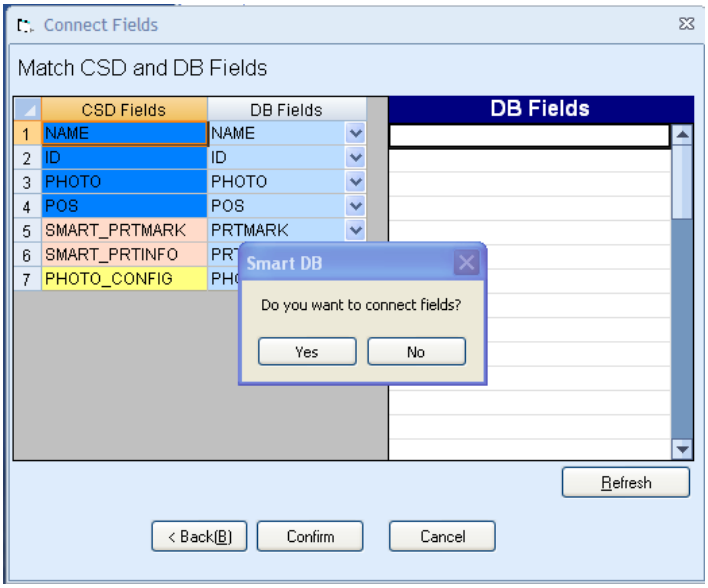


**Figure 4 Connect Fields**

⑥ Above image shows how CSD fields and Database(DB) are connected. Database is created and connected automatically according to CSD fields because “Auto Create Database” is activated.

In “CSD Fields” column, the blue color section is “Field Name” of the CSD file. Pink color section is print record fields, and these sections are used in SmartDB. Yellow color section is Image field. In case there are connected CSD field images, it will be created automatically.

When “Do you want to connect fields?” window appears, select **Yes** to connect CSD fields with DB fields.

After CSD fields are connected to DB fields, the program will be changed as Fig. 5.

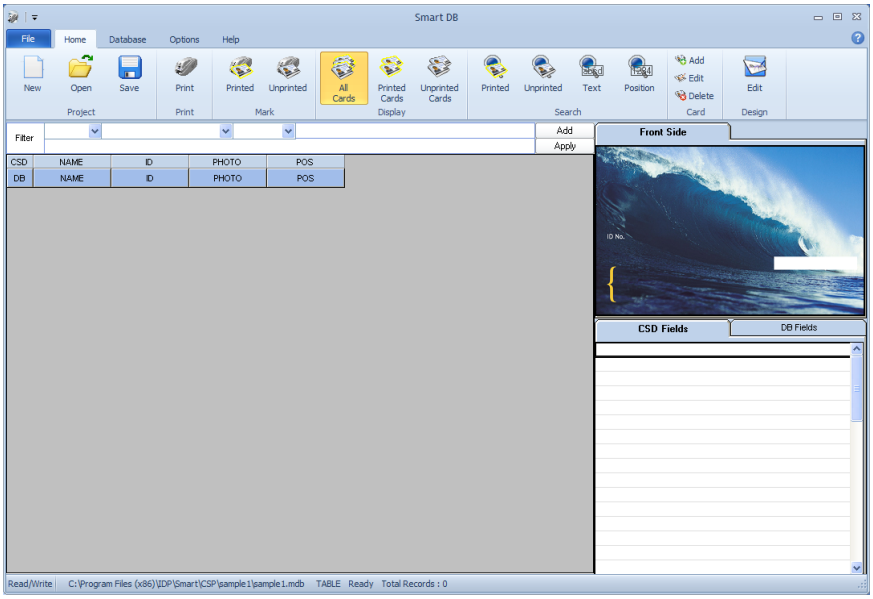


**Figure 5 After DB connection**

CSD preview will be shown on the right side and CSD fields and DB fields will be displayed.

⑦ From now on, you can input and edit data.

SmartDB manages projects as the unit of directories. So a directory is created as “Project Name” on appointed “Project Location” and all files will be copied in the folder.

## 2.3 Input Data

To input data, click **Add** in **Home** tab or **Database** tab.

Input data in **Value** column.

Select an image after click **“…”** button when you use image field. To delete or change an image, right-click **Value** column. When you select **Select Image** in context menu, you can select new image. And also you can delete an image using **Remove Image** in context menu.

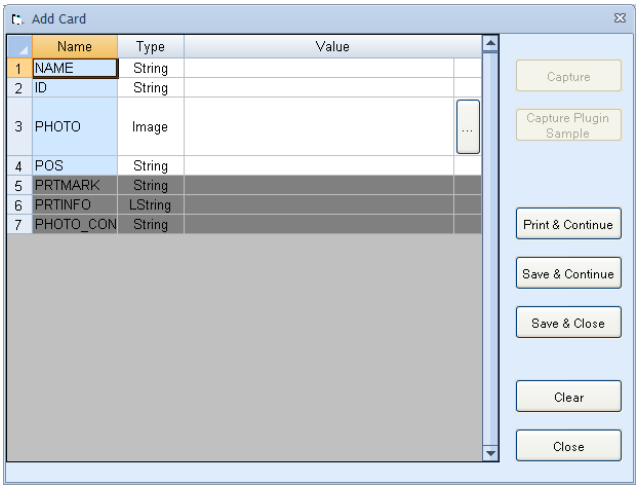


**Figure 6 Add Data**

If you click Image Field when **USB Camera** is connected to PC, Capture button will be activated on the right side. Click activated **Capture** button, and then camera view window appears like Fig. 7.

Click **Capture** button in the window, then program will save the image as Fig. 8.

**Figure 7 Capture**

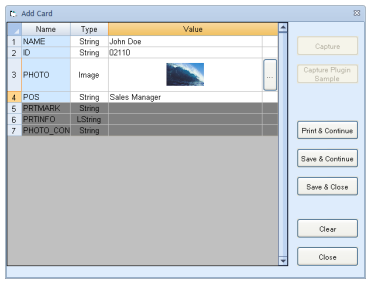

**Figure 8 Add Figure Image**

If **Plugin** is installed in SmartDB program folder, the buttons will be displayed as plugin names like figure 9. Click a button and execute plugin so that you can import an image. Please refer to Appendix – Plugin part.

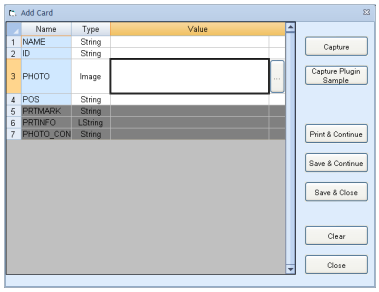

**Figure 9 Plugin Data Added**

To print the data in Add Card window, Click **Print & Continue** button.

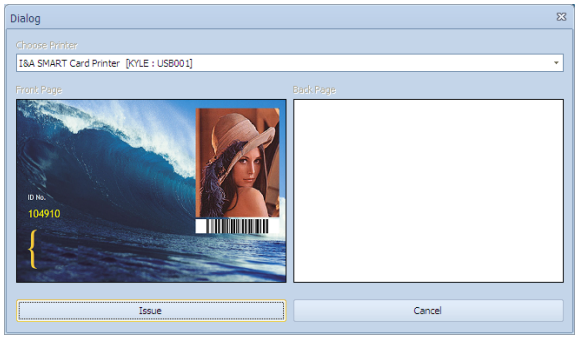


**Figure 10 Print & Continue**

Select a **printer** on upper combo box and click **Issue** button to print. After printing process, preview window will be closed automatically and the data of printed content will be added in Database automatically.

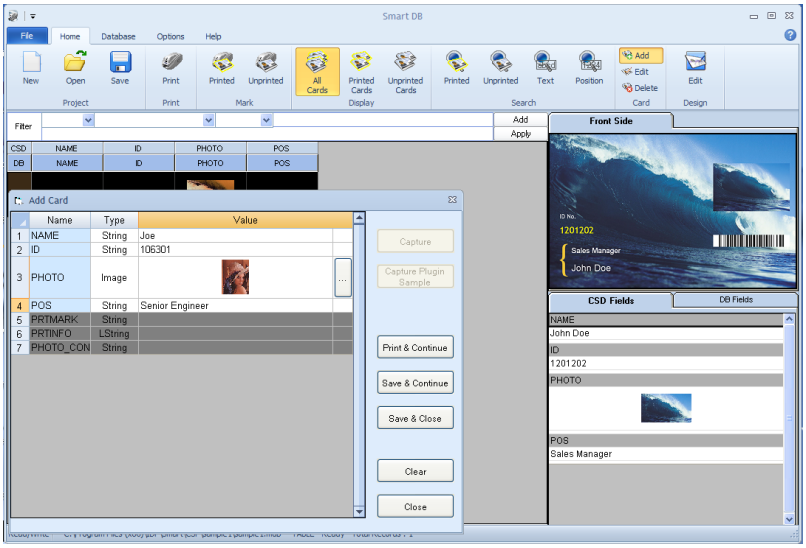


**Figure** 11 **Add Data**

To save data and keep the current window, click **Save & Close** button.

To save data and close the current window, click **Save & Continue** button.

To clear whole data in current window, click **Clear** button.

To close the window without save, click **Close** button.

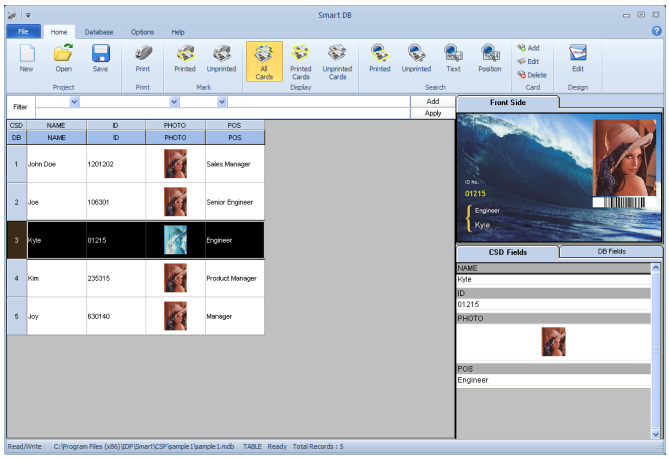


**Figure 12 After Adding Data**

If you add a card, the new data is updated on the screen as Figure 12.

Selected data is updated through data field and it is displayed in preview section.

## 2.4 Edit Data

There are two ways to edit data user entered.

Click Edit in Home tab or Database tab. Then it will be displayed as Fig. 13.

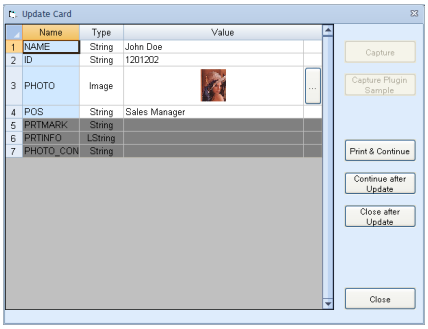


**Figure 13 Editing Data**

Double-click the **column** to edit. Then input data directly to the column as Fig. 14.

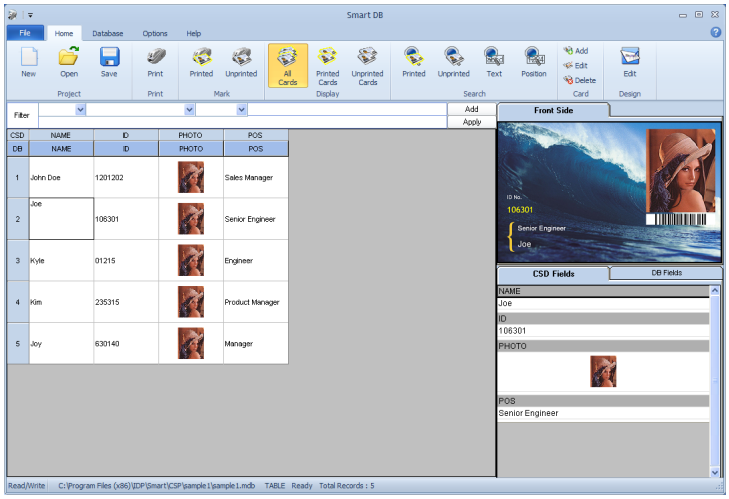


**Figure 14 Direct Input in Column**

## 2.5 Print

Connect a card printer to PC before printing.

① To print, select the data as Fig.15 and click **Print** button in **Home** tab or **File** tab.

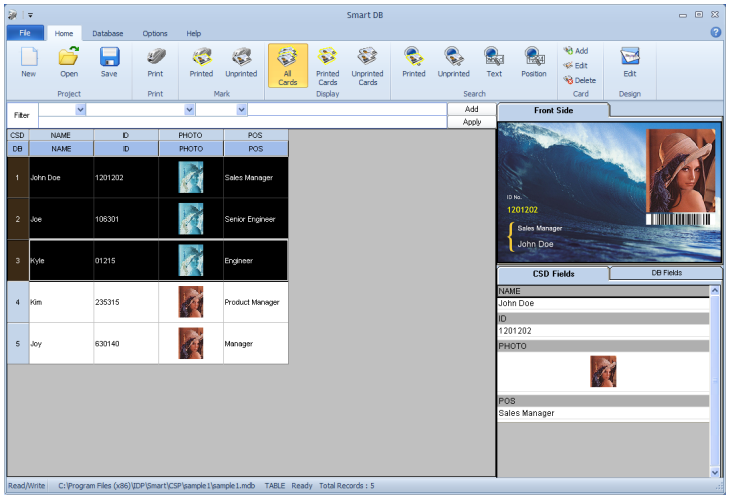


**Figure 15 Select Data for Printing**

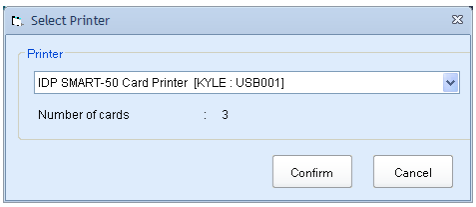


**Figure 16 Select Printer**

② Figure 16 shows available printer which is connected PC or network
The printers which have a word “USB” on the lists are connected to PC directly.
The printers which names start with IP address are connected to network.
Choose a printer and click **Confirm** button.

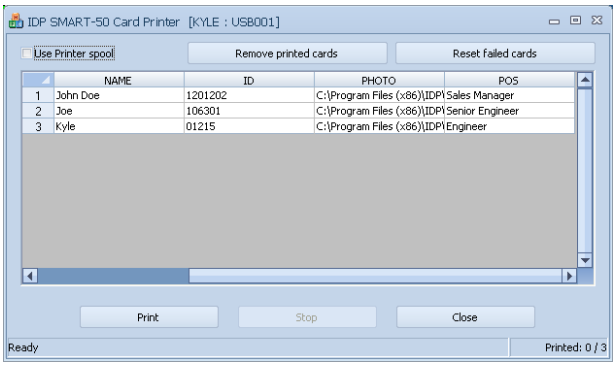


**Figure 17 Print Spooler**

③ Figure 17 shows printer spooler window.
Click **Print** button, then all data on the lists will be printed.

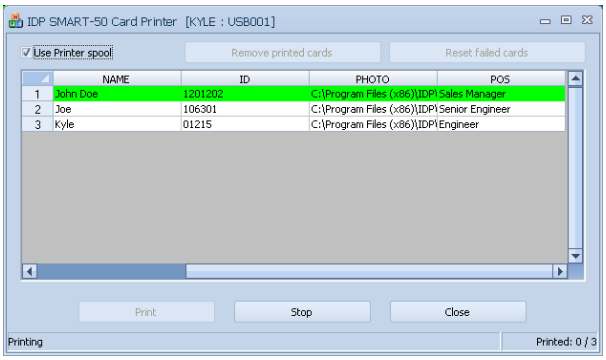


**Figure 18 Printing**

The white background means the data which is ready to print. The green color means under printing. After finish the printing, the color will be changed to yellow. If there is an error during the printing, it will be changed to red color. Even though it is under printing, you can add more data to print.

④ When printing is done, click **Close** to return to main program.
Printing results will be applied to data list as Fig. 19.

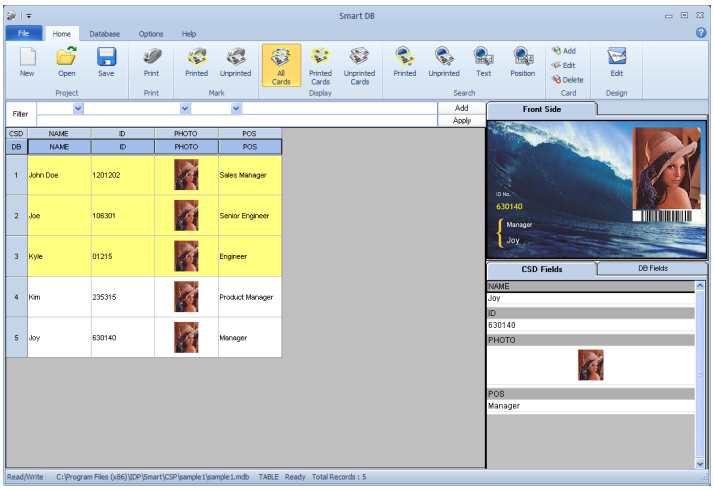


**Figure 19 After Printing**

# 3 Main Function

The main function will be explained in this chapter.

## 3.1 Create New Project

To connect a CSD file and Database, create new project.

⑧ Create a CSD file which has fields section using SmartDesign.
For more information, please refer to user manual of SmartDesign.

⑨ Click **File** tab and **New** button, or Click  **New** button on home tab.

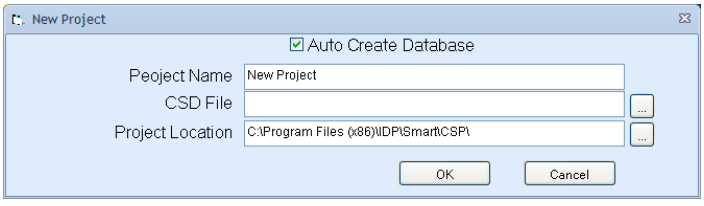


**Figure 20 New Project**

⑩ Click upper **"…"** button and open the saved CSD file. When you select a CSD file, Project Name will be changed as the CSD file name.

⑪ Click lower **"..."** button and select the path where the new project is saved in.

⑫ “Auto Create Database” is the function which creates and connects database automatically according to CSD fields when you make new project.
If it is connected with the existing database, you have to remove the mark in this section.

⑬ Project file will be generated as *.csp in the folder under the directory which set on “Project Location”.

## 3.2 Create / Modify Database

### 3.2.1 Auto Create Database

If “Auto Create Database” option is activated in chapter 3.1, MDB database will be created and connected automatically as CSD file’s fields.

In “CSD Fields” column, the blue color section is field name of the CSD file. Pink color section is the print record field, and these are used in SmartDB. Yellow color section is Image field. If there is connected image in SmartDB, it will add “_CONFIG” in the last part of the field name and will be created.

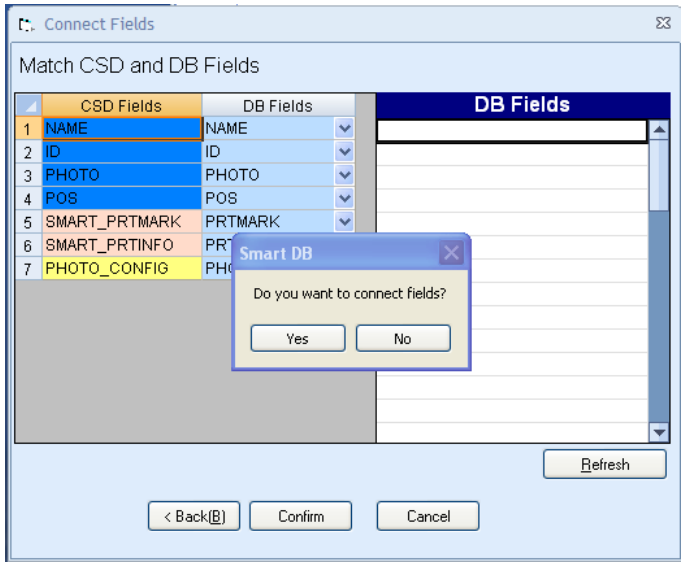


**Figure 21 Connect Fields**

If “Do you want to connect field?” box appears, click **Yes** to connect CSD field name and DB field name automatically.

### 3.2.2 Manual Create Database

If “Auto Create Database” option is not activated at chapter 3.1, it needs to

create database manually or connect to current ODBC.

Click **Database** tab and **Connect MDB** button. Then “MDB Management” window will be displayed as Fig. 22.

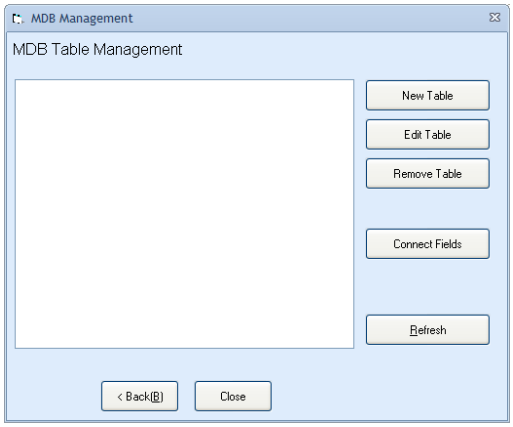


**Figure 22 MDB Management**

3.2.2.1 Auto Create MDB

① Click **Automatic Create MDB** button, then “Automatic Create MDB” windows as Fig. 23 will be displayed.

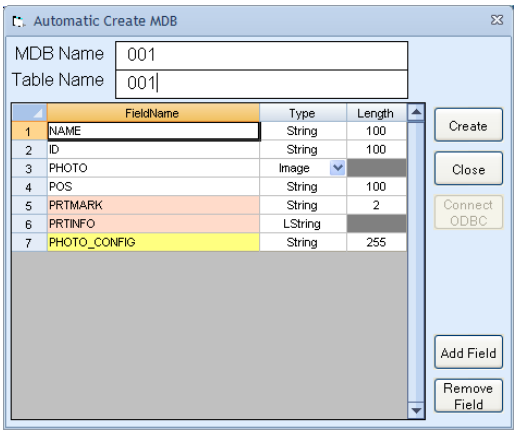


**Figure 23 Auto Create MDB**

“MDB Name” means that created MDB file name and you can change it. “Table Name” means that created table name in MDB file and you can also change it.

DB table structure will be created according to CSD file field structure.

② Click **Create** button, DB will be created as setting. It will be connected with CSD file’s fields automatically as below.

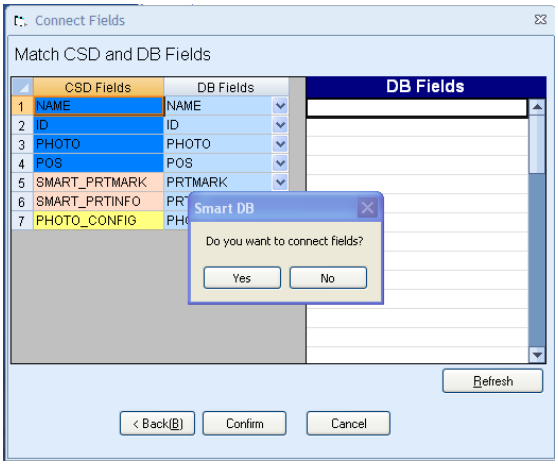


**Figure 24 Connect Fields**

③ Click **Yes**, then CSD file fields and DB fields will be linked.
Click **No**, then fields will not be synchronized.

3.2.2.1 Create MDB

① Click **Create MDB** button, then “Input MDB Name” window will be displayed.

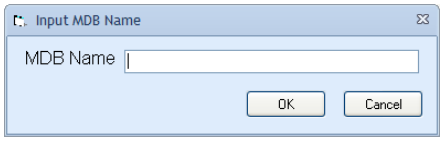


**Figure 25 Input MDB Name**

② Input MDB **name** and click **OK** button, “MDB Table Management” window will displayed as Fig. 26.

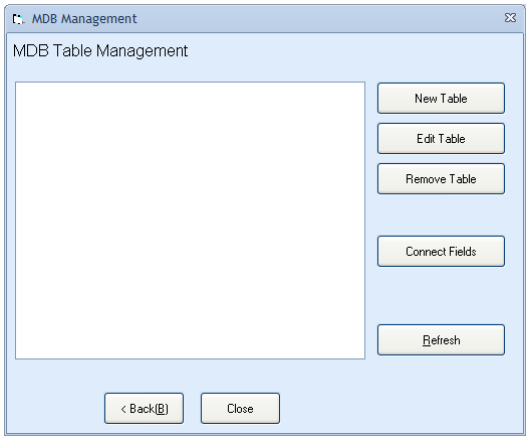

**Figure 26 MDB Table Management**

Selected table structure of MDB file will be displayed on the left list and you can create, edit and remove the table.

③ To create table, click **New Table** button.

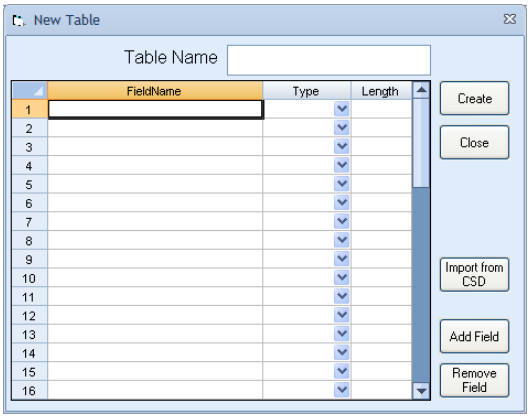


**Figure 27 Create New Table**

Input a **name** in “Table name” section.

To input/edit the Field name, double-click to the empty Fieldname index or data.

You can set the field type on Type column.

You can set the maximum data size of the field on Length column.

If you click **Import from CSD** button, table field will be set automatically according to CSD file field structure as below.

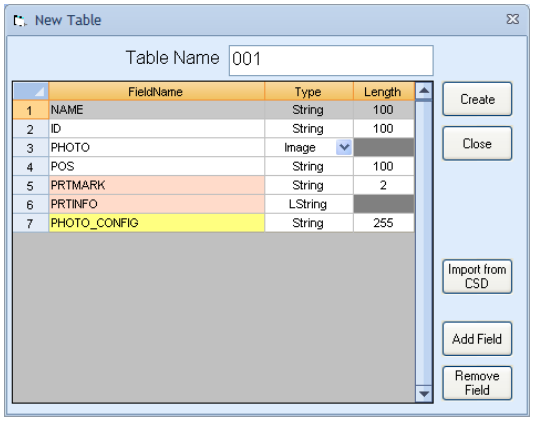


**Figure 28 Create New Table – Import from CSD**

When you click **Create** button, a question message will pop up.

To create it, click **OK**.

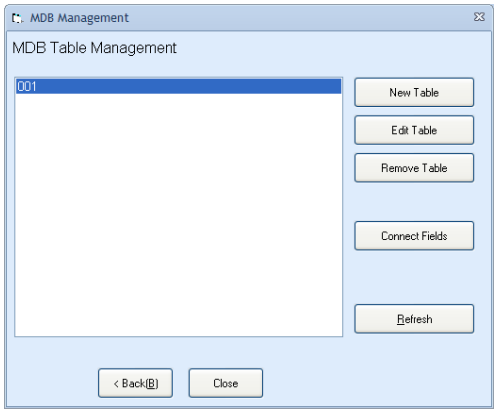


**Figure 29 MDB Management**

Figure 29 is created table.

④ To change selected table structure, click **Edit Table** button

To remove the selected table, click **Remove Table** button.

Click **Connect Fields** button, then you can connect the fields as Fig. 30.

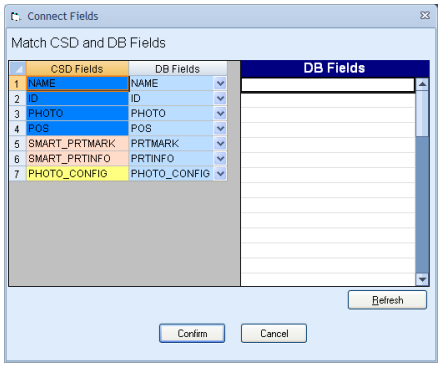


**Figure 30 Connect Fields**

## 3.3 Connect ODBC

To connect variety sorts of DBMS, SmartDB supports ODBC.

All databases supported by ODBC are available.

① Click **Database** tab and **Connect ODBC** button. Then “Select ODBC Data Source” window will be displayed as Fig. 31.

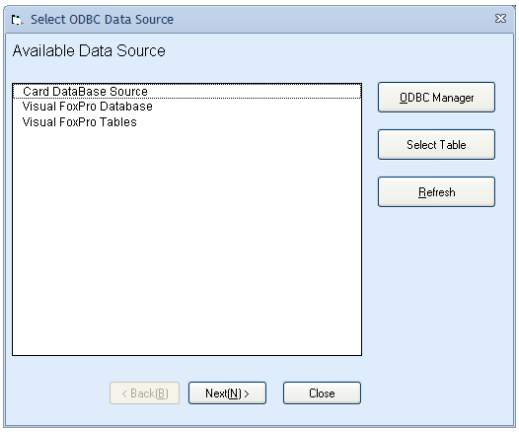


**Figure 31 Select ODBC Data Source**

On the left side, you can see the data format which is available to connect.
If you click **ODBC Manager** button, then “ODBC Data Source Administrator” window will be displayed as Fig. 32.

For more information, please refer to the related materials.

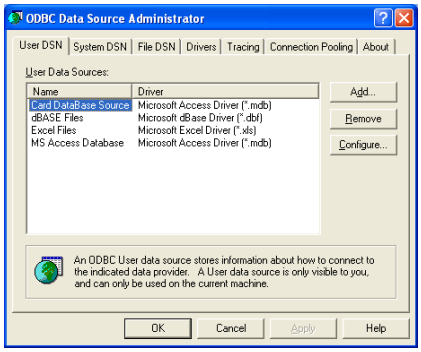


**Figure 32 ODBC Data Source Administrator**

If you click **Select Table** button after select one source, you can see the different window as per each source.
SMART Test data source is created from ODBC original administrator. Click **Select Table** button, then “Available Tables and Views” window will be displayed and you can select table or view.

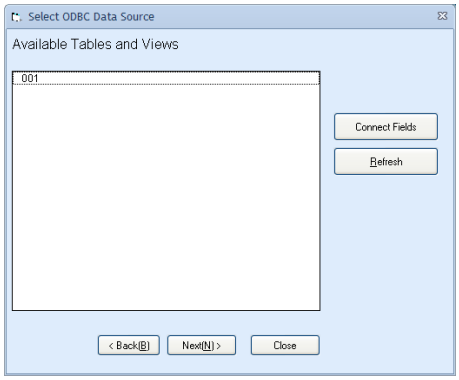


**Figure 33 Select ODBC Data Source**

Click **Connect Fields** button after selecting a table, then Fig. 34 will be displayed.

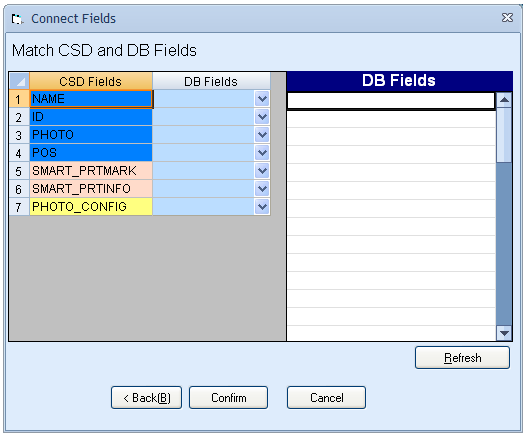


**Figure 34 Connect Fields**

CSD fields and DB fields can be connected.

While selecting **MS Access Database**, click **Select Table** button then you can select MDB file as Fig. 35

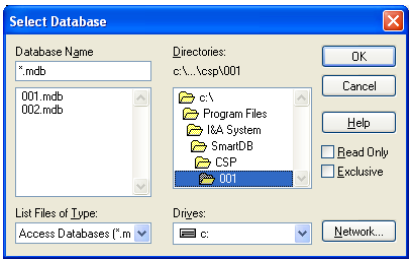


**Figure 35 Select MDB**

If select a MDB file and click **OK** button then table or window will be displayed.

Thereafter it will run through same process as described before.

## 3.4 Save Project

Information of CSD file and Database connection will be saved in the project file (*.csp). If it cannot be recorded, you need to proceed with many steps as above.

To avoid the inconvenience, if connection setting is finished, you need to save project file.

To save project, click **Save** button in **Home** tab or **File** tab.


**Figure 36 Save Project**

## 3.5 Open Project

To open project, click **Open** button in **Home** tab or **File** tab.

When you import a file, CSD fields automatically will be connected database fields.

## 3.6 Input Data

To input data, click **Add** in **Home** tab or **Database** tab.

Input data in **Value** column.

Select an image after click **"…"** button when you use image field. To delete or change an image, right-click **Value** column. When you select **Select Image** in context menu, you can select new image. And also you can delete an image using **Remove Image** in context menu.

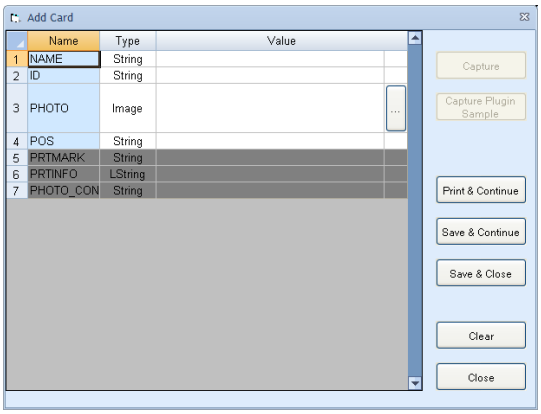


**Figure 37 Add Data**

If you click **Capture** button, you can use the images which are in USB Camera connected to PC.

To save data and keep the current window, click **Save & Close** button.

To save data and close the current window, click **Save & Continue** button.

To clear whole data in current window, click **Clear** button.

To close the window without save, click **Close** button.

When input data, entered data will be printed as Fig. 38

Selected data is updated through data field and it is displayed in preview section.

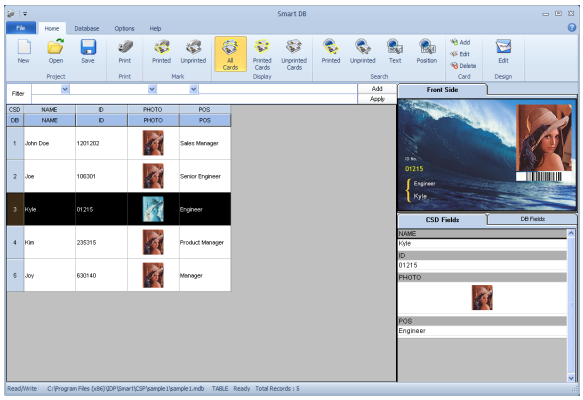

**Figure 38 View after Add Data**

## 3.7 Edit Image

The image which is connected on Image fields can be modified. You can edit image size and location easily using this function.

Double-click a point of indicated field in **preview screen** on the right side, then “Image Edit” window will be displayed. Or double-clicking **image field** on “CSD fields” on the bottom will be the same.

When “Image Edit” window be displayed, the amount of changes is displayed as percentage and values.
Blue dotted line on the center is the size of image field and the size is reduced according to the size of “Image Edit” window. To modify the location of an image, you can use left button of mouse by Drag and Drop.


**Figure 39 Image Editing Window**

In “Image Edit” window, there are many simple and useful tools for editing images.
**Brightness, Contrast, Zoom** and **Rotate** functions are included from the top

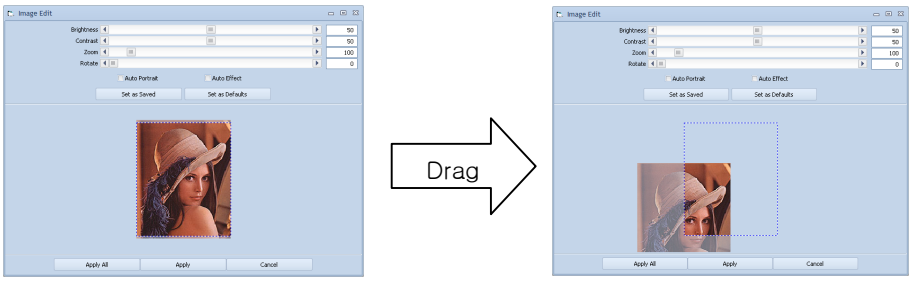


**Figure 40 Edit Image Location**

If brightness value is near to “0”, image will be darker.
If it is near to “100” image will be shown brighter.

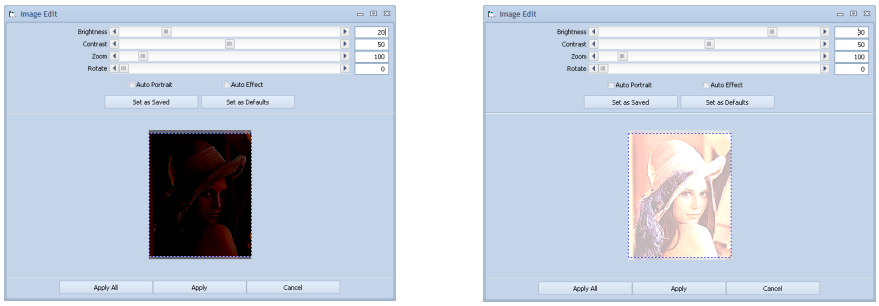


**Figure 41 Edit Brightness**

If contrast value is close to “0” then image will be more blur, on the contrary, if it is close to “100” then the image will be shaper.

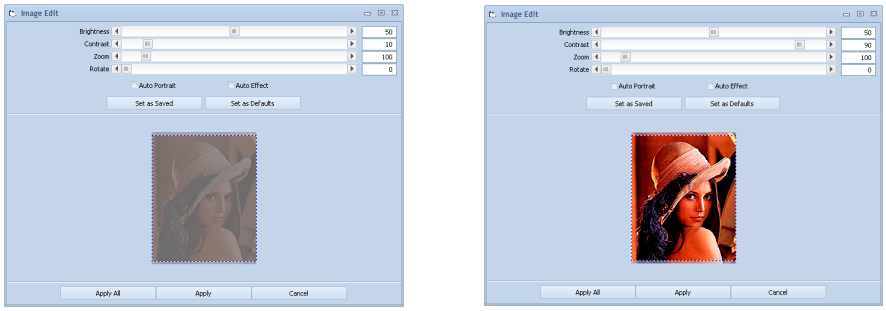


**Figure 42 Edit Contrast**

It is possible to resize an image by zoom function.

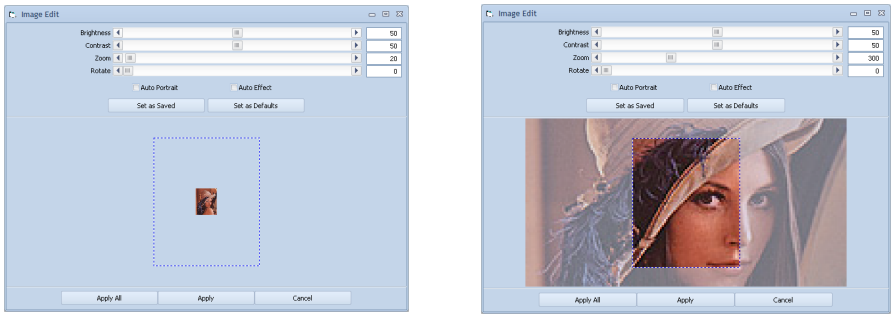


**Figure 43 Zoom in / Zoom out**

Image can be rotated by 90, 180, 270, 360 degrees through Rotate function.

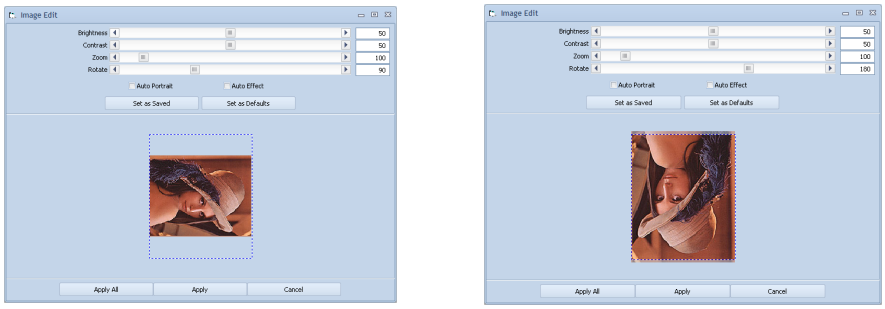


**Figure 44 Image Rotation**

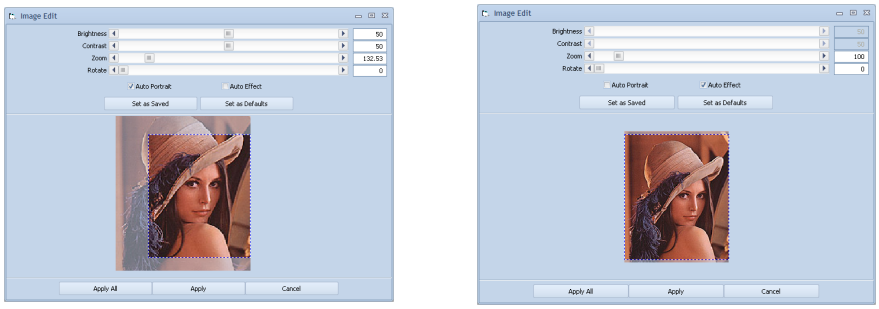


**Figure 45 Auto Portrait / Auto Effect**

To change brightness, contrast, zoom(minimize, maximize), rotator, move a scroll bar or input **value** in edit box, and press **Enter** key.
You can input zoom value until second minority. (0.xx).

Auto Portrait is the function to find the face in the Image automatically.
Click Auto Portrait check box, then the image is focused on the face and adjusts the size and position properly.
Auto Effect is the function to adjust the brightness and contrast in a picture automatically. Click Auto Effect check box, then brightness and contrast is adjusted properly.

To restore the default value, click Set as Saved button.
To turn back to the original CSD image file, click Set as Default button.
To save the changed setting value and close window, click Apply button.
To save and apply the changed setting value to all selected data and close window, click Apply All button. It will be take some time depend on the data volume.
To close without saving changed setting value, Click Cancel button.

If click **Apply** or **Apply All** button, you can see modified image on preview.

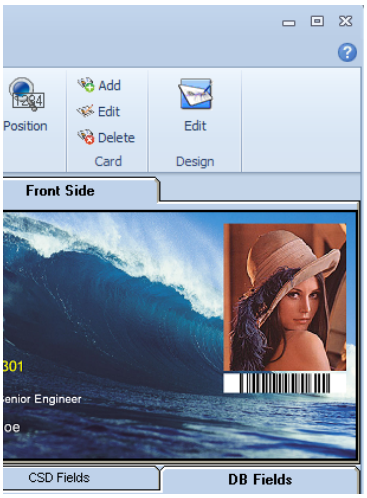


**Figure 46 Before Image Edit**

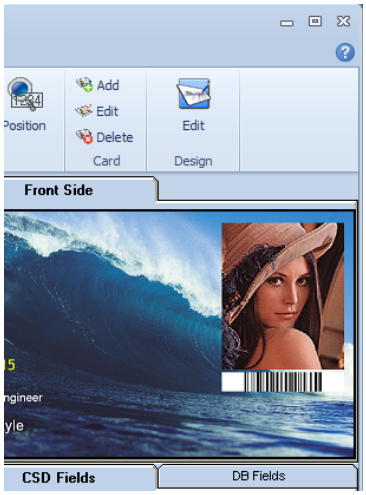


**Figure 47 After Image Edit**

When you modify the image, modified value will be set as Fig. 48.

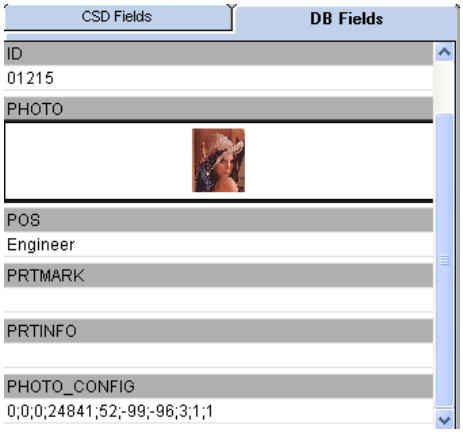


**Figure 48 Information of Edit Image**

## 3.8 Select Data

You can select data to print or remove printer data. When you select data, all of the lines will be selected. To select more data, you can click mouse left button and drag.

To select several data, press **Ctrl** Key and click lines. If a line has not selected, it will be added from selected lines. And if a line has selected already, it will be excluded from selected lines.

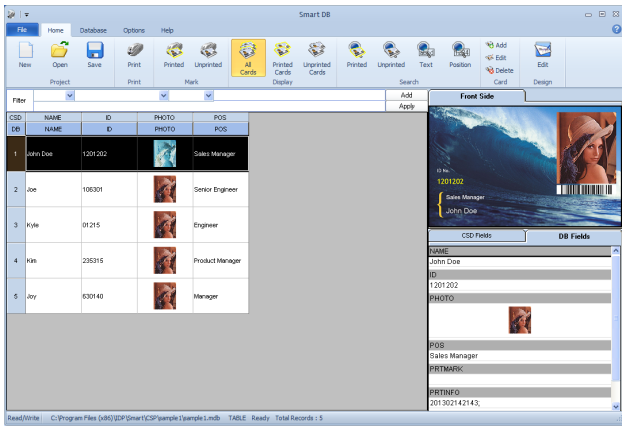

**Figure 49 Select data**

## 3.9 Print

① Select **data** to print edited cards.

② Click **Print** button in **File** tab or **Home** tab.


**Figure 50 Select Printer**

③ You can see the printer lists which are connected to PC or Network – Fig. 50

The printers which have a word “USB” on the lists are connected to PC directly.

The printers which names start with IP address are connected to network.

Select a **printer** on the list, and click **Confirm** button.

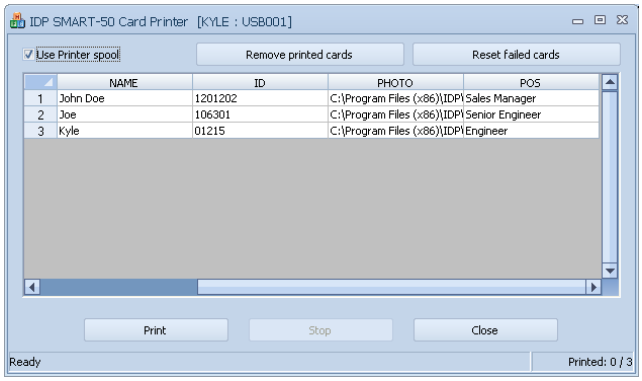


**Figure 51 Print Standby**

⑤ Figure 51 shows printer spooler window.

If there is several print data, you can click **Use Printer Spool** check box, so that continually you can print several data to save time.

Click **Print** button, then all data on the lists will be printed.

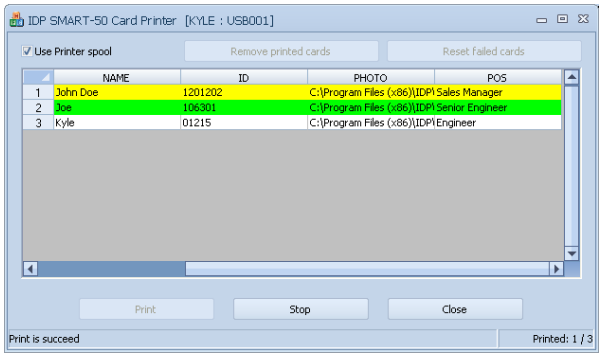


**Figure 52 Screen at the printing.**

The white background means the data which is ready to print. The green color means under printing. After finish the printing, the color will be changed to yellow. If there is an error during the printing, it will be changed to red color. Even though it is under printing, you can add more data to print.

⑥ If you click **Stop** button, it will stop all procedure after current printing data is done.

If you click **Close** button, it will close the window after current printing data is done. SmartDB will stay “Standby” mode if there are the data which are spooling or printing.

If you click **Remove Printed Card** button, Yellow color data that means printing is done will be removed.

If you click **Reset failed Card** button, it will retry to print Red color data that means

there was an error during printing.

⑦ When printing is done, click **Close** to return to main program.

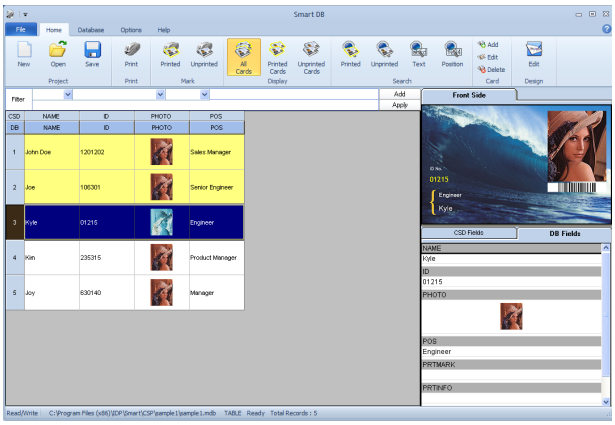

**Figure 53 Results**

Printing results will be applied on data lists as Fig. 53.

## 3.10 Change Printed Data

Printed data cannot be printed again to prevent reissuance,

If you want to re-print, select the card and click **Unprinted** button of **Mark** group on **Home** tab.

On the contrary, if you don't want to print unprinted data, select the card and click **Printed** button of **Mark** group on **Home** tab.

## 3.11 Use the Conditional Formula

If you want to display to specified data in the screen from lots of data, you can use "Filter Bar".


**Figure 54 Filter Bar**

First section defines the connecting method of conditional option, if there are more than two conditions, it will define the relations of two conditions.

| | |
|---|---|
| AND | The earlier condition and new input condition are satisfied both of them. |
| OR | The earlier condition or new input condition is satisfied. |

Second section is for selecting registered field name.

Third section defines to range of condition value.

| | |
|---|---|
| = | The data is equal to condition value |
| like | The data is including to condition value |
| > | The data is larger than condition value. |
| < | The data is smaller than condition value |
| <> | The data is different from condition value. |

Fourth section is for inputting condition value.

If you click" Add" button, it changes set value to conditional formula and display next line.

To create more than two "IF" formula , you can click "Add" button and after setting second conditional formula you can click "Add" button again.

If you click "Apply" button, the results of searching will be displayed from database by conditional formula.

For example, If you want to find data which includes "Der", you can find it as Fig. 55.

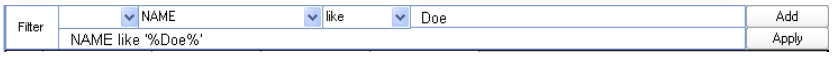


**Figure 55 Filterbar – Create Conditional formula**

# 4 Ribbon bar

## 4.1 Home tab

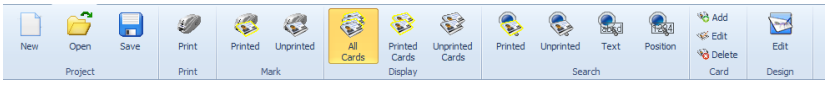


**Figure 56 Home tab**

The top of the program, at the bottom of the ribbon tab, ribbon bar is located.

### 4.1.1 Project

#### 4.1.1.1 New 

This button closes existed project, and create new project.

#### 4.1.1.2 Open 

This button loads saved project file.

#### 4.1.1.3 Save 

This button saves current project setting after create project.

### 4.1.2 Print

#### 4.1.2.1 Print 

This button prints selected data through card printer.

For more information, Please refer to Chapter 3.9

### 4.1.3 Mark

#### 4.1.3.1 Printed 

This button set selected data as print completed status.

#### 4.1.3.2 Unprinted 

This button set selected data as unprinted data so that you can print the data again.

### 4.1.4 Display

#### 4.1.4.1 All Cards 

This button shows all of the data of connected database.

#### 4.1.4.2 Printed Card 

This button shows all printed data of connected database.

#### 4.1.4.3 Unprinted Cards 

This button show unprinted data of connected database.

### 4.1.5 Search

#### 4.1.5.1 Printed 

This button finds next printed card

#### 4.1.5.2 Unprinted 

This button finds next unprinted card

#### 4.1.5.3 Text 

This button finds cards which have specified string.

If you click the button, the dialog box will be displayed as Fig. 57

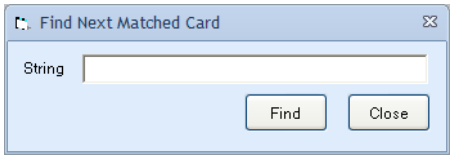


**Figure 57 find matched card**

To find specified data, click **Find** button after filling text box.

#### 4.1.5.4 Position 

This button finds the data which is on specified position.

If you click the button, the dialog box will be displayed as Fig. 58


**Figure 58 Find Position**

To find specified position, click **Find** button after filling data position.

### 4.1.6 Card

#### 4.1.6.1 Add 

This button adds new data. Please refer to Chapter 2.3.

#### 4.1.6.2 Edit 

This button edits selected data. Please refer to Chapter 2.4.

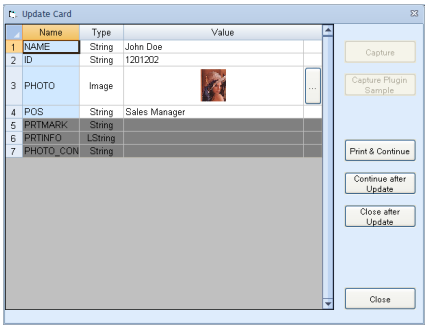


**Figure 59 Editing Data**

#### 4.1.6.3 Delete 

This button deletes the selected cards. If you removed data once, it could not be recovered. So please be careful to use this button.

### 4.1.7 Design

#### 4.1.7.1 Edit 

This button executes SmartDesign program to modify the used CSD file in the program.

## 4.2 File tab

### 4.2.1 New

After close existed project, create new project from designed CSD file.

### 4.2.2 Save

Save current project setting.

### 4.2.3 Open

After close existed project, load the saved project file.

### 4.2.4 Close

Close the opened project.

### 4.2.5 Print

#### 4.2.5.1 Print

Print a selected card. Please refer to Chapter 3.9

#### 4.2.5.2 Save Preview as BMP file

Save preview as BMP file. You can select the side to save.

### 4.2.6 DataBase

#### 4.2.6.1 Import data from ODBC

Through ODBC, you can input data from the other DB

When execute menu, the window like Fig. 60 “Select ODBC Data source” will be displayed.

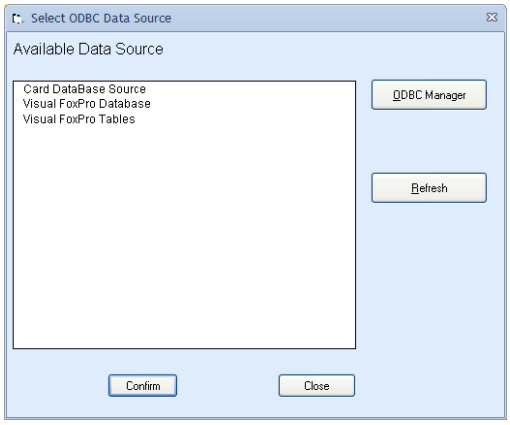


**Figure 60 Select ODBC Data Source**

To select data source, click **Confirm** button.

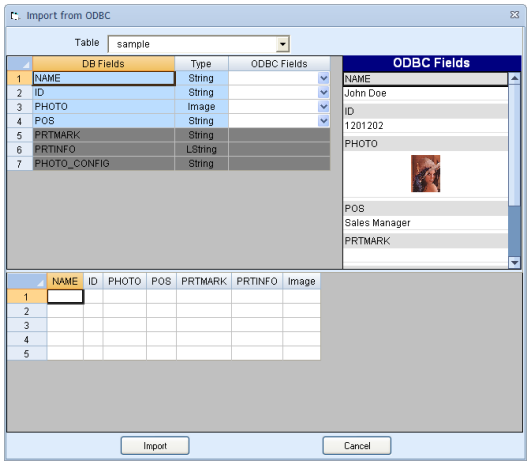


**Figure 61 Import from ODBC**

Structure of selected data source on the left of screen is displayed.

Connect ODBC field item to DB field.

If the connecting is done, click **Import** button.

#### 4.2.6.2 Import Excel

Import excel file and you can modify data.

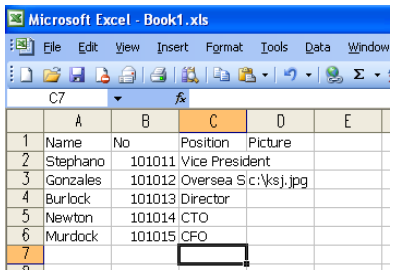


**Figure 62 Contents of Excel File**

Fig. 62 is the example of excel file.

Click **Import data from Excel** button and select excel file.

“Import from Excel” window will appear.

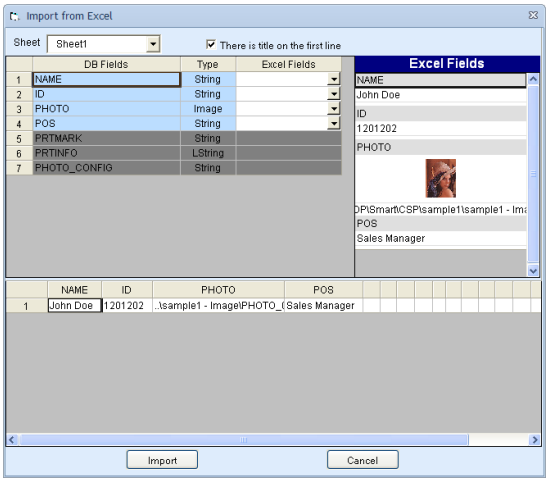


**Figure 63 Import from Excel**

You can check the bottom of screen and there will be descriptions of analyzed contents of excel file.

First line is field name. In fact, data will be imported from second line.

The field name and selected data from list of bottom will be displayed on the right side of screen.

On the left side of screen, you can check DB field name and you can

select excel file in “Excel fields” topic on the right side.

To Connect DB fields and excel fields, click **Import** button.

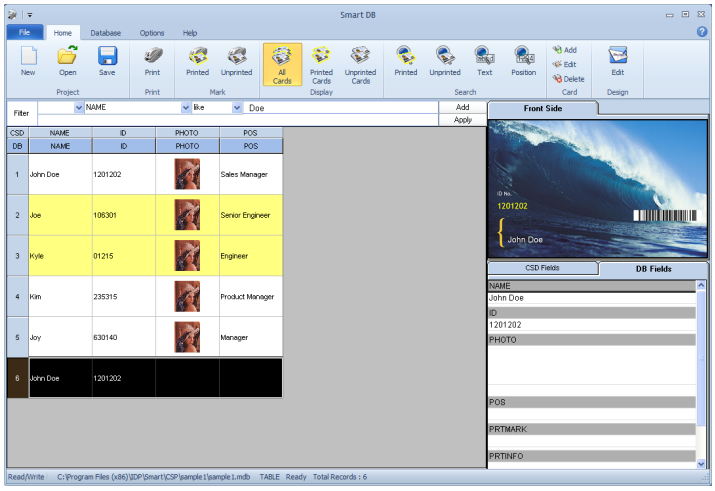


**Figure 64 after import from Excel**

### 4.2.6.3 Export data to ODBC

Save opened project data in MDB file.

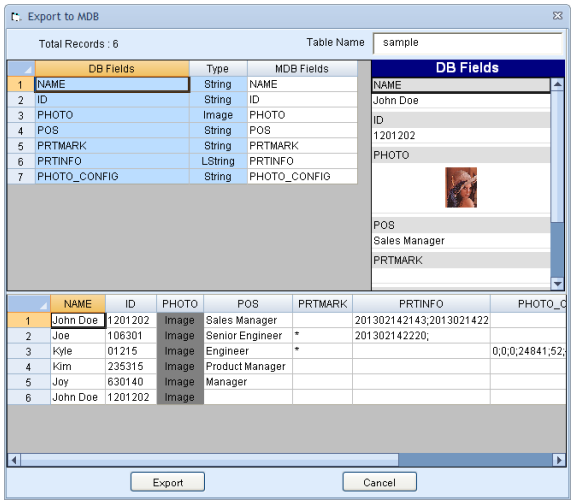


**Figure 65 Export to MDB**

All of data fields and data in the project will be displayed at the

bottom of the screen and selected data from list will be displayed with field name on the upper right.

DB field name and created MDB field name will be displayed on the upper left.

If you double-click “MDB Fields”, you can modify contents.

Image data will be saved to MDB from the OLE object type.

The number of all data will be shown on the upper screen. All fields will be created.

“Table” sets the name of created table on MDB. Basically current project name is determined.

Click **Export** button, and specify location of MDB file and name. If you specify MDB file name, project data will be recorded immediately.

#### 4.2.6.4 Export data to Excel

You can save the current project data using this function.

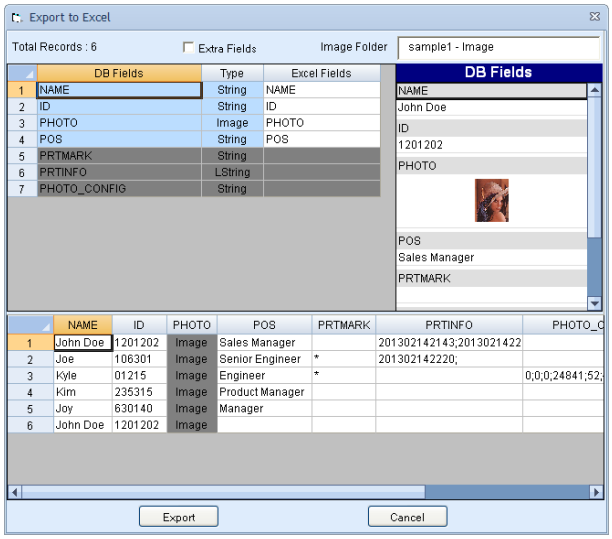


**Figure 66 Export to Excel**

All of data fields and data from project will be shown at the bottom of the screen. You can check selected data on the upper right of the screen. Also you can check DB field name and field name in excel file on the upper left side. If you double-click “Excel Fields”, you can edit contents.

"Total records" and "Extra Fields" will be shown on the upper screen. When "Extra Fields" is activated, hided fields (print results field, print history field, image field, modify history field) will be recorded in an excel file.

"Image directory" column defines the directory which is recorded image file in. It will create the specified excel file name which is saved to sub-directory.

If you click **Export** button, it will set the location and name of exported excel file. If you set an excel file name, project data will be record in an excel file.

### 4.2.7 Print History

Data print history from opened project data will be shown. When execute menu, "Log file select" window will pop up. Print history is recorded by monthly. When you select a file in specified month, print history appears as Fig. 67

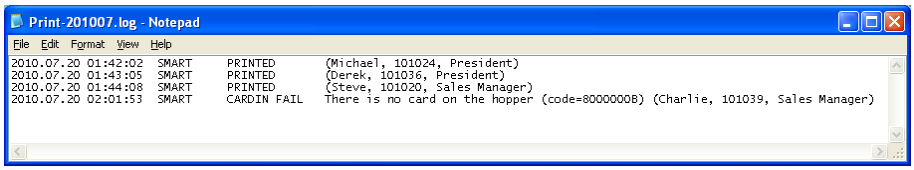


**Figure 67 Print History**

Print history will be recorded print time, date, used printer ID, results of printer, and simple information of error, error code and information of used data.

### 4.2.8 Exit

Exit the program. If you change the structure of project to exit the program, you have to save before you exit the program.

## 4.3 Database Tap

### 4.3.1 Connection

#### 4.3.1.1 Connect MDB

You can connect MDB file. Please refer to Chapter 3.2.2

#### 4.3.1.2 Connect ODBC

You can connect database through ODBC.

Please refer to Chapter 3.3

#### 4.3.1.3 Disconnect

This function is to disconnect database.

If the database is disconnected, data and information will be disappeared.

### 4.3.2 Setting

#### 4.3.2.1 Field Setting

When it is connected to database, a connection is composed between CSD field and database.

DB field which connects with CSD field appears on the left side and you can change DB field, on the right side, also you can check data in current DB Fields.

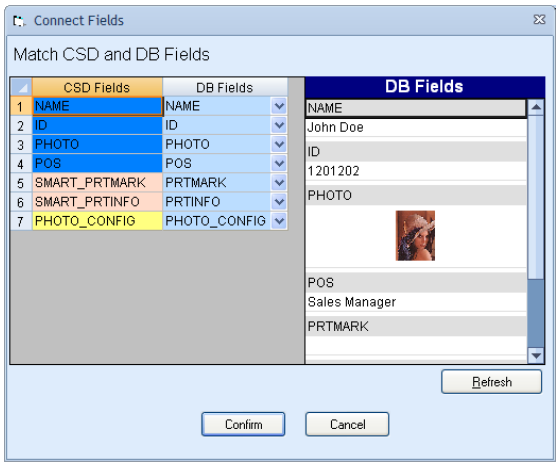


**Figure 68 Set field Connecting**

#### 4.3.2.2 Compress and Restore

If connected database type is MDB, you can use this function. MDB charges inefficient space more than needed. As time goes by, space for MDB is

increased more and more. At that time, “Compact and Recovery” function is able to use and through this function useless space can be removed.

### 4.3.3 DB Security

#### 4.3.3.1 Password

You can set a password in MDB of project.

Click **Password** button, “Password Configuration” window is displayed.

Click **OK** button after input the same password that you want in the “New Password” and “Confirm Password”, then password is set.

If you open the project locked by password, “Input Password” window is displayed. Input Password and click **OK** button to open the project.

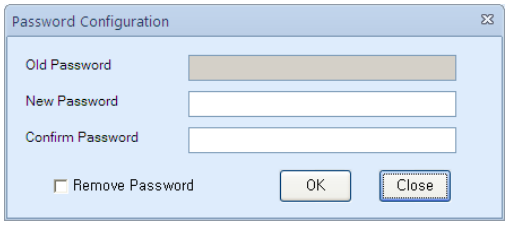


**Figure 69 Set Password**

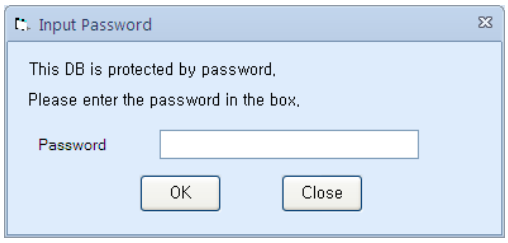


**Figure 70 Input Password**

### 4.3.4 Card

#### 4.3.4.1 Add

This button adds new data. Please refer to Chapter 2.3.

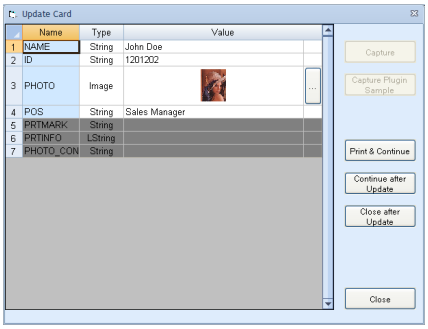


**Figure 71 Add Card**

#### 4.3.4.2 Edit

This button edits selected data. Please refer to Chapter 2.4.

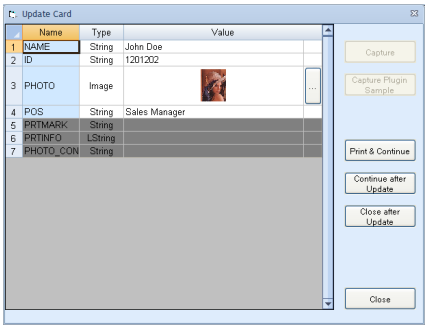


**Figure 72 Edit Card**

#### 4.3.4.3 Delete

This button deletes the selected cards. If you removed data once, it could not be recovered. So please be careful to use this button.

## 4.4 Options tab

### 4.4.1 Language

Select the preferred language.

### 4.4.2 Plugins

Select Plugin lists and Plugin you are going to use.

Please refer to Appendix.1

#### 4.4.2.1 Image Capture

The plugins those are available to capture images are displayed as low level of menu and you can select plugins for “Add” and “Edit”.

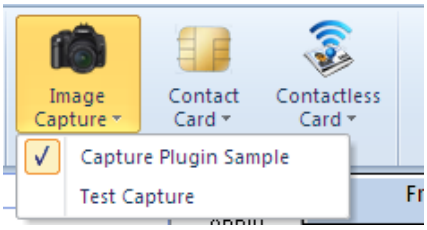


**Figure 73 Select Capture Plugins**

#### 4.4.2.2 Contact Card

The plugins those are available to encode contact card are displayed as low level of menu and you can select plugins for “Add” and “Edit”.

If you select plugin and click, “Set Contact Card Encoding” window is displayed.

To use contact card encoding, click **Use Contact Card Plugin Encoding** check box. Click **Encoding** check box in field to encode.
If you click **Setup** button, “Contact_Plugin_Sample.ini” file is open.

You can configure options of contact card encoding field. Now only sample is provided in contact card encoding, so field is empty. For further information, please refer to Appendix.1

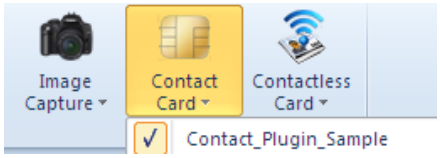


**Figure 74 Select Contact Plugins**

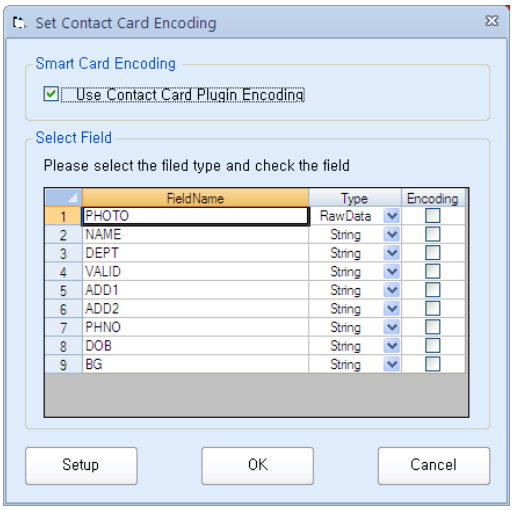


**Figure 75 Contact Plugins Option**

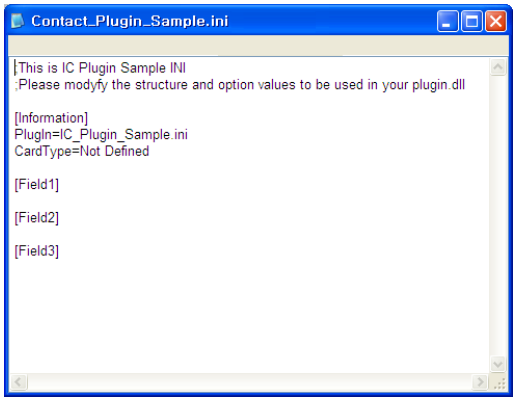


**Figure 76 Contact Plugins INI Setup**

#### 4.4.2.3 Contactless Card

The plugins those are available to encode contactless card are displayed as low level of menu and you can select plugins for “Add” and “Edit”.

If you select plugin and click, “Set Contactless Card Encoding” window is displayed.

To use contactless card encoding, click **Use Contactless Card Plugin Encoding** check box. Click **Encoding** check box in field to encode.
If you click **Setup** button, “Contactless_Plugin_Mifare1k.ini” file is open.

You can configure options of contactless card encoding field. Now only Mifare

1K is provided in contactless card encoding. For further information, please refer to Appendix.1

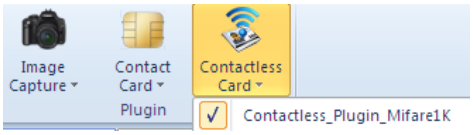


**Figure 77 Select Contactless Plugins**

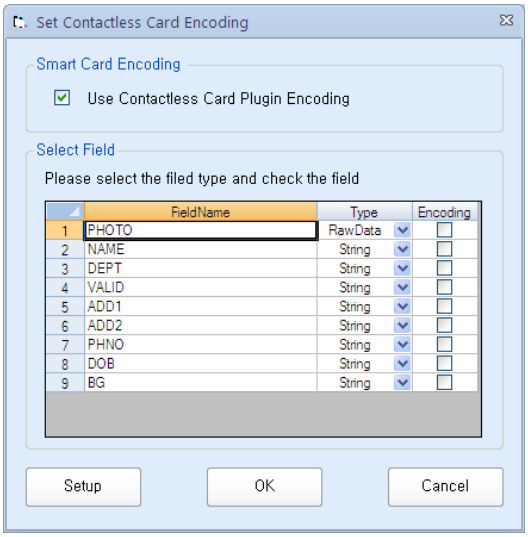


**Figure 78 Contactless Plugins Option**

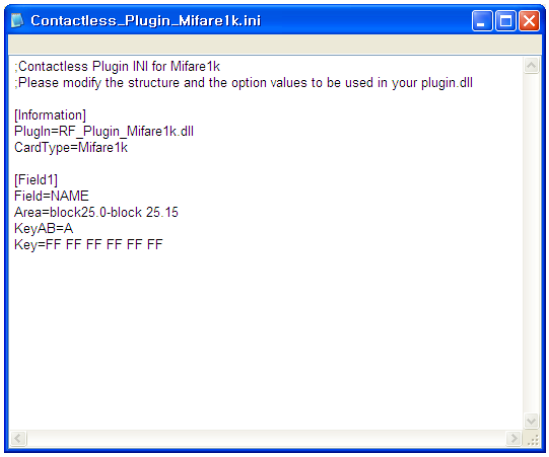


**Figure 79 Contactless Plugins INI Setup**

## 4.5 Help tab

### 4.5.1 Manual

#### 4.5.1.1 Manual

This SmartDB user manual will be open.

### 4.5.2 About

#### 4.5.2.1 About

Introduce program version, information etc.


**Figure 80 SmartDB Information**

# APPENDIX

# 1 Plugin

Other functions besides basic functions in SmartDB are supported as Plugin type.

If you make a file according to SmartDB plugin regulation, the plugin can be used in SmartDB.

## 1.1 Plugin Registration

You can use any name for plugin file but the extension type should be ".dll"

If you copy plugin files in "plugin" folder where SmartDB is installed in, and restart SmartDB, you can use the plugin in SmartDB program.

If there is no plugin in "plugin" folder, plugin buttons are inactivated.

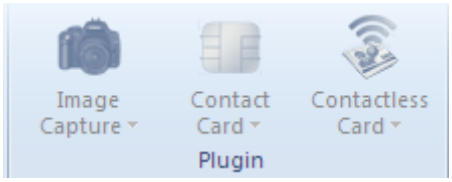


**Figure 81 No plugin.dll**

After copying plugin files like Figure 75 and restart SmartDB, the plugins are registered automatically and plugin buttons are activated.

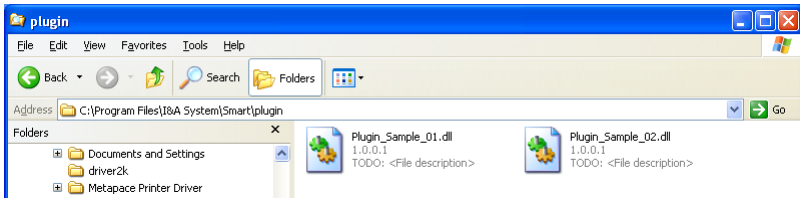


**Figure 82 Copy of Plugins**

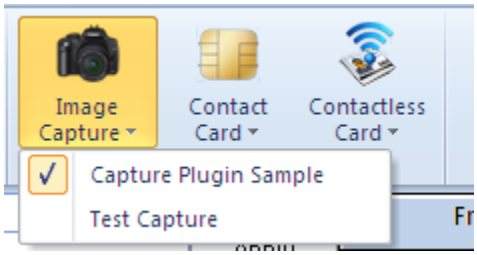


**Figure 83 Plugin Auto-Registration**

## 1.2 Plugin Development

You can develop plugins as “DLL” type.

There are some regulations for developing plugin and you must follow the regulations.

### 1.2.1 Plugin Functions

Plugin must include below functions.

| **int WINAPI GetPluginInfo(SPI_INFO * pInfo)** |
|---|
| **int WINAPI StartPlugin(HANDLE hDone, SPI_VDATA* pInput, int nSize)** |
| **int WINAPI EndPlugin(SPI_VDATA* pOutput, int nSize)** |

The below is the explanation of each function.

#### ① GetPluginInfo()

This function gets plugin information.

| **int WINAPI GetPluginInfo(SPI_INFO * pInfo)** | |
|---|---|
| Imports plugin information | |
| Parameter | **** pInfo*** |
| | Structure pointer for the information of plugin. |
| | Please see the explanation of **SPI_INFO** below. |
| Return | 0 : Success |
| | Others : Fail |

#### ② StartPlugin()

This function begins plugin action.

| **int WINAPI StartPlugin(HANDLE hDone, SPI_VDATA* pInput, int nSize)** | |
|---|---|
| Begins plugin action. | |
| Parameter | ***hDone***<br>An event handler for alarming to SmartDB after image capture is done.<br>If bUseEvent of SPI_INFO is false, *hDone* will be NULL and there is no event after capture is done.<br>But all the capture process should be done in SPI_Start function.<br>If bUseEvent of SPI_INFO is true and Plugin action is done, you should generate an event using *hDone* handle.<br>generate an event like below code.<br>`::SetEvent( hDone );`<br>If calling EndPlugin(), this function is finished though Plugin action is in progress.<br>****pInput***<br>A pointer of data to get from SmartDB<br>***nSize***<br>Size of structure of SPI_VDATA to get from SmartDB |
| Return | 0 : Success<br>Others : Fail |

### ③ EndPlugin()

This function finishes plugin action.

| **int WINAPI EndPlugin(SPI_VDATA* pOutput, int nSize)** | |
|---|---|
| Finishes plugin action.<br>Please end plugin process in this function. | |
| Parameter | ****pOutput***<br>A pointer of SPI_VDATA to send to SmartDB when plugin action is finished. |

| | |
|---|---|
| | ***nSize***<br>Size of structure of SPI_VDATA to send to SmartDB |
| Return | 0 : Success<br>Others : Fail |

### 1.2.2 Plugin Structure

#### ① SPI_INFO

This structure shows the information of plugin.

```
typedef struct
{
    WCHAR   szName[64];   // The name of plugin
    WCHAR   szDesc[256];  // The description of plugin
    int     nClassId;     // Class information of plugin
    BOOL    bUseEvent;    // Check if event handle use or not
    int     nTimeOut;     // Timeout of plugin
    BYTE    reserved[56];
} SPI_INFO;
```

Shows the information of plugin

***szName*** is plugin's name. Please be careful of overlapping between this and other names.

2 Byte Wide String (Unicode), MAX size is 64 characters including NULL.

***szDesc*** is the brief information of plugin.

2 Byte Wide String (Unicode), MAX size is 256 characters including NULL.

***nClassid*** means class-code which plugin belongs to.

```
#define SPI_CLASS_UNKNOWN              0xFFFFFFFF
#define SPI_CLASS_IMAGEACQUISITION     0x00000001
```

```
#define SPI_CLASS_CONTACT_CARD          0x00000010
#define SPI_CLASS_CONTACTLESS_CARD      0x00000100
```

SPI_CLASS_IMAGEACQUISITION is the class code of plugins to get image from camera, sign-pad and etc. SPI_CLASS_CONTACT_CARD, _CONTACTLESS_CARD are the class code of plugins to encode contact card or contactless card with PC/SC protocol.

***bUseEvent*** sets whether event handler to alarm that plugin action is done is used or not. If the value is TRUE, StartPlugin() can get event handler. If it is FALSE, plugin action should be done in StartPlugin function.

***nTimeOut*** finishes plugin action by calling EndPlugin() when it receives no response during the setting time after calling StartPlugin(). If bUseEvent is FALSE, input this value 0. If it is TRUE, input this value in seconds.

***reserved*** is not available now. Fill it with 0.

② **SPI_VDATA**

SPI_VDATA is the variable size structure that plugin function and SmartDB sends and gets. The input/output data of this structure depends on the class of plugin.

Please see the explanation of class of plugin below.

```
typedef struct
{
    int    nVersion;     // Version of SPI_VDATA
    int    nTotalSize;   // Total size of structure includes header and data
    int    nFields;      // number of fields
    SPI_VDATA_VFIELD     field[nFields];
} SPI_VDATA;
```

the variable size structure that plugin function

***nVersion*** is the version of SPI_VDATA. Now version is 1.

***nTotalSize*** is the total size of structure includes header and data. Be careful because it depends on size of SPI_VDATA_VFIELD.

***nFields*** means the number of fields. It depends on the input/output state of nClassID of plugin. SPI_VDATA_VFIELD is defined variably according to nFields value.

***field[nFields]*** is the SPI_VDATA_VFIELD structure of field information. Please see the explanation of structure below.

```
typedef struct
{
    WCHAR   szName[32];      // Name of Field
    int     nType;           // Type of Field
    int     nSize;           // size of Field
    BYTE    value[nSize];    // Field Data to input
} SPI_VDATA_VFIELD;
```

the variable size structure of field information declared in SPI_VDATA.

***szName*** is the name of field.

2 Byte Wide String (Unicode), MAX size is 32 characters including NULL.

***nType*** is defined by data type depends on field value.

```
#define   SPI_FIELD_DATATYPE_INT      1  // Integer
#define   SPI_FIELD_DATATYPE_STRING   2  // 2Byte Wide String
#define   SPI_FIELD_DATATYPE_RAW      3  // Raw data
```

***nSize*** is the size of field (Byte). It is depends on field type.

| ***nType*** | ***nSize*** |
|---|---|
| SPI_FIELD_DATATYPE_INT | 4 |

| | |
|---|---|
| SPI_FIELD_DATATYPE_STRING | Size of string including 2Byte NULL |
| SPI_FIELD_DATATYPE_RAW | Size of Raw data |

***value*** is data of field. If nType is SPI_FIELD_DATATYPE_STRING, 2Byte Wide String (Unicode) including NULL.

### 1.2.3 Plugin Class

There are 3 plugin classes which are SPI_CLASS_IMAGEACQUISITION, SPI_CLASS_CONTACT_CARD and SPI_CLASS_CONTACTLESS_CARD now. More classess will be added in the future.
The variable structure SPI_VDATA depends on Class type.

#### ① SPI_CLASS_IMAGEACQUISITION

If nClassId is SPI_CLASS_IMAGEACQUISITION, the pointer *pInput and *pOutput is defined as below.

**[*pInput]**

There is no field data to get from SmartDB in SPI_CLASS_IMAGE ACQUISITION. The value is NULL in StartPlugin() function.

**[*pOutput]**

EndPlugin() function send the path of captured image to SmartDB. You should input the path and data of captured image in *pOutput.

| **SPI_VDATA** | | ***pOutput** |
|---|---|---|
| **nVersion** | | 1 |
| **nTotalSize** | | Total size of SPI_DATA |
| **nFields** | | 1 |
| **field[0]** | **szName** | L"ImageCap" |
| | **nType** | SPI_FIELD_DATATYPE_STRING |
| | **nSize** | Size of value including NULL |
| | **value** | Path of captured image + (NULL) |

Input 1 which is SPI_VDATA version now in nVersion.

Input the total size of SPI_DATA including version information.
Input 1 in nFields because the path of captured image is used as one field.
Input Wide String L"ImageCap" including NULL in field[0].szName.
Input SPI_FIELD_DATATYPE_STRING in field[0].nType because the path of captured image is string type.
Input the size of value including NULL(2Byte).
Input the path of captured image in field[0].value.
Then it can send the data about captured image to SmartDB by *pOutput pointer.

For example, if the path of captured image is "C:\image.bmp", SPI_VDATA *pOutput is as below.

| **SPI_VDATA** | | ***pOutput** |
|---|---|---|
| **nVersion** | | 1 |
| **nTotalSize** | | 110 |
| **nFields** | | 1 |
| **field[0]** | **szName** | L"ImageCap" |
| | **nType** | SPI_FIELD_DATATYPE_STRING |
| | **nSize** | 26 |
| | **value** | L"C:\\image.bmp" |

The Pseudo code of plugin with SPI_CLASS_IMAGEACQUISITION is as below.

## [Pseudocode]

```
int WINAPI GetPluginInfo(SPI_INFO* pInfo)
{
    //Input the information of Plugin
    pInfo->szName = L" Capture plugin" ;
    pInfo->nClassid = SPI_CLASS_IMAGEACQUISITION;
    pInfo->nTimeOut = 0;
    pInfo->bUseEvent = false;
    return nres;
}
```

```
int WINAPI StartPlugin(HANDLE evtDone, SPI_VDATA* pInput, int nSize)
{
    //Capture Image and Save the path of image
    GetCaptureImage();
    SaveImagePath();
    return nres;
}

int WINAPI EndPlugin(SPI_VDATA* pOutput, int nSize)
{
    //Stop Thread in progress and Return the path of image
    pOutput->nVersion = 1;
    pOutput->nField = 1;
    wcscpy(pOutput->field[0].szName[0], szImageName);
    pOutput->field[0].nType = SPI_FIELD_DATATYPE_STRING;
    pOutput->field[0].nSize = wcslen(szImagePath) + 2;
    memcpy(pOutput->field[0].value, szImgPath, pOutput->field[0].nSize);
    pOutput->nTotalSize = 12 + 72 + pOutput->field[0].nSize;
    return nres;
}
```

### ② SPI_CLASS_CONTACT_CARD, SPI_CLASS_CONTACTLESS_CARD

If nClassId is SPI_CLASS_CONTACT_CARD / SPI_CLASS_CONTACTLESS_CARD, the pointer *pInput and *pOutput is defined as below.

**[*pInput]**

You can get the data about Smart Card encoding in SmartDB through *pInput.

| **SPI_VDATA** | ***pInput** |
|---|---|
| **nVersion** | 1 |
| **nTotalSize** | Total size of SPI_VDATA |

| nFields | | 1 + k |
|---|---|---|
| **field[0]** | **szName** | Name of Smart Card Reader |
| | **nType** | SPI_FIELD_DATATYPE_RAW |
| | **nSize** | 4 |
| | **value** | Function Pointer of Transmit() function |
| **field[1]** | **szName** | Name of 1st Field |
| | **nType** | Type of 1st Field |
| | **nSize** | Size of 1st Field |
| | **value** | Data of 1st Field |
| **field[2]** | | 2nd Field |
| **field[3]** | | 3rd Field |
| **……** | | …… |
| **filed[k]** | | kth Field |

Input 1 which is SPI_VDATA version now in nVersion.
Input the total size of SPI_DATA including version information.
Input the name of Smart Card Reader and Function pointer of Transmit() function (4Byte) in 1st Field.
If nClassId is SPI_CLASS_CONTACT_CARD, function pointer should be indicated SmartComm_ICTransmit() function of SDK. If nClassId is SPI_CLASS_CONTACTLESS_CARD, function pointer should be indicated SmartComm_RFTransmit() function of SDK.

For example, this function pointer is used like below code.

```
//Definition
typedef int (*PFN)(int , DWORD , BYTE* , DWORD* , BYTE* );

//Usage in the function
PFN TransmitAPDU;
TransmitAPDU = *(PFN*)(theApp.pVData->field[0].value);
TransmitAPDU ( DEV_INTERNALRF, nlencmd, btCmd, dwlenrcv, btRcv);
```

For further information, please refer to Smart SDK manual.
And refer to source code in this plugin in **"Program Files\IDP\Smart \PluginSample"** after SmartDB installation.

The field data to get from SmartDB begins field[1].
Input the name of 1st field in field[1].szName.
Input the type of 1st field in field[1].nType.
Input the size of 1st field in field[1].nSize.
Input the field data to get from SmartDB in field[1].value.
If there are more than 2 fields from SmartDB, 2nd field data is entered in field[2]. In this way, k numbers of fields are entered and send to plugin function through *pInput pointer.

For example, consider that you do Contactless Card Encoding.
nClassId is SPI_CLASS_CONTACTLESS_CARD, Smart Card Reader is "OMNIKEY CardMan 5X21-CL 0", the number of the field to get from SmartDB is 1, name of field is "Name" and value is "John".
SPI_VDATA is as below.

| **SPI_VDATA** | | ***pInput** |
|---|---|---|
| **nVersion** | | 1 |
| **nTotalSize** | | 170 |
| **nFields** | | 2 |
| **field[0]** | **szName** | L"OMNIKEY CardMan 5X21-CL 0" |
| | **nType** | SPI_FIELD_DATATYPE_RAWDATA |
| | **nSize** | 4 |
| | **value** | 4bytes Function pointer |
| **field[1]** | **szName** | "Name" |
| | **nType** | SPI_FIELD_DATATYPE_STRING |
| | **nSize** | 10 |
| | **value** | "John" |

**[*pOutput]**
There is no field data to send to SmartDB in Contact/Contactless card encoding. The value is NULL in EndPlugin() function.

The Pseudo code of plugin with SPI_CLASS_CONTACT_CARD, SPI_CLASS_CONTACTLESS_CARD is as below.

## [Pseudocode]

```
typedef int (*PFN)(int , DWORD , BYTE* , DWORD* , BYTE* );

int WINAPI GetPluginInfo(SPI_INFO* pInfo)
{
    //input the information of Plugin
    pInfo->szName = L" Plugin Smart Card" ;
    pInfo->nClassid = SPI_CLASS_CONTACTLESS_CARD;
    pInfo->nTimeOut = 0;
    pInfo->bUseEvent = false;
    return nres;
}

int WINAPI StartPlugin(HANDLE evtDone, SPI_VDATA* pInput, int nSize)
{
    //Read the field information from SmartDB and INI information
    ReadSetupFile();
    //Get the Function Pointer of Transmit()
    PFN TransmitAPDU;
    TransmitAPDU = *(PFN*)( pInput->field[0].value);

    //Authentification and ReadWrite
    BYTE comdbuf[] = L"……..";  // Define APDU Command
    TransmitAPDU(DEV_INTERNALRF, nlencmd, comdbuf, dwlenrcv, btRcv);
    // Repeat Transmit APDU to read/write smart card
return nres;
}

int WINAPI EndPlugin(SPI_VDATA* pOutput, int nSize)
{
    //Stop Thread in progress and quit plugin
    TerminateCapThread();
    return nres;
}
```

## 1.3 Usage of RF_ Plugin_Mifare1k.dll

SmartDB provides plugin DLL of SPI_CLASS_CONTACTLESS_CARD for encoding Mifare1K of contactless card encoding. Please set the fields to encode in SmartDB and input the information of encoding Mifare.

### 1.3.1 SmartDB Configuraion

Click **Contactless card** button in **Option** tab, select **Contactless _Plugin_Mifare1K**.

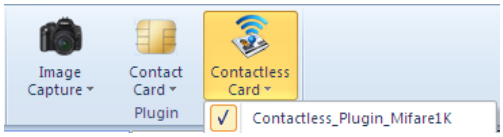


**Figure 84 Plugin Selection**

Then "Set Contactless Card Encoding" window is displayed.

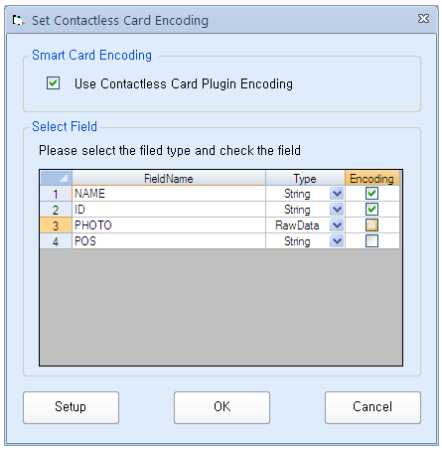


**Figure 85 Set Plugin option**

To use contactless card encoding, click **Use Contactless Card Plugin Encoding** check box. Click **Encoding** check box in field to encode.

In above Figure, NAME and ID fields are checked, so *pInput data of SPI_VDATA is as below.

| **SPI_VDATA** | ***pInput** |
|---|---|

| | | |
|---|---|---|
| **nVersion** | | 1 |
| **nTotalSize** | | 256 |
| **nFields** | | 3 |
| **field[0]** | **szName** | L"OMNIKEY CardMan 5X21-CL 0" |
| | **nType** | SPI_FIELD_DATATYPE_RAWDATA |
| | **nSize** | 4 |
| | **value** | 4bytes Function pointer |
| **field[1]** | **szName** | "NAME" |
| | **nType** | SPI_FIELD_DATATYPE_STRING |
| | **nSize** | 10 |
| | **value** | "John" |
| **field[2]** | **szName** | "ID" |
| | **nType** | SPI_FIELD_DATATYPE_STRING |
| | **nSize** | 14 |
| | **value** | L"201302" |

### 1.3.2 INI File Configuration

Next, edit INI file to set the configuration to encode Mifare card.

If you click **Setup** button, "Contactless_Plugin_Mifare1k.ini" file is open.

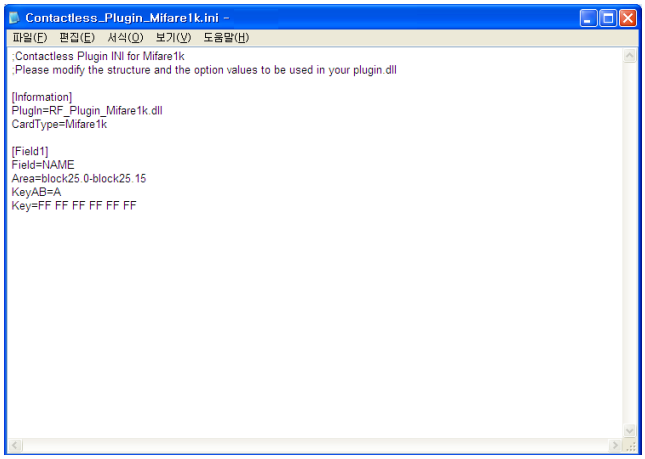


**Figure 86 INI Plugin Configuration**

Basically, Contactless_Plugin_Mifare1k.ini file is defined as below.

| **Contactless_Plugin_Mifare1K.ini** |
|---|
| **[Information]**<br>**PlugIn=RF_Plugin_Mifare1k.dll**<br>**CardType=Mifare1k**<br><br>**[Field1]**<br>**Field=NAME**<br>**Area=block25.0-block25.15**<br>**KeyAB=A**<br>**Key=FF FF FF FF FF FF** |
| Configuration value for encoding in DLL. You can edit it for your purposes.<br><br>***[Information]*** is the information of plugin.<br>***PlugIn*** means the dll file to use this ini file.<br>***CardType*** is the type of card to encode.<br><br>***[Field#]*** is the paragraph to define the information of field to encoding. You can make it as many fields as encoding.<br>***Field*** should be entered as the name of field to get from SmartDB. For example, if Field=NAME is defined, DLL file matches the string "NAME" in INI file with the string "NAME" to get from SmartDB, field[i].szName, then use the same data to encode.<br>**Area** is the information of position to encode in Smart Card. In this case, it is defined by block on Mifare1k DLL. "block25.0-block25.15" means that encoded block is 25th block and it encodes from 0th Byte to 15th Byte of 25th block. You can define "block25" instead. It is analyzed by RF_Plugin_Mifare1k.dll. Also if it needs several block to encode, you can use more block like "block25-block26".<br>This RF_Plugin_Mifare1k.dll is for Mifare1K card to encode only. Mifare1K don't allow encoding in 0th block and every 4th block (3, 7, 11, 15….). If user sets "Area=block3", it encodes next block (4th block) automatically because RF_Plugin_Mifare1k.dll analyzes the syntax. |

***KeyAB*** defines Key Side whether to use A of Key or B of Key.

***Key*** means the value of Key to load Key Side defined KeyAB. 6 Byte, Hex String.

### 1.3.3 Data Encoding

In this sample Contactless_Plugin_Mifare1K.INI in 1.3.2 and predefined *pInput of SPI_VDATA in 1.3.1, only 1st field will be encoded because only [Field1] section is defined in INI file. DLL encodes “John”, the data of field[1].value of *pInput to match “Field=NAME” with. Area is defined from 0th Byte to 15th Byte of 25th block.

| block 25 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | J | | o | | h | | n | | | | | | | | | |

In above table, the size of real encoded data is 10 Bytes because it is Unicode and includes NULL. It is encoded from 0th Byte to 9th Byte of 25th block. However user defined Area to 15th Byte, NULL is filled from 10th Byte to 15th Byte.

And data of field[2] in *pInput of SPI_VDATA is not encoded because only [Field1] section is defined in INI file.

Refer to source code in “Program Files\IDP\Smart\Plugin” folder after SmartDB installation.