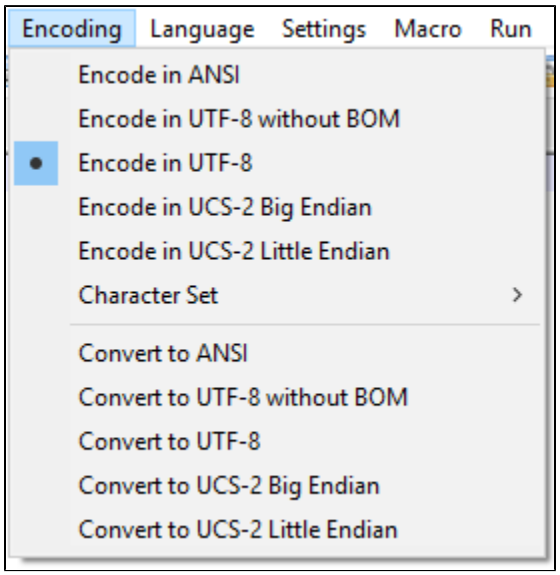
CSV Data Import

Many organizations use spreadsheet software for their IT documentation to view information in form of tables. This is not only confusing, but requires also a lot work for maintenance and updating. Using the CSV import allows you to import data comfortably, for example, from Microsoft Excel, OpenOffice or LibreOffice Calc, to *i-doit.*

Contents

Requirements

To be able to use your data for CSV import it is important to save them in **.csv** format with **UTF-8** encoding. Some spreadsheet applications are not able to set the encoding when saving. In this case you can use a text editor which has a conversion function after you exported your data to `.CSV` format. In the following screenshot you can see the **`Convert to UTF-8`** function in Notepad++:



The structure of a suitable `.CSV` file is approximately as shown in the following example:

	A	B	C	D	E	F	G
1	Object Title	Inventory Number	Manufacturer	Model	IP Address	CPU Name	CPU Performance
2	Client 01	ABCD-1000	SYN-Client	Alpha	192.168.10.27	Double-Core	2 GHz
3	Client 02	ABCD-2000	SYN-Client	Alpha	192.168.10.11	Double-Core	2 GHz
4	Client 02					Double-Core XL	3 GHz
5	Client 03	ABCD-3000	SYN-Client	Omega	192.168.10.19	Double-Core	2 GHz
6	Client 04	ABCD-4000	SYN-Client	Omega	192.168.10.86	Double-Core	2 GHz

The first line is used as a header and serves as the identification of the attribute. Each successive line represents an individual object of your IT documentation.
You can also use your favorite text editor to create csv files.

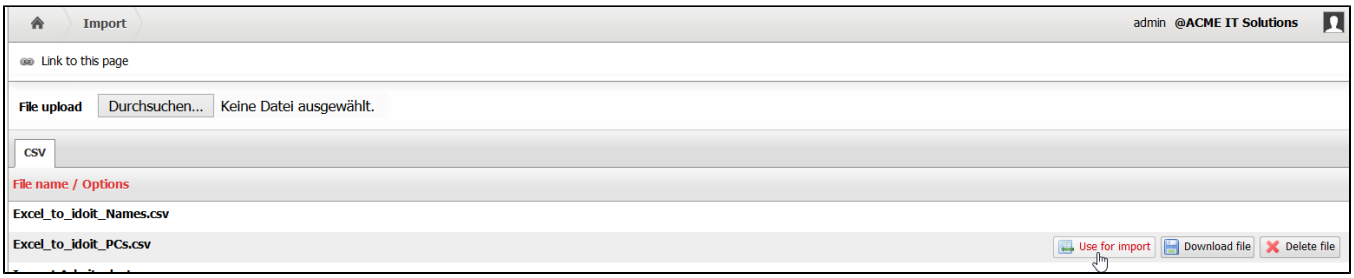
```
1  Object Title;Iventory Number;Manufacturer;Model;IP Address;CPU Name;CPU Performance
2  Client 01;ABCD-1000;SYN-Client;Aplha;192.168.0.27;Quad-Core;4 GHz
3  Client 02;ABCD-2000;SYN-Client;Aplha;192.168.0.11;Quad-Core;3 GHz
4  Client 02;;;;;Quad-Core XL;4 GHz
5  Client 03;ABCD-3000;SYN-Client;Omega;192.168.0.19;Quad-Core;2 GHz
6  Client 04;ABCD-4000;SYN-Client;Omega;192.168.0.86;Quad-Core;2 GHz
```

Once the `.CSV` file has been prepared to this point, you can begin the import.

Upload File

Under **`Extras CMDB Import CSV Import`** you can find the CSV import. You don't need to configure it in advance.

Choose your `.csv` file in the first step by using the **`Browse...`** button and upload the file from your system. Now the file is shown in the list and you can choose the actions **`Use for import`**, **`Download file`** and **`Delete file`** when hovering over the row of the file with the mouse cursor.



To get to the next step of the import you need to click **`Use for import`**.

Set Options

You have to define some options before the mapping process between the table columns and attributes in i-doit can begin.

Object Type

When you select an object type via the drop-down menu **`Global object type`**, all objects in the `.CSV` file will be imported as this type of object. If you do not define a global object type, you require an additional field in the `.CSV` file to enter the database constant. The object type to which the object will be associated is determined by this constant for every line. This allows you to import objects from different object types (client, printer, monitors ...) from one file. This information has to be maintained for all objects if no global object type is used. You can't use a mixed form.

Separators, Headers and Consider default template

If you use a different **`separator`** than the semicolon (`;`), it is possible to specify the used separator.

If you do not use a header line, you can deactivate the header so that the first line in the `.csv` file will be interpreted as the first object.

The objects are created with the data from the default template.

Empty Values

If you want to update existing objects with CSV data import, you can decide how you want to handle blank cells in the `.CSV` file. With the item **`Adopt empty values`** you have the option to choose either **`Yes`** or **`No:`**

- **`Yes`**: Blank cells mean that existing attributes (if available) are overwritten.
- **`No`**: Blank cells are ignored so that existing attributes (if available) are preserved.

Handling of Entries in List Categories (Multi-Valued)

Furthermore, you have to state in which form list categories (multi-valued) appear in the `.csv` file. If no categories of this type are involved in the data import, you can ignore the following options.

What is striking in the example shown above is that “Client 02” appears multiple times. Since this client possesses two CPUs, two entries (one per CPU) need to be generated in the **`CPU`** list category. By using an additional **`line`** the object receives both entries in the **`CPU`** category. It is not required to enter unvarying attributes multiple times ín further lines of the object. This means that you do not have to set inventory number “ABCD-2000” again.

Alternatively, it is possible to store the single entries of list categories either in a **`column`** or in a field as **`comma-separated`** list.

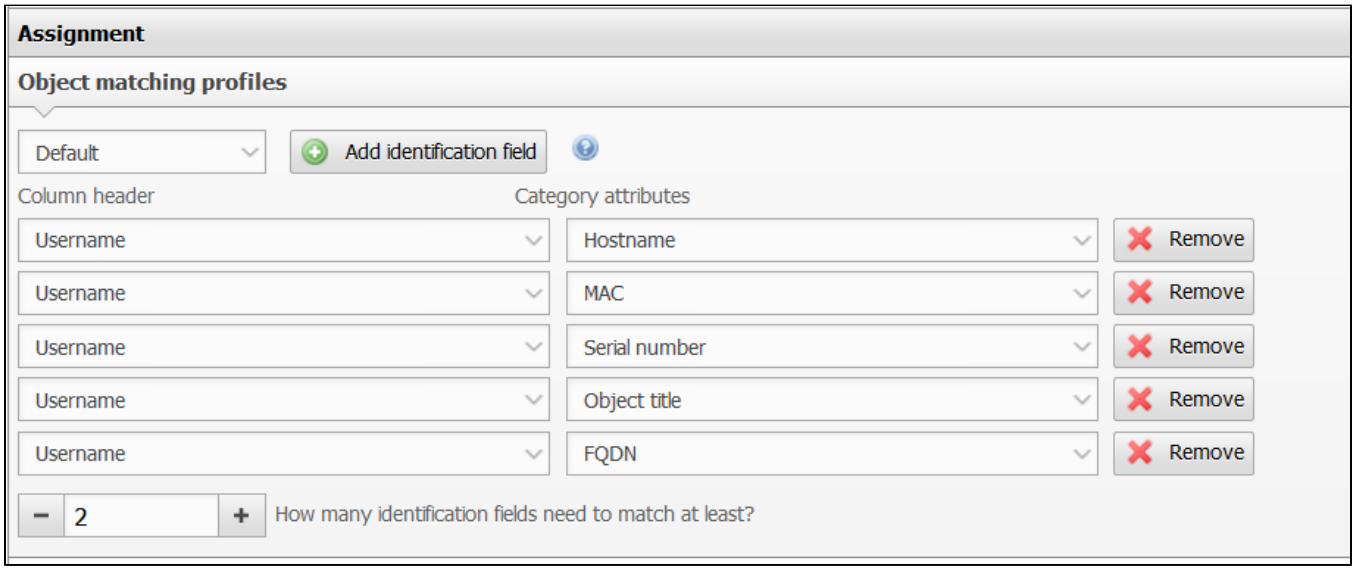
If category entries are already present, you have some more options:

- **`Create category entries only if the category is empty (create if empty)`**
- **`Create category entries and keep existing ones (add)`**
- **`Create category entries and replace existing ones (replace)`**

By clicking the **`Prepare mapping`** button you can reach the next step.

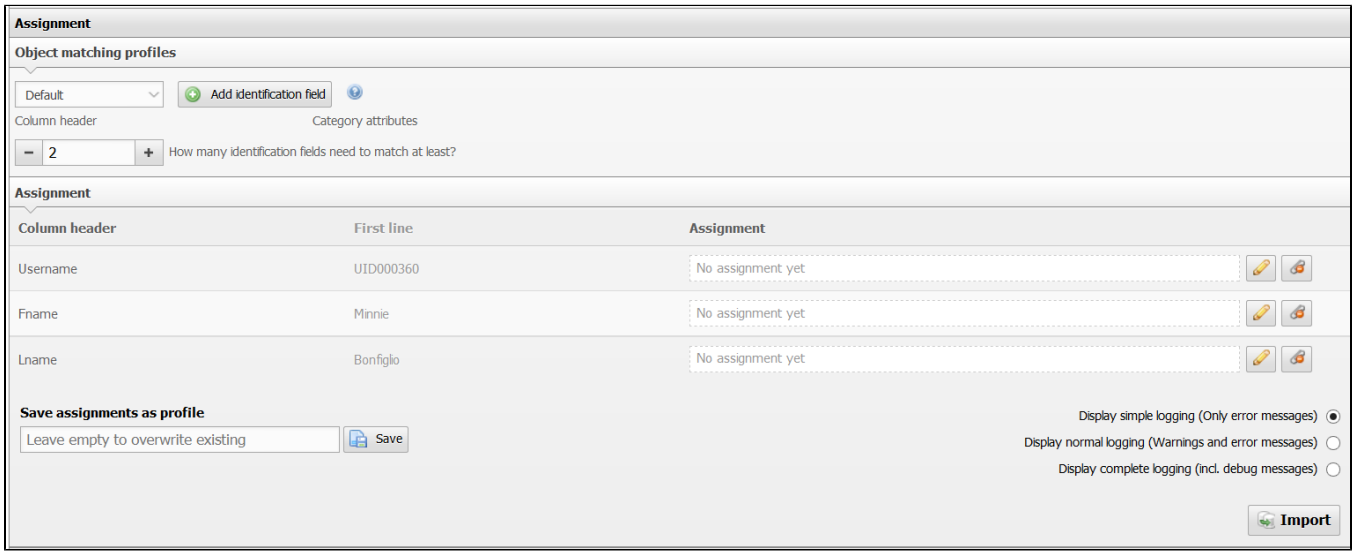
Define Identifying Features

As soon as you have adjusted the options you can start with mapping. First of all you can select an **`object-matching profile`** if you want to update existing objects. You can edit this profile at a later time.



Assignment of Columns to Attributes

Each column receives its own row in the mapping. This way you can link each row of the `.CSV` file with an attribute from *i-doit*. Click the pencil icon of the row in order to activate the input field for selecting the associated attribute. The right attribute can now be selected from the drop-down field or you can enter its name directly in the field to use the suggest feature. The input will be confirmed using the **`Apply`** button. Link each column this way until all allocations are complete. You can remove allocations subsequently. Columns without an allocation will be ignored during data import.



Mandatory details

Stating the object title *and* the object type is both *mandatory*. If you defined a global object type in the data import options, you just have to link the object title with a column in the mapping. If you did not set a global object type, then a link to a column as object type will also be required. Otherwise it is not possible to start the import function. Setting the object type is carried out via its database constant (for example **C_OBJTYPE _SERVER**). Setting the name of the object type (e.g. "**`Server`**") is **not** sufficient.

Creating a Profile

If you wish to import further .CSV files with an identical structure regarding the column allocation, you can save the configuration of the current mapping as a profile. The same applies to updating the currently used file later on and then importing it. Thus you can avoid reccurrent work steps. A saved profile can be selected and loaded or be deleted in the upper area of the options.

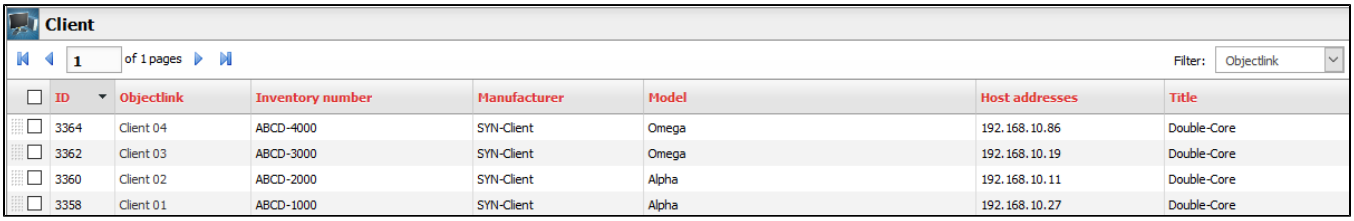
If there is an already matching profile, you can overwrite it without specifying a name.

Start of Data Import

The level of detail for logging the CSV import can be set beneath the mapping. The more extensive the logging is, the more time and resources are needed for the import. The logging of debug messages can be helpful for possibly required troubleshooting.

Use the **Import** button beneath the mapping to start the import. The time needed for the import depends on the extent of the information you wish to import as well as the selected level of logging.

Once the import has been completed, information regarding the import as well as a confirmation of its completion will be indicated. The imported or updated objects are linked directly. The content of these objects can be changed manually anytime, if needed.



Import of Relations (Linking of Objects)

The CSV import is capable of creating links between objects (relations) if these are set via a category. The object that is to be linked can also be put in a column of the .CSV file while the field for linking can be set as attribute which is to be assigned. An example of a statement of the physical location in column H:

	A	B	C	D	E	F	G	H
1	Object Title	Inventory Number	Manufacturer	Model	IP Address	CPU Name	CPU Performance	Location
2	Client 01	ABCD-1000	SYN-Client	Alpha	192.168.10.27	Double-Core	2 GHz	Room 01
3	Client 02	ABCD-2000	SYN-Client	Alpha	192.168.10.11	Double-Core	2 GHz	Room 01
4	Client 02					Double-Core XL	3 GHz	
5	Client 03	ABCD-3000	SYN-Client	Omega	192.168.10.19	Double-Core	2 GHz	Room 02
6	Client 04	ABCD-4000	SYN-Client	Omega	192.168.10.86	Double-Core	2 GHz	Room 02

After selecting the field during the mapping, you can decide whether the selection of the object you are going to link is made in an unrestricted way or if there will be a restriction to an object type of your choice. Furthermore, objects which could not be found can be created automatically. Setting the object type of the linked objects is required in this case so that an object of this type will be created. For some types of links, it is possible to set the attributes in the special assignment by which the identification is made for the objects that are going to be linked. As a standard, the object title is used.

> **Note**
>
> The special assignment cannot be found in all linking fields. If it is not available, only the object title is used for the identification of the object that is to be updated.

Import of Values with Units

Some attributes contain values and units. You have to enter them separately in the corresponding forms of the Web GUI. For example, in the category **Monitor** the **Display** attribute consists of the field for the value and the field for the unit (inch, cm etc.). In order to import this attribute analogous via CSV file import, value and unit have to be together in one cell. Examples:

Category	Attribute	Assignment with CSV file import	Cell in CSV file

CPU	**CPU frequency**	**CPU frequency (unit)**	2.5 GHz
Local mass storage Device	**Capacity**	**Capacity (unit)**	4 TB
Monitor	**Display**	**Display (unit)**	24 Inch

Automated Import of CSV Files

The import of `.CSV` files can be automated using the controller and a cron job. Use the following `csv_import` handler:

```
./controller -u [username] -p [password] -i [client ID] -v -m csv_import [PATH/TO/FILE] [profile ID]
"[separation mark]" [multivalue mode]
```

Replace the placeholders of this example as follows:

- **[username]:** The username of an *i-doit* user who is permitted to use the *i-doit* controller.
- **[password]:** The password of the user mentioned before.
- **[tenant ID]:** The numeric ID of the tenant in which the import will be done.
- **[PATH/TO/FILE]:** The absolute path to the `.CSV` file on the system on which *i-doit* is installed. You can only state one file.
- **[profile ID]:** The ID of the mapping profile which will be used for the import.[1]
- **[separation mark]:** The separation mark which is used in the `.CSV` file to separate the values from each other. The input has to be put in quotation marks (").
- **[multivalue mode]:** The mode that will be used for the import of list categories. The following modes are available:
    - **row** (Statement of multi-value categories in rows)
    - **column** (Statement of multi-value categories in several columns)
    - **comma** (Statement of multi-value categories by using separating commas)

[1] The profile that is going to be used has to be created in the web interface of *i-doit* beforehand. The IDs of your profiles can be shown by the controller by executing it in the following way:

```
./controller -u [username] -p [password] -i [client ID] -v -m csv_import
```

This way the existing profiles of the client (depending on the client ID) are shown additionally to the required parameters if you have already created profiles via the user interface.

As an example, a complete execution of a CSV import looks like this:

```
./controller -u admin -p admin -i 1 -v -m csv_import imports/clients/client_import.csv 2 ";" row
```

According to this example, the user **`admin`** is used for login. He/she uses the password **`admin`** and logs in to the client with the ID **`1`**. The file **`imports/clients/client_import.csv`** and the mapping profile with the ID **`2`** are used for the import. The separator in this file is the semicolon (;) and the list categories have been entered per row.

Now you can use this execution for the periodic automated import of single or multiple `.CSV` files by using it in a cron job. Data imports should not take place at the same time in order to prevent conflicts.