# EXT: External Import Tutorial

Extension Key: externalimport_tut

Language: en

Keywords: forAdmins, forIntermediates

# Table of Contents

# Introduction

This tutorial is designed to help you get started with the “External import” extension (key: external_import). Indeed configuring tables for receiving external data is not quite trivial and this document will hopefully make things clearer by providing examples.

## Questions and support

If you have any questions about this extension, please ask them in the TYPO3 English mailing list (typo3.english), so that others can benefit from the answers. Please use the bug tracker on forge.typo3.org  to report problem or suggest features (http://forge.typo3.org/projects/extension-externalimport_tut/issues).

## Keeping the developer happy

It's unfortunately not possible to rate extensions anymore (maybe that will change again), so feel free to send thanks to the developers or praise the extension in general. If you really want to give something back, you may consider my Amazon wish list: http://www.amazon.co.uk/registry/wishlist/G7DI2AN99Y4F

You may also take a step back and reflect about the beauty of sharing. Think about how much you are benefiting and how much yourself is giving back to the community.

# Installation

This extensions requires the following other extensions:

- external_import
- news
- svconnector_csv (which itself requires svconnector)
- svconnector_feed (ditto)

Upon installation, the extension will create or alter the necessary tables.

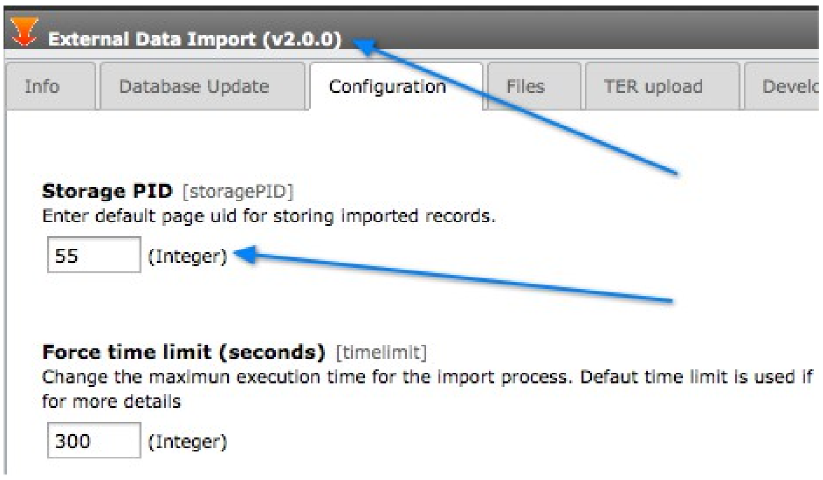
## Upgrading to 1.4.0

As of version 1.4.0, this tutorial relies on extension "news" for the RSS feed import example rather than "tt_news". If you are upgrading don't be surprised if your "tt_news" records don't get updated anymore upon import. If nothing gets imported, it may be because you don't have "news" installed.

Nothing serious, just a little something to pay attention to.

## Verifying the configuration

To ensure that the records inserted or deleted by the external import process don't interfere with the other data in your TYPO3 installation, don't forget to properly define a page for storage in the configuration of the "external_import" extension.

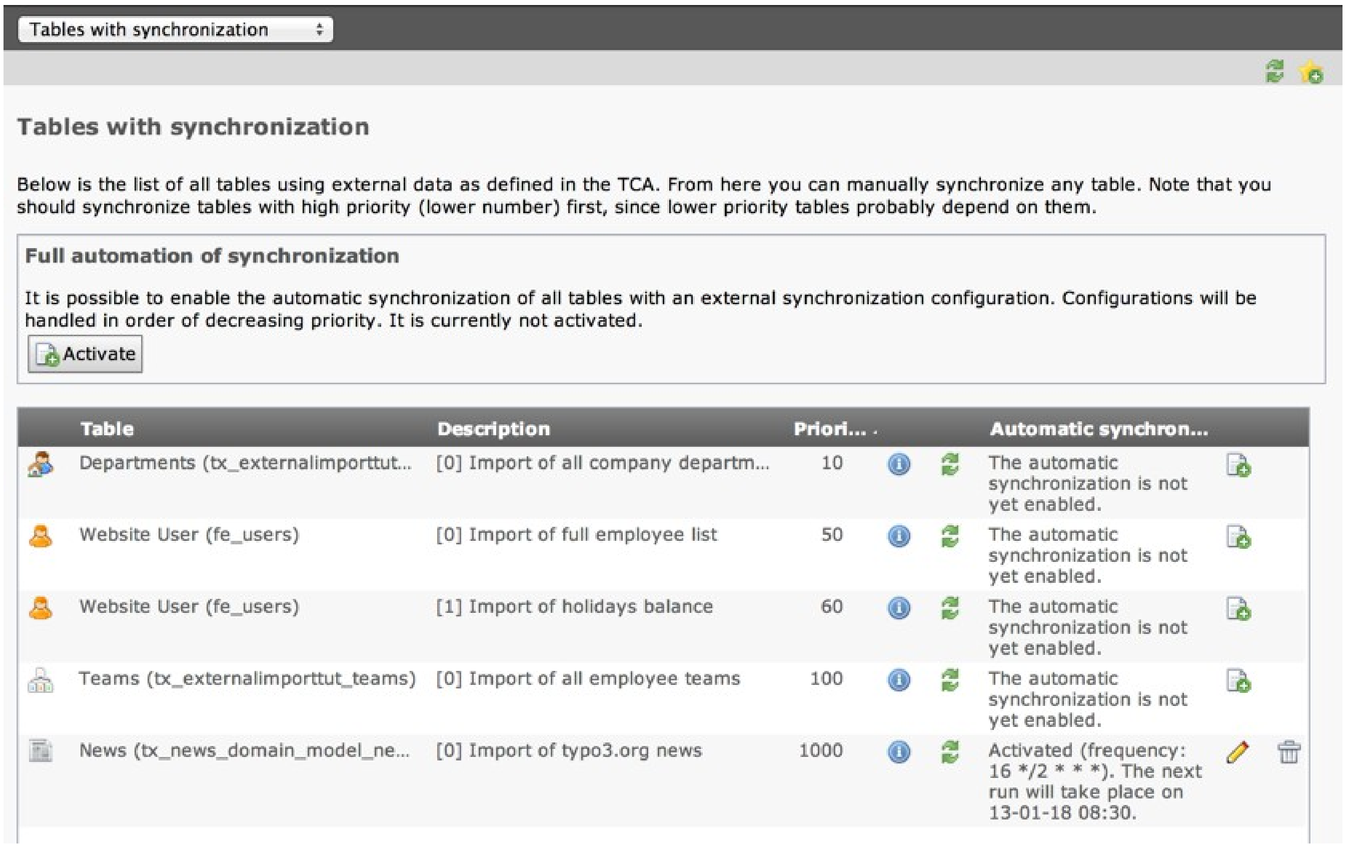


**Storage PID** [storagePID]
Enter default page uid for storing imported records.
55 (Integer)

**Force time limit (seconds)** [timelimit]
Change the maximun execution time for the import process. Defaut time limit is used if
for more details
300 (Integer)

## Verifying the setup

After installation, go to the External Import BE module. It should look like:



### Tables with synchronization

Below is the list of all tables using external data as defined in the TCA. From here you can manually synchronize any table. Note that you should synchronize tables with high priority (lower number) first, since lower priority tables probably depend on them.

**Full automation of synchronization**

It is possible to enable the automatic synchronization of all tables with an external synchronization configuration. Configurations will be handled in order of decreasing priority. It is currently not activated.

| Table | Description | Priori... | Automatic synchron... |
|---|---|---|---|
| Departments (tx_externalimporttut... | [0] Import of all company departm... | 10 | The automatic synchronization is not yet enabled. |
| Website User (fe_users) | [0] Import of full employee list | 50 | The automatic synchronization is not yet enabled. |
| Website User (fe_users) | [1] Import of holidays balance | 60 | The automatic synchronization is not yet enabled. |
| Teams (tx_externalimporttut_teams) | [0] Import of all employee teams | 100 | The automatic synchronization is not yet enabled. |
| News (tx_news_domain_model_ne... | [0] Import of typo3.org news | 1000 | Activated (frequency: 16 */2 * * *). The next run will take place on 13-01-18 08:30. |

If you want to follow the process in more details, it is advised to activate the debug mode of "external_import" and to install an extension capable of using the Developer's Log (such as “devlog”).

# The scenario

## Main scenario

The basic scenario of this tutorial is the following: we have flat files exported from some other system, e.g. a company ERP. These exports contains a list of employees, a list of departments, a list of holidays balances for each employees and a list of teams along with their members (employees). The goal is to import all this data into TYPO3 tables all the while maintaining the relationships.

The departments and the teams will go into new tables. The employees should go into the fe_users table, so that each employee can then log into the corporate intranet powered by TYPO3. One implication is that user names and passwords must be created on the fly for each new employee.

One particular constraint is the order in which the data is imported. If we want to keep the relationship between employees and their departments, we must make sure that the departments are imported **before** the employees. On the other hand the relationship between employees and their teams is found inside the teams file. So teams must be imported **after** the employees.

## Second scenario

In order to be more complete, this tutorial proposes a second workflow: in this other case we want to import an RSS feed into the tx_news_domain_model_news table. This scenario demonstrates the processing of an XML input.

# Running the employees import

This chapter describes how each table was defined (or modified) and what TCA configuration was used for the import of the external data to be successful. The results of the imports are discussed.

## The departments

Since the departments table does not exist inside TYPO3, we create it with a full SQL statement and a full TCA definition. The SQL looks like this:

```
CREATE TABLE tx_externalimporttut_departments (
      uid int(11) NOT NULL auto_increment,
      pid int(11) DEFAULT '0' NOT NULL,
      tstamp int(11) DEFAULT '0' NOT NULL,
      crdate int(11) DEFAULT '0' NOT NULL,
      cruser_id int(11) DEFAULT '0' NOT NULL,
      deleted tinyint(4) DEFAULT '0' NOT NULL,
      hidden tinyint(4) DEFAULT '0' NOT NULL,
      code varchar(4) DEFAULT '' NOT NULL,
      name varchar(255) DEFAULT '' NOT NULL,

      PRIMARY KEY (uid),
      KEY parent (pid)
);
```

and the TCA looks like this (without the "columns" section, which is in the dynamic configuration file):

```
$TCA['tx_externalimporttut_departments'] = array(
      'ctrl' => array(
            'title'     =>
'LLL:EXT:externalimport_tut/locallang_db.xml:tx_externalimporttut_departments',
            'label'     => 'name',
            'tstamp'    => 'tstamp',
            'crdate'    => 'crdate',
            'cruser_id' => 'cruser_id',
            'default_sortby' => 'ORDER BY name',
            'delete' => 'deleted',
            'enablecolumns' => array(
                  'disabled' => 'hidden',
            ),
            'dynamicConfigFile' => t3lib_extMgm::extPath($_EXTKEY).'tca.php',
            'iconfile'          =>
t3lib_extMgm::extRelPath($_EXTKEY).'icon_tx_externalimporttut_departments.gif',
            'external' => array(
                        0 => array(
                                    'connector' => 'csv',
                                    'parameters' => array(
                                                'filename' => t3lib_extMgm::extPath($_EXTKEY,
'res/departments.txt'),
                                                'delimiter' => "\t",
                                                'text_qualifier' => '"',
                                                'skip_rows' => 1,
                                                'encoding' => 'latin1'
                                    ),
                                    'data' => 'array',
                                    'reference_uid' => 'code',
                                    'priority' => 10,
                                    'description' => 'Import of all company departments'
                        )
            )
      ),
);
```

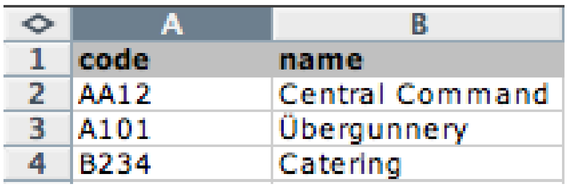
Compared to a traditional "ctrl" section, this TCA declaration contains an "external" sub-section which is used to described the external source from which the data will come. This external sub-section is itself an indexed array (note index "0" in the TCA extract above). Several indices are used when a table is synchronised with multiple external sources. We will see with with the fe_users table.

The first property used above is "connector". This defines the sub-type of connector service that is needed for connecting to and reading data from the external source. In this case we are reading flat files, so we request an instance of a connector service of sub-type "csv", which is able to read such files.

Next is the "parameters" property. This is an array of values that are passed to the connector. What values need to be defined is dependent upon the sub-type of connector service. In the case of the "csv" sub-type, the parameters include the name of the file to read, what delimiter is used to separate the columns (a tab in this case), what character is used to surround strings (a double quote in this case), how many rows must be skipped off the top of the file (generally because they contain header information, and it will generally be only 1 line) and – finally – what is the encoding of the file. This last information will enable the CSV connector to convert the file's data as appropriate if the file's charset does not match your BE's charset.

Then the "data" property indicates in what format the data will be provided by the connector. "array" means that it will be a PHP array. The "reference_uid" property indicates in which field from the departments table the primary key from the external source will be stored.

This is a critical information. Let's look at the contents of the "departments.txt" file:



|  | A | B |
|---|---|---|
| 1 | code | name |
| 2 | AA12 | Central Command |
| 3 | A101 | Übergunnery |
| 4 | B234 | Catering |

Each department has a code. It is this code that makes a department unique in the external data. If we want to be able to keep track of which item in the external source is new, which has already been imported at least once and which doesn't exist anymore, we have to know what primary key is used in the external data and store it internally, so that we can check it upon the next synchronisation.

The next property is "priority". As was mentioned before, external data needs to be imported in a precise order if relationships between tables are to be preserved. The priority property takes care of that. Lower priorities go first.

Finally the "description" property is used to stored some useful information about that particular configuration. This information will be displayed in the BE module and is there only for reference. So make it relevant.

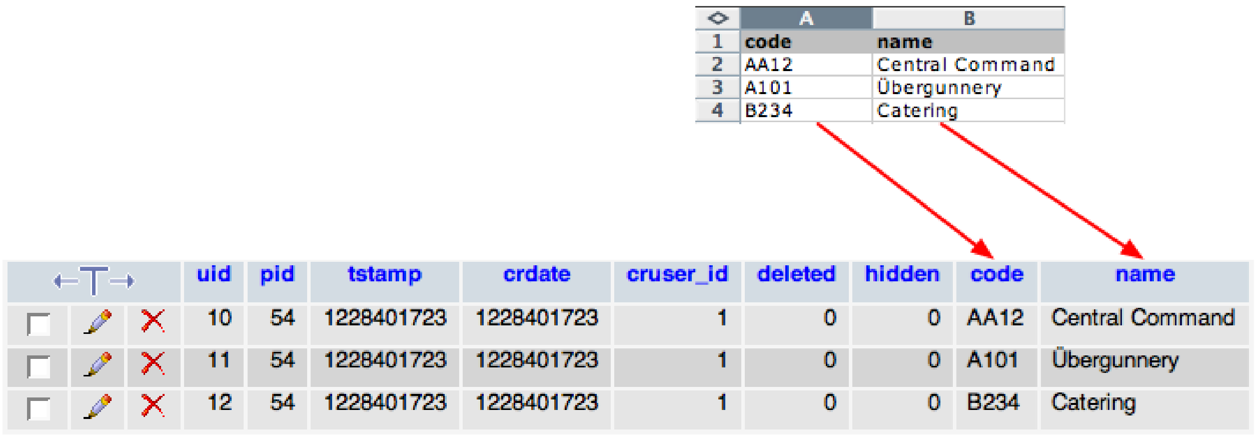
The next step is to defined external information for each column. Indeed this is where the real mapping takes places: which column in the external data will fit into which field in the internal database. This is how it looks for the departments table (full TCA included for having the information in its context, just concentrate on the parts in bold):

```
$TCA['tx_externalimporttut_departments'] = array(
      'ctrl' => $TCA['tx_externalimporttut_departments']['ctrl'],
      'interface' => array(
            'showRecordFieldList' => 'hidden,code,name'
      ),
      'feInterface' => $TCA['tx_externalimporttut_departments']['feInterface'],
      'columns' => array(
            'hidden' => array(
                  'exclude' => 1,
                  'label'   => 'LLL:EXT:lang/locallang_general.xml:LGL.hidden',
                  'config'  => array(
                        'type'    => 'check',
                        'default' => '0'
                  )
            ),
            'code' => array(
                  'exclude' => 0,
                  'label' =>
'LLL:EXT:externalimport_tut/locallang_db.xml:tx_externalimporttut_departments.code',

                  'config' => array(
                        'type' => 'input',
                        'size' => '10',
                        'max' => '4',
                        'eval' => 'required,trim',
                  ),
                  'external' => array(
                        0 => array(
                                    'field' => 'code'
                        )
                  )
            ),
            'name' => array(
                  'exclude' => 0,
                  'label' =>
'LLL:EXT:externalimport_tut/locallang_db.xml:tx_externalimporttut_departments.name',

                  'config' => array(
                        'type' => 'input',
                        'size' => '30',
                        'eval' => 'required,trim',
                  ),
                  'external' => array(
                        0 => array(
                                    'field' => 'name'
                              )
                        )
                  ),
            ),
            'types' => array(
                  '0' => array('showitem' => 'hidden;;1;;1-1-1, code, name')
            ),
            'palettes' => array(
                  '1' => array('showitem' => '')
            )
);
```

The departments table is quite simple and is comprised of only three fields beyond the usual complement of TYPO3 fields (uid, pid, etc.). The "hidden" field is not mapped to the external data. The "code" field is mapped to a similarly named field in the external data (with the "field" property). The same goes for the "name" field. This mapping is quite simple. We will see more complicated configurations with the other tables. Note that the names used for the fields ("code" and "name") match the names used in the first row of the file. This is how fields from the file can be matched to columns from the database.

The important thing to note here is that the indices used in the "external" configuration for each field match the indices used in the "external" configuration found inside the "ctrl" section. This is crucial. If the indices don't match the different bits of "external" configurations will not know to which other bits they relate.

At this point you can run the import of the departments table. The result that you should have can be represented like this:



| | A | B |
|---|---|---|
| 1 | code | name |
| 2 | AA12 | Central Command |
| 3 | A101 | Übergunnery |
| 4 | B234 | Catering |

| | uid | pid | tstamp | crdate | cruser_id | deleted | hidden | code | name |
|---|---|---|---|---|---|---|---|---|---|
| | 10 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | AA12 | Central Command |
| | 11 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | A101 | Übergunnery |
| | 12 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | B234 | Catering |

## The employees (and their holidays)

The list of employees will be stored in the fe_users table, which must be extended to add the necessary fields:

```
CREATE TABLE fe_users (
      tx_externalimporttut_code varchar(10) DEFAULT '' NOT NULL,
      tx_externalimporttut_department text,
      tx_externalimporttut_holidays int(11) DEFAULT '0' NOT NULL,
);
```

These new columns are added to the TCA of the fe_users table. At this point we don't yet set the external data for these columns, as we will do it later for all relevant columns. As this is a standard TCA operation, it is not repeated here and can be simply looked up in the ext_tables.php file.

Next we add the external information to the “ctrl” section of the fe_users table:

```
$TCA['fe_users']['ctrl']['external'] = array(
      0 => array(
            'connector' => 'csv',
            'parameters' => array(
                  'filename' => t3lib_extMgm::extPath($_EXTKEY, 'res/employees.txt'),
                  'delimiter' => ';',
                  'text_qualifier' => '',
                  'skip_rows' => 1,
                  'encoding' => 'utf8'
            ),
            'data' => 'array',
            'reference_uid' => 'tx_externalimporttut_code',
            'additional_fields' => 'last_name,first_name',
            'priority' => 50,
            'disabledOperations' => '',
            'enforcePid' => true,
            'description' => 'Import of full employee list'
      ),
      1 => array(
            'connector' => 'csv',
            'parameters' => array(
                  'filename' => t3lib_extMgm::extPath($_EXTKEY, 'res/holidays.txt'),
                  'delimiter' => ',',
                  'text_qualifier' => '',
                  'skip_rows' => 0,
                  'encoding' => 'utf8'
            ),
            'data' => 'array',
            'reference_uid' => 'tx_externalimporttut_code',
            'priority' => 60,
            'disabledOperations' => 'insert,delete',
            'description' => 'Import of holidays balance'
      )
);
```

The first thing to note is that there are 2 external configurations in this case. As was described in the description of the scenario, the fe_users users table will be synchronised with the employees list and with a second file containing the balance of holidays. Two things are worth of notice in the first configuration:

1. the use of the “additional_fields” property: two fields from the external data (“last_name” and “first_name”) will be used during the import for assembling the user's full name and its password. So we need to keep those two fields

for calculations, but they won't be stored at the end of the process.

2. The “enforcePid” property is set to true so that not all fe_users records will be affected be the import process. If some fe_users are stored in a different page than the one where the imported records are stored, those records will not be considered for updates or deletion. This makes it possible to have several sources of fe_users without interference with one another.

In the second configuration, we make use of the “disabledOperations” property. Indeed the holidays balance file will contain only information about the number of holidays still available for each employee. It does not contain complete information so it cannot be used as a reference for creating new users. Hence the “insert” operations is disabled. Since it is not a reference anyway, it does not make sense to allow this particular synchronisation to delete users. So the “delete” operation is also disabled.

Finally we set the external configuration for each column that will receive external data.

```
$TCA['fe_users']['columns']['name']['external'] = array(
      0 => array(
            'field' => 'last_name',
            'userFunc' => array(
                  'class' =>
'EXT:externalimport_tut/class.tx_externalimporttut_transformations.php:&tx_externalimporttut_transformat
ions',
                  'method' => 'assembleName'
            )
      )
);
$TCA['fe_users']['columns']['username']['external'] = array(
      0 => array(
            'field' => 'last_name',
            'userFunc' => array(
                  'class' =>
'EXT:externalimport_tut/class.tx_externalimporttut_transformations.php:&tx_externalimporttut_transformat
ions',
                  'method' => 'assembleUserName',
                  'params' => array(
                        'encoding' => 'utf8'
                  )
            )
      )
);
$TCA['fe_users']['columns']['starttime']['external'] = array(
      0 => array(
            'field' => 'start_date',
            'userFunc' => array(
                  'class' =>
'EXT:external_import/samples/class.tx_externalimport_transformations.php:&tx_externalimport_transformati
ons',
                  'method' => 'parseDate'
            )
      )
);
$TCA['fe_users']['columns']['tx_externalimporttut_code']['external'] = array(
      0 => array(
            'field' => 'employee_number'
      ),
      1 => array(
            'field' => 0
      )
);
$TCA['fe_users']['columns']['email']['external'] = array(
      0 => array(
            'field' => 'mail'
      )
);
$TCA['fe_users']['columns']['telephone']['external'] = array(
      0 => array(
            'field' => 'phone'
      )
);
$TCA['fe_users']['columns']['company']['external'] = array(
      0 => array(
            'value' => 'The Empire'
      )
);
$TCA['fe_users']['columns']['title']['external'] = array(
      0 => array(
            'field' => 'rank',
            'mapping' => array(
                  'valueMap' => array(
                        '1' => 'Captain',
                        '2' => 'Senior',
                        '3' => 'Junior'
                  )
            ),
            'excludedOperations' => 'update'
      )
);
$TCA['fe_users']['columns']['tx_externalimporttut_department']['external'] = array(
      0 => array(
            'field' => 'department',
            'mapping' => array(
                  'table' => 'tx_externalimporttut_departments',
                  'reference_field' => 'code'
            )
      )
);
$TCA['fe_users']['columns']['tx_externalimporttut_holidays']['external'] = array(
      1 => array(
            'field' => 1
      )
);
```

Several columns have more interesting configurations than the departments table described before. They have been highlighted in bold. The first three fields will use a user function. The user functions are defined using a "class" property and a "method" property. Additional parameters can be passed to the function using the "parameters" property. So what happens for these three fields?

1. For the "name" field, a method called `assembleName()` will be called, from a class defined in this tutorial extension. Let's look at what this method does:

   ```
   function assembleName($record, $index, $params) {
         $fullName = $record['last_name'].' '.$record['first_name'];
         return $fullName;
   }
   ```

   The method receives the record being handled, so that all fields (mapped fields and additional fields) from the external data are available for calculations. The `$index` argument contains the key of the field that is to be affected by the transformation. The third argument is an array containing additional parameters. In this case it is not used.

   To obtain the user's full name we just concatenate the values from the "last_name" and "first_name" external fields. This value is returned as the method's result.

2. For the "username" field a similar method is called, but which takes extra care to return a viable user name, i.e. converting any character that is not strict ASCII and stripping other inappropriate character. Note that this is just an example. A real-world implementation of such a method would also check that the generated user name is unique.

3. The "starttime" field is mapped to the external "start_date". However that date is stored in a "yyyy-mm-dd" format, which is not convenient for storing in the "starttime" field. We convert to a timestamp using a sample user function provided by the external_import extension itself. This method can perform several transformations, but it returns a simple timestamp when called without parameters, as is the case here.

4. The "company" field is actually not filled with values coming from the external source, but with a fixed value. This is achieved by using the "value" property instead of the "field" property. In this example, the "company" field for every fe_users record will contain the value "The Empire".

5. The same goes for the "title" field, but a bit more sophisticated. In this case the values from the external source are matched to other values using a simple array. For example, if the external data is "1", the title will be "Captain". This way we can avoid creating a separate table for titles, assuming there are only a few and they don't change often. Furthermore we decided that this field should not be updated (using the "excludedOperations" property). This means that this field will be written when a new record is created, but will be left untouched during further updates. That way the field can be safely modified from within TYPO3 and changes will not be overwritten.

6. Last but not least, the "tx_externalimporttut_department" will need to relate to the department the employee works in. Now we don't want to use the primary key of the external data for departments as a foreign key in the fe_users table. We want the uid from the departments as they were inserted into the TYPO3 database. This is the task of the "mapping" property. The first sub-property – "table" – is used to point to the table where the records are stored inside the TYPO3 database. The second sub-property – "reference_field" – indicates in which field from that table the external primary key for departments has been stored.
   What will happen during import is that the mapping function will build a hash table relating the external primary keys from the departments table ("code" column) to the internal primary keys ("uid" column). This hash table is then used to find out the foreign keys for the fe_users.

One more operation happens during the import process, but is not visible in the TCA. In the ext_localconf.php file, we make use of the "insertPreProcess" hook:

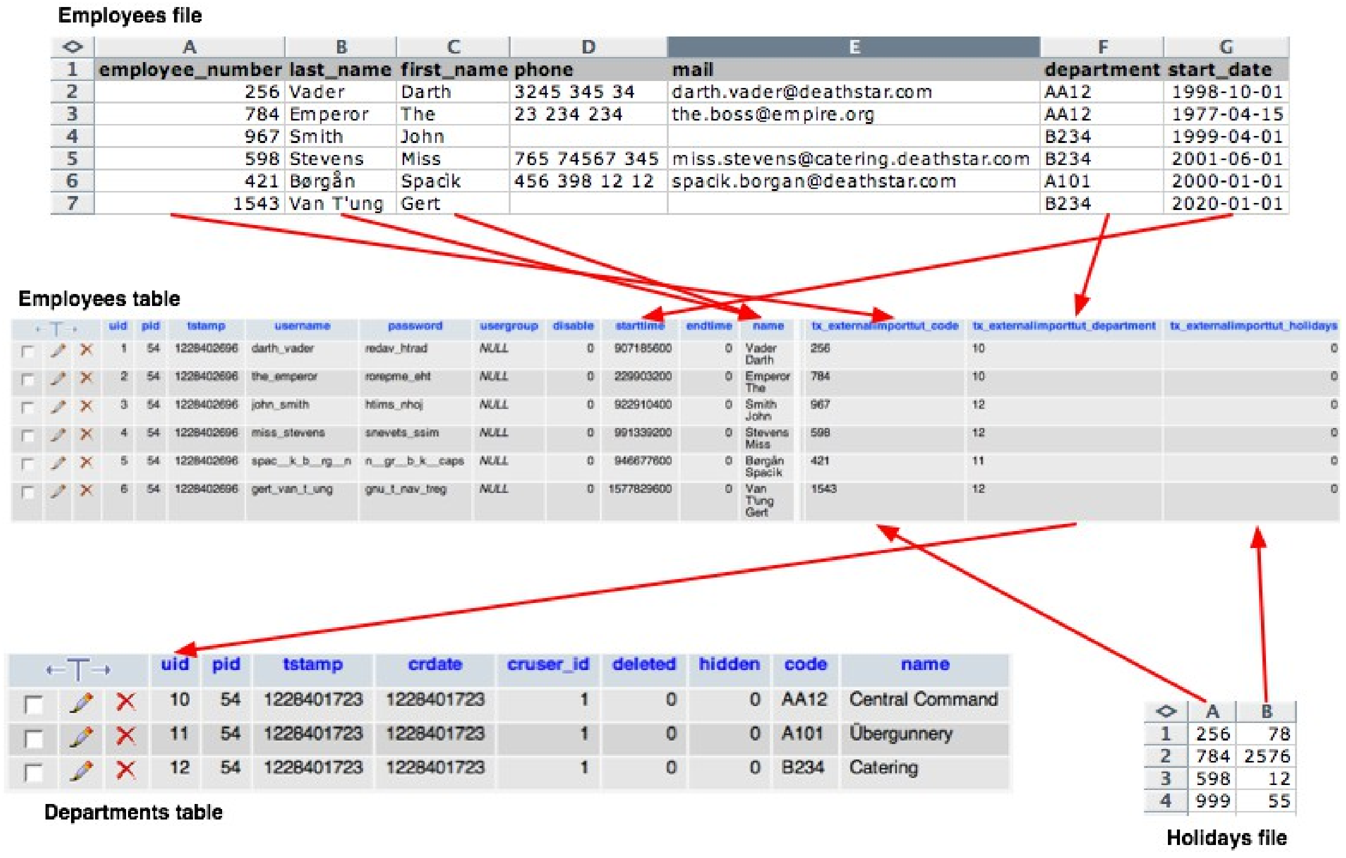
```
$TYPO3_CONF_VARS['EXTCONF']['external_import']['insertPreProcess'][] =
'EXT:externalimport_tut/class.tx_externalimporttut_hooks.php:&tx_externalimporttut_hooks';
```

This makes it possible to add an automatically generated password to the data to be stored, but only in the case when it is a new user (insert operation). To be really clean we could also make use of the "updatePreProcess" hook to remove the username field from the records to be updated, as we don't really want to change the username automatically. This is left as an exercise for the reader. You may also want to make sure that new users belong to some default fe_group.

All the external configuration shown above also included information for importing the holidays balance. There are a couple of things worth noticing:

1. The file "holidays.txt" does not contain a header row. Thus it is not possible to use field names for mapping the external data. Instead we have to rely on column numbers. So "tx_externalimporttut_code" is matched to field 0 and " tx_externalimporttut_holidays" is matched to field 1.
2. Strictly speaking it is not necessary to store the employee number again in the "tx_externalimporttut_code" column, so you might think that this mapping could be dropped. It is however necessary to keep it, because this is how existing records will be detected (by matching the value imported into the "tx_externalimporttut_code" column to the external primary keys already stored in the database).

If you now run the employees and then the holidays synchronisations, you should end up with a situation that can be represented like this (sorry, it's a bit small but hopefully still readable):



Employees file

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | employee_number | last_name | first_name | phone | mail | department | start_date |
| 2 | 256 | Vader | Darth | 3245 345 34 | darth.vader@deathstar.com | AA12 | 1998-10-01 |
| 3 | 784 | Emperor | The | 23 234 234 | the.boss@empire.org | AA12 | 1977-04-15 |
| 4 | 967 | Smith | John | | | B234 | 1999-04-01 |
| 5 | 598 | Stevens | Miss | 765 74567 345 | miss.stevens@catering.deathstar.com | B234 | 2001-06-01 |
| 6 | 421 | Børgån | Spacik | 456 398 12 12 | spacik.borgan@deathstar.com | A101 | 2000-01-01 |
| 7 | 1543 | Van T'ung | Gert | | | B234 | 2020-01-01 |

Employees table

| uid | pid | tstamp | username | password | usergroup | disable | starttime | endtime | name | tx_externalimporttut_code | tx_externalimporttut_department | tx_externalimporttut_holidays |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 54 | 1228402696 | darth_vader | redav_htrad | NULL | 0 | 907185600 | 0 | Vader Darth | 256 | 10 | 0 |
| 2 | 54 | 1228402696 | the_emperor | rorepme_eht | NULL | 0 | 229903200 | 0 | Emperor The | 784 | 10 | 0 |
| 3 | 54 | 1228402696 | john_smith | htims_nhoj | NULL | 0 | 922910400 | 0 | Smith John | 967 | 12 | 0 |
| 4 | 54 | 1228402696 | miss_stevens | snevets_ssim | NULL | 0 | 991339200 | 0 | Stevens Miss | 598 | 12 | 0 |
| 5 | 54 | 1228402696 | spac__k_b__rg__n | n__gr__b_k__caps | NULL | 0 | 946677600 | 0 | Børgån Spacik | 421 | 11 | 0 |
| 6 | 54 | 1228402696 | gert_van_t_ung | gnu_t_nav_treg | NULL | 0 | 1577829600 | 0 | Van T'ung Gert | 1543 | 12 | 0 |

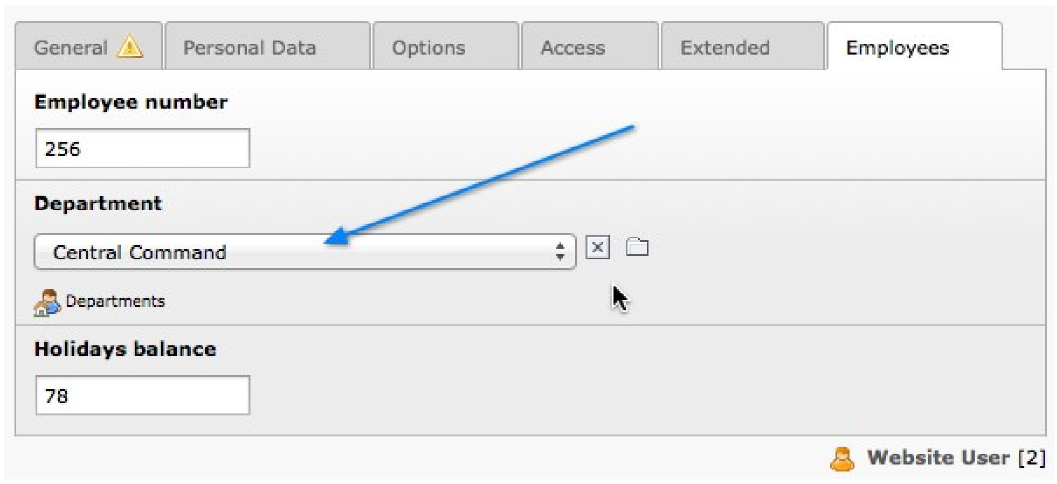
| uid | pid | tstamp | crdate | cruser_id | deleted | hidden | code | name |
|---|---|---|---|---|---|---|---|---|
| 10 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | AA12 | Central Command |
| 11 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | A101 | Übergunnery |
| 12 | 54 | 1228401723 | 1228401723 | 1 | 0 | 0 | B234 | Catering |

Departments table

| | A | B |
|---|---|---|
| 1 | 256 | 78 |
| 2 | 784 | 2576 |
| 3 | 598 | 12 |
| 4 | 999 | 55 |

Holidays file

Most importantly we can see that the "tx_externalimporttut_department" column contains foreign keys that correspond to the internal (TYPO3) primary keys of the departments table. If you open a fe_user record in the TYPO3 BE, you will see that it cleanly relates to a department.

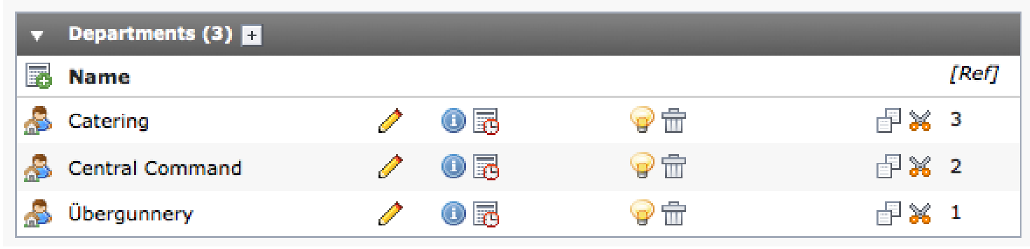


Employee number
256

Department
Central Command

Holidays balance
78

And since the data manipulation operations rely on TCEmain the reference index has been kept up to date, as you can see by looking at the departments in mode Web > List (e.g. "Catering" should have a reference count of 3).



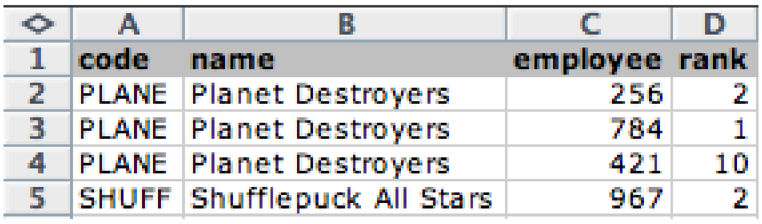
Departments (3)

| Name | [Ref] |
|---|---|
| Catering | 3 |
| Central Command | 2 |
| Übergunnery | 1 |

## The teams

The last data to be imported is the teams. This is mostly like departments, except that teams have a many-to-many relationship to employees. Indeed a team will be comprised of several employees and any employee may be part of several teams.

When such data comes along, the external_import extension expects it to be denormalised. This means that if team A contains 3 employees, there will be 3 entries for team A, each with a relationship to a different employee. Let's look at the example data:



|  | A | B | C | D |
|---|---|---|---|---|
| 1 | code | name | employee | rank |
| 2 | PLANE | Planet Destroyers | 256 | 2 |
| 3 | PLANE | Planet Destroyers | 784 | 1 |
| 4 | PLANE | Planet Destroyers | 421 | 10 |
| 5 | SHUFF | Shufflepuck All Stars | 967 | 2 |

We clearly see that the "Planet Destroyers" team appears three times, because it is comprised of employees 256, 421 and 784. External import takes this into account by making sure that it keeps a single copy of each team, based on the external primary key (the "code" field in this case).

It is not currently possible to create MM-relationships with data in some other format (for example, the comma-separated list of keys, commonly used in TYPO3).

In the example data above, you can see that there's a "rank" field. This will be used to set the "sorting" column in the MM-relations table.

The SQL for the teams table is not repeated here as it is quite standard. The MM-relations table is also an absolutely standard TYPO3 table for MM-relations:

```
CREATE TABLE tx_externalimporttut_teams_feusers_mm (
      uid_local int(11) DEFAULT '0' NOT NULL,
      uid_foreign int(11) DEFAULT '0' NOT NULL,
      sorting int(11) DEFAULT '0' NOT NULL,
      KEY uid_local (uid_local),
      KEY uid_foreign (uid_foreign)
);
```

The "external" definition of the team's table "ctrl" section is also not repeated here as it does not contain anything not already covered in this tutorial. The really interesting part is the "external" information for creating the relationship between teams and fe_users. The definition is to be found in the "members" column of the teams table:

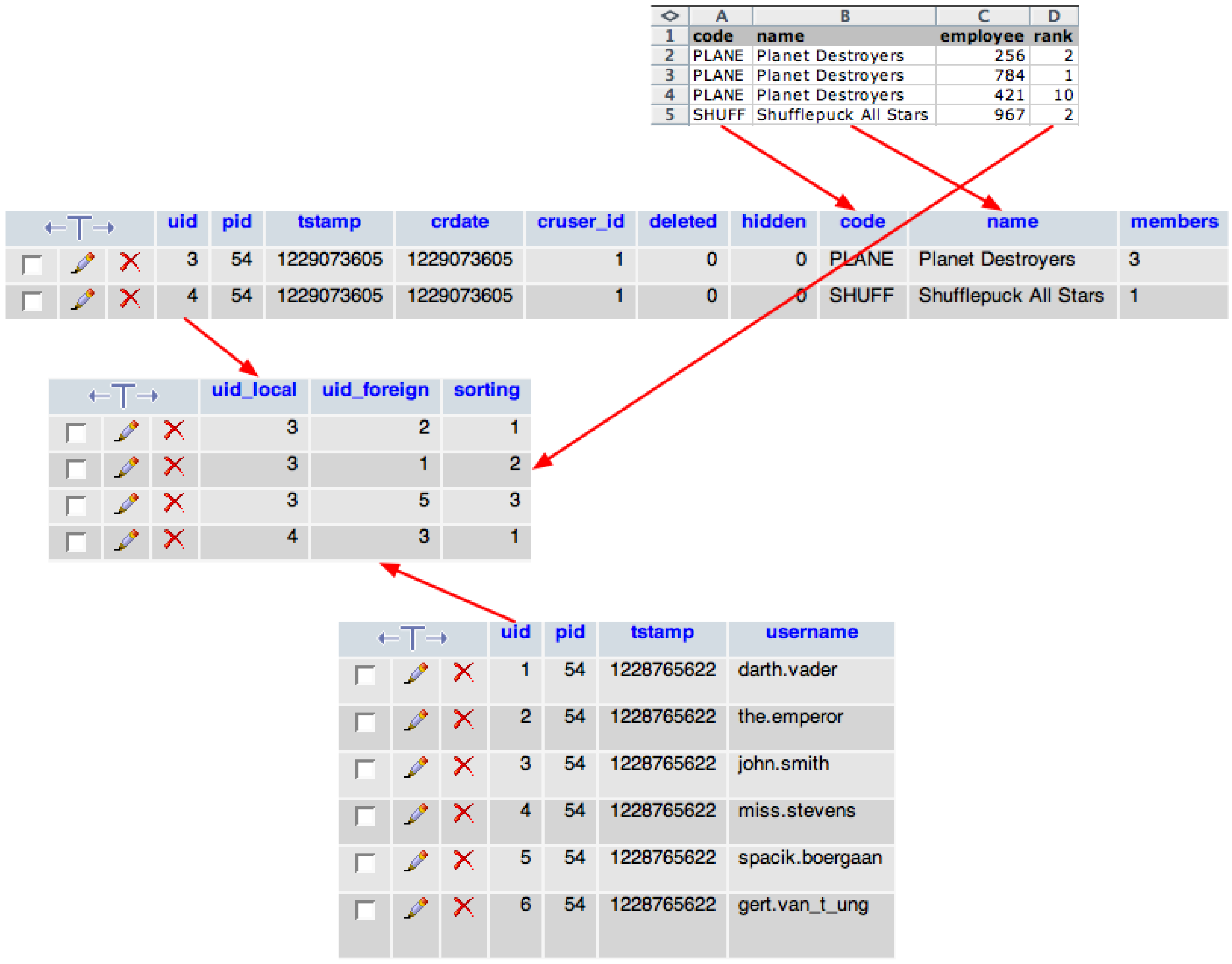
```
$TCA['tx_externalimporttut_teams'] = array(
      ...
      'columns' => array(
            …
            'members' => array(
                  'exclude' => 0,
                  'label' =>
'LLL:EXT:externalimport_tut/locallang_db.xml:tx_externalimporttut_teams.members',
                  'config' => array(
                        'type' => 'group',
                        'size' => 5,
                        'internal_type' => 'db',
                        'allowed' => 'fe_users',
                        'MM' => 'tx_externalimporttut_teams_feusers_mm',
                        'maxitems' => 100
                  ),
                  'external' => array(
                        0 => array(
                              'field' => 'employee',
                              'MM' => array(
                                    'mapping' => array(
                                          'table' => 'fe_users',
                                          'reference_field' => 'tx_externalimporttut_code',
                                    ),
                                    'sorting' => 'rank'
                              )
                        )
                  )
            ),
      ),
      …
);
```

Looking at the TCA for this column, you can see that it contains the traditional information for a MM column. The external part describes the column as pointing to the "employee" field, which contains the employee number as we saw above. It then contains a "MM" section which describes how the relationships can be created on the TYPO3 side. The main part is the "mapping" information which is similar in syntax to what we saw for simply mapped fields (a.k.a one-to-many relationships). It references a table and the column in that table which contains the external primary key.

Additionally it is possible to choose a field from the external data that will be used to define the sorting of the relationships. In this case it is the "rank" field. If this is not defined, the sorting will be simply based on the first in, first out basis (i.e. the first record will have a sorting of 1, the second a sorting of 2, etc.).

Note that it is currently not possible to define a sorting field for the "sorting_foreign" column, should you have such a configuration. Sorting is always relative to the "uid_local" column.

After running the teams import, you should get something like this:



| | A | B | C | D |
|---|---|---|---|---|
| 1 | code | name | employee | rank |
| 2 | PLANE | Planet Destroyers | 256 | 2 |
| 3 | PLANE | Planet Destroyers | 784 | 1 |
| 4 | PLANE | Planet Destroyers | 421 | 10 |
| 5 | SHUFF | Shufflepuck All Stars | 967 | 2 |

| uid | pid | tstamp | crdate | cruser_id | deleted | hidden | code | name | members |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 54 | 1229073605 | 1229073605 | 1 | 0 | 0 | PLANE | Planet Destroyers | 3 |
| 4 | 54 | 1229073605 | 1229073605 | 1 | 0 | 0 | SHUFF | Shufflepuck All Stars | 1 |

| uid_local | uid_foreign | sorting |
|---|---|---|
| 3 | 2 | 1 |
| 3 | 1 | 2 |
| 3 | 5 | 3 |
| 4 | 3 | 1 |

| uid | pid | tstamp | username |
|---|---|---|---|
| 1 | 54 | 1228765622 | darth.vader |
| 2 | 54 | 1228765622 | the.emperor |
| 3 | 54 | 1228765622 | john.smith |
| 4 | 54 | 1228765622 | miss.stevens |
| 5 | 54 | 1228765622 | spacik.boergaan |
| 6 | 54 | 1228765622 | gert.van_t_ung |

We can see that the teams were properly related to the fe_users. The sorting has also been kept correctly although with a renumbering (done automatically by TCEmain).

# Running the RSS import

Our second scenario is about importing an RSS feed into the tx_news_domain_model_news table (from the "news" extension). As an example we will take the main RSS feed from typo3.org in Atom format: http://typo3.org/xml-feeds/rss.xml.

## The data to import

Here's an excerpt of the file (of course, the exact content will vary, since news keep being added to the feed):

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:content="http://purl.org/rss/1.0/modules/content/">
      <channel>
            <title>typo3.org: Latest News</title>
            <link>http://typo3.org</link>
            <description>Latest news from typo3.org</description>
            <language>en</language>
            <image>
                  <title>typo3.org: Latest News</title>
                  <url>http://typo3.org/clear.gif</url>
                  <link>http://typo3.org</link>
                  <description>Latest news from typo3.org</description>
            </image>
            <generator>TYPO3 - get.content.right</generator>
            <docs>http://blogs.law.harvard.edu/tech/rss</docs>
            <lastBuildDate>Fri, 18 Jan 2013 16:16:40 +0100</lastBuildDate>
            <item>
                  <title>Explanation of the T3A 2013 budget</title>
                  <link>http://typo3.org/news/article/explanation-of-the-t3a-2013-budget/</link>
                  <description>In this article we are going to explain how the budgets have been
assigned for 2013 and the reasons that made us decide like this.</description>
                  <content:encoded><![CDATA[<h2>Introduction</h2>
<h3>Facts and Figures</h3>
It is the first year that the TYPO3 Association...</p>]]>
                  </content:encoded>
                  <pubDate>Wed, 23 May 2012 01:30:00 +0200</pubDate>
            </item>
      </channel>
</rss>
```

The part in which we are really interested has been highlighted: we want to import the various entries of the RSS feed, which correspond to the `<item>` tag.

## The external import setup

First of all, please note that a new column was added to the tx_news_domain_model_news table. It will be used to store the external id found in the RSS feed.

Now here's the setup for the "ctrl" section:

```
$TCA['tx_news_domain_model_news']['ctrl']['external'] = array(
      0 => array(
            'connector' => 'feed',
            'parameters' => array(
                  'uri' => 'http://typo3.org/xml-feeds/rss.xml'
            ),
            'data' => 'xml',
            'nodetype' => 'item',
            'reference_uid' => 'tx_externalimporttut_externalid',
            'enforcePid' => TRUE,
            'disabledOperations' => 'delete',
            'description' => 'Import of typo3.org news'
      ),
);
```

Note that we don't use the same connector service as before. Indeed, we now need the "feed" sub-type, which is provided by extension "svconnector_feed". This connector is specialized in getting XML data from some source (remote or local), which is defined with the `uri` property inside the `parameters` array.

Next, we declare that the data will be provided in XML format and that the reference node type in "item". With this instruction, External Import will take all nodes of type "item" and import each of them. The `enforcePid` property is set to `TRUE` so that the import takes place only in the predefined page and that existing news items entered somewhere else are not deleted. This is a useful precaution to take.

Also note that the delete operation is deleted. This makes sense in this case, as an RSS feed normally contains only the latest news items. Thus if you don't want each import to delete the data from the previous import, the delete operation should be disabled.

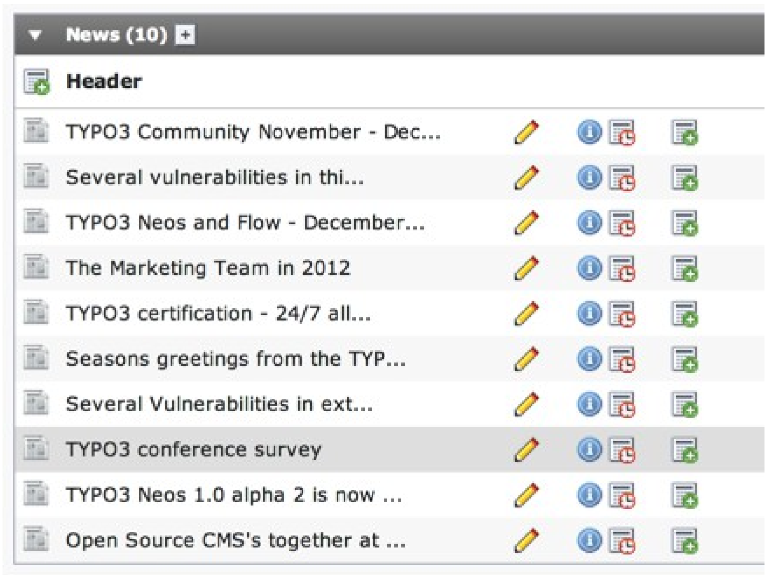
Let's now look at the setup for the columns:

```
$TCA['tx_news_domain_model_news']['columns']['title']['external'] = array(
      0 => array(
            'field' => 'title'
      )
);
$TCA['tx_news_domain_model_news']['columns']['tx_externalimporttut_externalid']['external'] = array(
      0 => array(
            'field' => 'link'
      )
);
$TCA['tx_news_domain_model_news']['columns']['datetime']['external'] = array(
      0 => array(
            'field' => 'pubDate',
            'userFunc' => array(
                  'class' =>
'EXT:external_import/samples/class.tx_externalimport_transformations.php:&tx_externalimport_transformati
ons',
                  'method' => 'parseDate'
            )
      )
);
$TCA['tx_news_domain_model_news']['columns']['short']['external'] = array(
      0 => array(
            'field' => 'description',
            'trim' => TRUE
      )
);
$TCA['tx_news_domain_model_news']['columns']['bodytext']['external'] = array(
      0 => array(
            'field' => 'encoded',
            'rteEnabled' => TRUE
      )
);
$TCA['tx_news_domain_model_news']['columns']['ext_url']['external'] = array(
      0 => array(
            'field' => 'link'
      )
);
$TCA['tx_news_domain_model_news']['columns']['type']['external'] = array(
      0 => array(
            'value' => 0
      )
);
$TCA['tx_news_domain_model_news']['columns']['hidden']['external'] = array(
      0 => array(
            'value' => 0
      )
);
```

For most of the fields, the setup is just as simple as if we were importing database records, thanks to the connector services, which have abstracted the tediousness of getting data in different formats. However XML format allows for more complicated retrieval of data via the use of XPath or attributes.

The only particular configuration above is for the "bodytext" field, which uses the "rteEnabled" property to indicate that the content from this field is rich text and RTE transformations should be applied upon saving. This helps ensure that such content can be edited correctly in a RTE-enabled field in the TYPO3 backend, although the varying quality of available HTML makes it impossible to guarantee a 100% smooth process.

After running the import, check out the page/folder where the imported news items are stored. It should look something like this:


News (10)

Header

TYPO3 Community November - Dec...

Several vulnerabilities in thi...

TYPO3 Neos and Flow - December...

The Marketing Team in 2012

TYPO3 certification - 24/7 all...

Seasons greetings from the TYP...

Several Vulnerabilities in ext...

TYPO3 conference survey

TYPO3 Neos 1.0 alpha 2 is now ...

Open Source CMS's together at ...

# To-Do list

Nothing planned yet, but feel free to submit your ideas and wishes via Forge.