Samuel Bramm, Benjamin Schmelcher

Daimler AG

iCON-MAgicScripts

Documentation and Guideline

Contents

[Introduction 5](#_Toc2264176)

[Project overview 5](#_Toc2264177)

[.*idea* 5](#_Toc2264178)

[*Config* 5](#_Toc2264179)

[*Log* 5](#_Toc2264180)

[*Output* 5](#_Toc2264181)

[*Python-2.7.13* 5](#_Toc2264182)

[*Script* 5](#_Toc2264183)

[*Icon\_scripty.py* 6](#_Toc2264184)

[*Start\_iCON\_scripts.bat* 6](#_Toc2264185)

[How to use the iCON-MagicScript-Collection 6](#_Toc2264186)

[Usage of Patterns 7](#_Toc2264187)

[Db\_Connection 7](#_Toc2264188)

[Usage in the Code 7](#_Toc2264189)

[Integrated Scripts 8](#_Toc2264190)

[Analyze\_I23 [Sebastian] 8](#_Toc2264191)

[Input: 8](#_Toc2264192)

[Output: 8](#_Toc2264193)

[Process 8](#_Toc2264194)

[CBFC\_Duplicate\_Removal [Tejmur] 8](#_Toc2264195)

[Input 8](#_Toc2264196)

[Output 8](#_Toc2264197)

[Process 8](#_Toc2264198)

[Cofico\_feedback\_splitter [Gudy] 9](#_Toc2264199)

[Input 9](#_Toc2264200)

[Output 9](#_Toc2264201)

[Process 9](#_Toc2264202)

[Compare\_user\_rights [Gudy] 9](#_Toc2264203)

[Input 9](#_Toc2264204)

[Output 9](#_Toc2264205)

[Process 9](#_Toc2264206)

[Contingent\_mapper [Sebastian] 10](#_Toc2264207)

[Input 10](#_Toc2264208)

[Output 10](#_Toc2264209)

[Process 10](#_Toc2264210)

[Contingent\_Solution [Sebastian] 10](#_Toc2264211)

[Important 10](#_Toc2264212)

[Input 10](#_Toc2264213)

[Output: 13](#_Toc2264214)

[Process: 14](#_Toc2264215)

[Create\_credit\_notes [Steffen] 14](#_Toc2264216)

[Input 14](#_Toc2264217)

[Output 14](#_Toc2264218)

[Process 14](#_Toc2264219)

[CSV\_Changer 15](#_Toc2264220)

[Doos\_creator [Sebastian] 15](#_Toc2264221)

[Input 15](#_Toc2264222)

[Output 15](#_Toc2264223)

[Process 16](#_Toc2264224)

[Excel\_to\_Utf8\_CSV 16](#_Toc2264225)

[Input 16](#_Toc2264226)

[Output 16](#_Toc2264227)

[Process 16](#_Toc2264228)

[Gems\_compare\_user\_rights [Gudy] 16](#_Toc2264229)

[Gems\_iQuote\_Generator [Daniel] 16](#_Toc2264230)

[Input 16](#_Toc2264231)

[Output 16](#_Toc2264232)

[Process 16](#_Toc2264233)

[Gems\_remove\_tenants [Gudy] 16](#_Toc2264234)

[Input 17](#_Toc2264235)

[Ouptut 17](#_Toc2264236)

[Process 17](#_Toc2264237)

[Gems\_ user\_rights [Gudy] 17](#_Toc2264238)

[Generate\_rule\_config [Tejmur] 17](#_Toc2264239)

[Important 17](#_Toc2264240)

[Input 17](#_Toc2264241)

[Output 17](#_Toc2264242)

[Process 17](#_Toc2264243)

[Configuration File 18](#_Toc2264244)

[iQuote\_gems\_mapper [Daniel] 18](#_Toc2264245)

[Input 18](#_Toc2264246)

[Output 18](#_Toc2264247)

[Process 18](#_Toc2264248)

[iQuote\_Gems\_mapper\_scoped2 [Daniel] 19](#_Toc2264249)

[Migrated\_data\_test [Tejmur] 19](#_Toc2264250)

[Important 19](#_Toc2264251)

[Input 19](#_Toc2264252)

[Output 19](#_Toc2264253)

[Process 19](#_Toc2264254)

[Configuration File 20](#_Toc2264255)

[Parse\_I150\_logs [Gudy] 21](#_Toc2264256)

[Hint 21](#_Toc2264257)

[Input 21](#_Toc2264258)

[Output 21](#_Toc2264259)

[Process 21](#_Toc2264260)

[Parse\_I155\_logs [Gudy] 21](#_Toc2264261)

[Hint 21](#_Toc2264262)

[Input 21](#_Toc2264263)

[Output 21](#_Toc2264264)

[Process 21](#_Toc2264265)

[user\_rights [Gudy] 22](#_Toc2264266)

[Hint 22](#_Toc2264267)

[Input 22](#_Toc2264268)

[Output 22](#_Toc2264269)

[Process 22](#_Toc2264270)

[Configuration File 22](#_Toc2264271)

[Compare invoice runs [Tejmur] 22](#_Toc2264272)

[Important 22](#_Toc2264273)

[Input 22](#_Toc2264274)

[Output 22](#_Toc2264275)

[Process 22](#_Toc2264276)

[Check Salesman [Steffen] 23](#_Toc2264277)

[Important 23](#_Toc2264278)

[Input 23](#_Toc2264279)

[Output 23](#_Toc2264280)

[Process 23](#_Toc2264281)

[Configuration File 24](#_Toc2264282)

[Diff salesman [Steffen] 24](#_Toc2264283)

[Important 24](#_Toc2264284)

[Input 24](#_Toc2264285)

[Output 24](#_Toc2264286)

[Process 24](#_Toc2264287)

[Configuration File 24](#_Toc2264288)

[Gems\_Int\_To\_Prod\_Entitlements 24](#_Toc2264289)

[Input 24](#_Toc2264290)

[Output 25](#_Toc2264291)

[Vega\_I156\_Dealer\_Comparison [Rajesh] 25](#_Toc2264292)

[Input 25](#_Toc2264293)

[Output 25](#_Toc2264294)

[Process: 25](#_Toc2264295)

[XML\_Dealers / XML\_Salesman [Tejmur] 25](#_Toc2264296)

[Input 25](#_Toc2264297)

[Output 27](#_Toc2264298)

[Background Information 28](#_Toc2264299)

[Python 28](#_Toc2264300)

[Pycharm 28](#_Toc2264301)

# Introduction

The iCON-MagicScript Collection contains some useful scripts which I developed during my time as an intern in the iCON-Project. The scripts mostly automate repetitive tasks which would take a lot of time to do manually. This document is there to highlight the different capabilities and to give a detailed view of the scripts. Additionally the document explains how to set up a development environment for python, in case some of the scripts are not working anymore or have to be updated.

# Project overview

This chapter provides a basic project overview. Figure 1 displays the overall project structure as it can be seen from the windows explorer. In the subsequent sections a short explanation for each folder and file is provided.

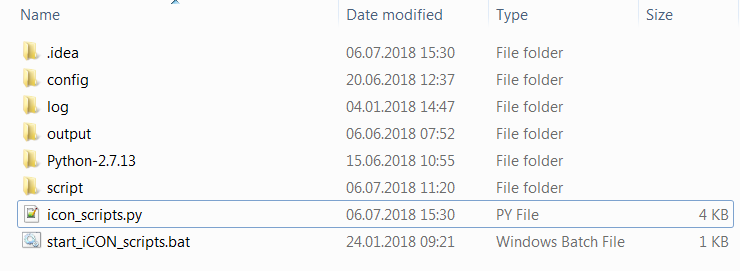


Figure 1project overview

## .idea

This folder contains global settings which are needed for development. There are no changes which have to be made to this folder.

## Config

Some of the scripts have dedicated configuration files. For example to provide access to the database or to easily maintain different settings for the script. For a better overall view the configuration files have the same name as the script files.

## Log

The Log directory contains log files which are created by the scripts. If the cause of the error is known then a log file is created.

## Output

If a scripts creates a output file and does not ask specifically for a destination the generated file will be located in this directory.

## Python-2.7.13

This folder contains the python framework. Please do not change anything in here unless you exactly know what you are doing.

## Script

For better usability and maintainability each script is separated in its own file. These files are located in the script directory.

## Icon\_scripty.py

This is the main script for the iCON-MagicScript-Collection. It contains a wrapper function and enables the execution of the scripts.

## Start\_iCON\_scripts.bat

This batch-file is used to execute the iCON-MagicScript-Collection.

# How to use the iCON-MagicScript-Collection

The usage is pretty straight forward. You just have to double click on the file “Start\_iCON\_scripts.bat” to execute the main application. This will launch the python environment and execute the necessary python files. Once the application has started you will see a screen similar to the one in Figure 2

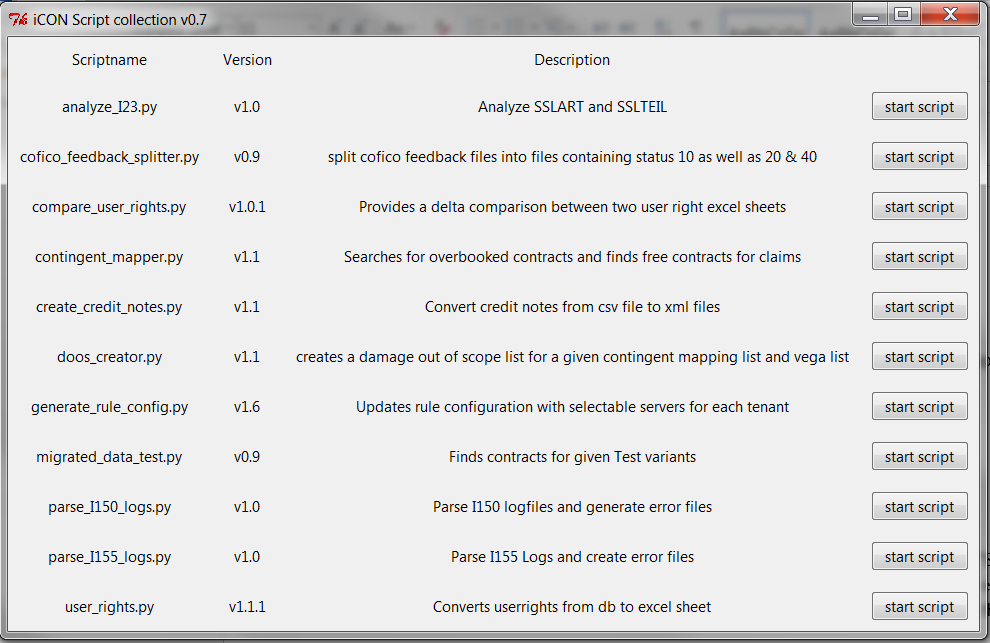


Figure 2main selection menu

In this screen you will see all available scripts and a short description for each script. The scripts can be executed by simply clicking on the “start script”-Button. This will minimize the main menu and execute the associated script. In the following section each script is described in detail.

# Usage of Patterns

## Db\_Connection

The class db\_connection in the module db\_connection provides functions to access our db2 databases easily.

It is necessary to use this class whenever you have to connect to any of our databases because it is strictly not allowed to save credentials in configuration files or anything else. The connection information needs to be put in manually by the user each time a connection is required.

1. Establish a putty connection to the required server
2. Enter username, password and port of the server you want to connect to (The credentials you normally use for Squirrel)
3. Click connect.

## Usage in the Code

When using this class, always make sure to add the line ‘root.protocol(“WM\_DELETE\_WINDOW”, exit\_function)’ to your root window of tkinter in order to always call the exit\_function when the program is closed unintentionally.

In the exit\_function, make sure to call cb\_connection.close\_connection().

The other methods are pretty easy to use, show\_db\_data\_window provides a UI for the credential input and execute\_query is used for sending querys to the database (returns a list containing all result\_tuples).

# Integrated Scripts

## Analyze\_I23 [Sebastian]

### Input:

* SSLART file
* SSLTEIL file
* Damage code and damage type database extract (pre and after import of the files)

### Output:

* Excel file containing the Analysis

### Process

The script produces an Excel file with an analysis of the SSLART und SSLTEIL files. Inside the Excel you will find how much damage codes and types have been added or deleted from the database and you will find an overview over the associated instanceversions. Figure 3 shows the main application interface.

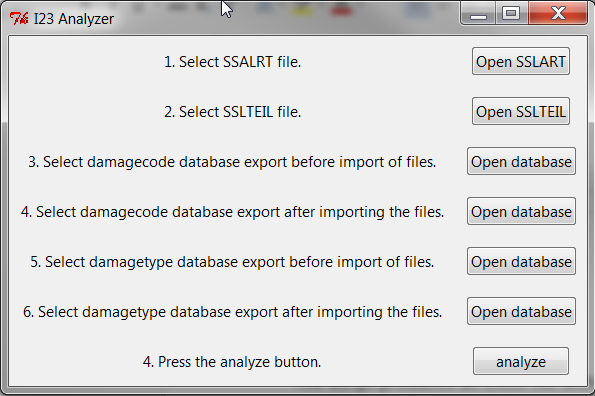


Figure 3I23 Analyzer

## CBFC\_Duplicate\_Removal [Tejmur]

### Input

Directory containing one ore multiple CBFC.txt files with duplicate entries.

### Output

CBFC files (\*.txt) with one entry per file.

### Process

The Script is designed to remove duplicated entries in several CBFC.txt files.

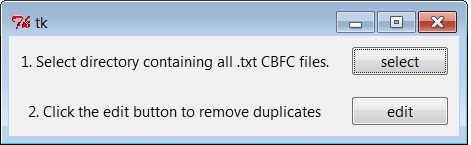


Figure 4 CBFC Duplicate Removal

## Cofico\_feedback\_splitter [Gudy]

### Input

* Folder containing CoFiCo Feedback Files

### Output

* Two text files containing different status codes

### Process

The Script (see Figure 4) is designed to split multiple CoFiCo files into two separate files. One of the files contains all lines from the input files which start with “10” the other file contains all lines which start either with “20” or “40”.

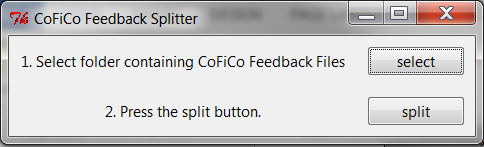


Figure 5CoFiCo feedback splitter

## Compare\_user\_rights [Gudy]

### Input

* Two Excel files containing user rights (produced by the script user\_rights)

### Output

* Excel file containing changes

### Process

The Idea behind this script is to provide a compact overview which users have been added deleted or modified between two distinct dates. Therefore the script (see Figure 5) parses two user\_right Excel files and compares them.

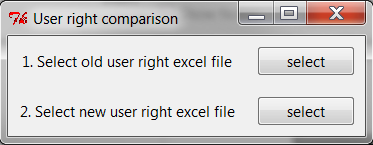


Figure 6user right comparison

## Contingent\_mapper [Sebastian]

### Input

* Excel file containing overbooked contract and claims the file must have the following columns in exact the described order (Column A to O)
  + VIN
  + NUMBER
  + COST\_EXTERNALID
  + DAMAGECODE
  + PLANNEDCONTRACTSTART
  + PLANNEDCONTRACTEND
  + REPAIRDATE
  + FINANCIALDOCUMENTISSUER\_NUMBER
  + CODE
  + ID
  + CONTINGENTCURRENTUSED
  + CONTINGENTINITIAL
  + CODECLAIMINGSYSTEM
  + FK\_PRODUCT
  + Costs

### Output

* Excel file which has to be sent to VEGA
* Error file containing the encountered problems

### Process

The script, which is displayed in Figure 6, searches for overbooked contracts inside the provided Excel file. If such a contract is encountered it determines the claim or claims, which cause the contract to be overbooked. For each claim it the tries to find another contract which has still space left for this claim type. These changes are written to the Excel file, which has to be sent to VEGA to be processes to a damage out of scope list. If there are overbooked contracts, which cannot be resolved automatically the script will write these contracts to the error file.

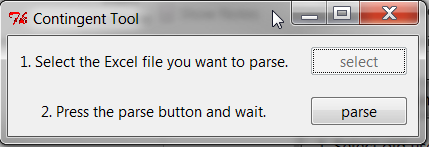


Figure 7contingent mapper

## Contingent\_Solution [Sebastian]

### Important

This script can take several minutes, up to a long time based on the data provided.

### Input

3 Excel files containing DB\_Extracts of the following statements (Adjust the Statements to fit your needs (market, Masterdatareleaseversion, …)

#### Statement 1 (File Sv\_DevX\_YYYYmmdd.xlsx):

Select a.number, e.VIN, d.CONTINGENTCURRENTUSED, d.CONTINGENTINITIAL, b.FK\_PRODUCT, d.CODE, d.ID FROM icon.CONTRACT\_VEHICLECONTRACT a

INNER JOIN icon.CONTRACT\_VEHICLECONTRACTSTABLESTATE b ON a.ACTIVESTABLESTATE\_OBJECTID =b.objectid

INNER JOIN icon.CONTRACT\_VEHICLECONTRACTSTABLESTATE\_ATTR\_COVERAGE c ON c.VEHICLECONTRACTSTABLESTATE\_OBJECTID = b.objectid

INNER JOIN icon.CONTRACT\_COVERAGE d ON d.id = c.COVERAGE\_ID

INNER JOIN icon.VEHICLE\_AUTOMOTIVEOBJECT e ON b.AUTOMOTIVEOBJECT\_OBJECTID = e.objectid

//INNER JOIN icon.PRODUCT\_PRODUCTTOCLAIMINGSYSTEM pps on b.FK\_PRODUCT = pps.FK\_PRODUCT and a.MASTERDATARELEASEVERSION = pps.RELEASEVERSION

WHERE d.code LIKE 'P001'

AND a.number NOT LIKE 'GS%'

AND d.CONTINGENTINITIAL > 1

AND a.TENANTID = '77530BR'

AND b.TENANTID = a.TENANTID

AND d.TENANTID = a.TENANTID

AND e.TENANTID = a.TENANTID

#### Statement 2 (Costs\_P001\_DevX\_YYYYmmdd.xlsx):

//P001

Select vc.number, ao.vin, LEFT(cost.EXTERNALID,8) AS COST\_ExternalID, cd.DAMAGECODE, cd.AMOUNTNET,

vcss.PLANNEDCONTRACTSTART, vcss.PLANNEDCONTRACTEND, cost.REPAIRDATE, cost.FINANCIALDOCUMENTISSUER\_NUMBER, vcss.FK\_PRODUCT, cc.CODE, cc.ID, cc.CONTINGENTCURRENTUSED, cc.CONTINGENTINITIAL

FROM icon.CONTRACT\_VEHICLECONTRACT AS vc

inner join ICON.CONTRACT\_VEHICLECONTRACTSTABLESTATE as vcss on vcss.OBJECTID = vc.ACTIVESTABLESTATE\_OBJECTID

inner JOIN icon.vehicle\_automotiveobject ao ON ao.objectid = vcss.AUTOMOTIVEOBJECT\_OBJECTID

inner join ICON.CONTRACT\_VEHICLECONTRACTSTABLESTATE\_ATTR\_COVERAGE as vcssac on vcssac.VEHICLECONTRACTSTABLESTATE\_OBJECTID = vcss.OBJECTID

inner join ICON.CONTRACT\_COVERAGE as cc on cc.ID = vcssac.COVERAGE\_ID and cc.TENANTID = vcssac.TENANTID

inner join ICON.CONTRACT\_VEHICLECONTRACTVOLATILESTATE as vcvs on vcvs.OBJECTID = vc.ACTIVEVOLATILESTATE\_OBJECTID and vcvs.CONTRACTSTATE like 'contract%'

inner join ICON.COST\_COSTDAMAGE as cd on cd.FK\_COVERAGE = cc.ID and cd.DAMAGECODE in ('00011','00012') and cd.AMOUNTNET > 0

INNER JOIN icon.COST\_COSTDAMAGECOLLECTION\_ATTR\_DAMAGE dcad ON dcad.FK\_DAMAGE\_OBJECTID = cd.objectid

INNER JOIN icon.COST\_COSTDAMAGECOLLECTION dc ON dc.objectid = dcad.COSTDAMAGECOLLECTION\_OBJECTID

INNER JOIN icon.cost\_cost cost ON cost.ACTIVEDAMAGECOLLECTION\_OBJECTID = dc.OBJECTID

INNER join ICON.PARTNER\_DEALERASSIGNMENT pd on vcss.DEALERASSIGNMENT\_OBJECTID = pd.OBJECTID

INNER join icon.partner\_partner p ON cost.FINANCIALDOCUMENTISSUER\_NUMBER = p.number

where vc.TENANTID = '77530BR'

AND cc.CODE IN ('P001')

AND vc.number NOT LIKE 'GS%'

AND cc.CONTINGENTINITIAL > 1

AND vcss.TENANTID = vc.TENANTID

AND cc.TENANTID = vc.TENANTID

AND cost.TENANTID = vc.TENANTID

AND cd.TENANTID = vc.TENANTID

AND p.TENANTID = vc.TENANTID

AND pd.TENANTID = vc.TENANTID

AND dc.TENANTID = vc.TENANTID

AND ao.TENANTID = vc.TENANTID

//AND vc.number IN ('00Q10101/000169')

//AND ao.VIN IN ('LE42049371L314254')

#### Statement 3 (Costs\_D000\_DevX\_YYYYmmdd.xlsx):

//D000

Select vc.number, ao.vin, LEFT(cost.EXTERNALID,8) AS COST\_ExternalID, cd.DAMAGECODE, cd.AMOUNTNET,

vcss.PLANNEDCONTRACTSTART, vcss.PLANNEDCONTRACTEND, cost.REPAIRDATE, cost.FINANCIALDOCUMENTISSUER\_NUMBER, vcss.FK\_PRODUCT, cc.CODE, cc.ID, cc.CONTINGENTCURRENTUSED, cc.CONTINGENTINITIAL

FROM icon.CONTRACT\_VEHICLECONTRACT AS vc

inner join ICON.CONTRACT\_VEHICLECONTRACTSTABLESTATE as vcss on vcss.OBJECTID = vc.ACTIVESTABLESTATE\_OBJECTID

inner JOIN icon.vehicle\_automotiveobject ao ON ao.objectid = vcss.AUTOMOTIVEOBJECT\_OBJECTID

inner join ICON.CONTRACT\_VEHICLECONTRACTSTABLESTATE\_ATTR\_COVERAGE as vcssac on vcssac.VEHICLECONTRACTSTABLESTATE\_OBJECTID = vcss.OBJECTID

inner join ICON.CONTRACT\_COVERAGE as cc on cc.ID = vcssac.COVERAGE\_ID and cc.TENANTID = vcssac.TENANTID

inner join ICON.CONTRACT\_VEHICLECONTRACTVOLATILESTATE as vcvs on vcvs.OBJECTID = vc.ACTIVEVOLATILESTATE\_OBJECTID and vcvs.CONTRACTSTATE like 'contract%'

inner join ICON.COST\_COSTDAMAGE as cd on cd.FK\_COVERAGE = cc.ID and cd.DAMAGECODE in ('00011','00012') and cd.AMOUNTNET > 0

INNER JOIN icon.COST\_COSTDAMAGECOLLECTION\_ATTR\_DAMAGE dcad ON dcad.FK\_DAMAGE\_OBJECTID = cd.objectid

INNER JOIN icon.COST\_COSTDAMAGECOLLECTION dc ON dc.objectid = dcad.COSTDAMAGECOLLECTION\_OBJECTID

INNER JOIN icon.cost\_cost cost ON cost.ACTIVEDAMAGECOLLECTION\_OBJECTID = dc.OBJECTID

INNER join ICON.PARTNER\_DEALERASSIGNMENT pd on vcss.DEALERASSIGNMENT\_OBJECTID = pd.OBJECTID

INNER join icon.partner\_partner p ON cost.FINANCIALDOCUMENTISSUER\_NUMBER = p.number

where vc.TENANTID = '77530BR'

AND cc.CODE IN ('D000')

AND vc.number NOT LIKE 'GS%'

//AND cc.CONTINGENTINITIAL > 1

AND vcss.TENANTID = vc.TENANTID

AND cc.TENANTID = vc.TENANTID

AND cost.TENANTID = vc.TENANTID

AND cd.TENANTID = vc.TENANTID

AND p.TENANTID = vc.TENANTID

AND pd.TENANTID = vc.TENANTID

AND dc.TENANTID = vc.TENANTID

AND ao.TENANTID = vc.TENANTID

//AND vc.number IN ('00Q10101/000169')

//AND ao.VIN IN ('LE42049371L314254')

### Output:

Excel file containing the endresult to use in script contingent\_mapper.

Endresult is Step 20 in the “Contingent\_Solution\_Instruction.txt” file in this directory.

### Process:

This script reads the 3 database excel files (containing the extracts of the queries given above) and reproduces the steps which were done in an access tool before this script existed.

The script runs the steps of the access tool and therefore produces the endresult file which then can be used for the further process.

## Create\_credit\_notes [Steffen]

### Input

* The script expects a utf-8 formated csv file with the following columns in the mentioned order
  + Contract number
  + Financial document receiver number
  + Credit amount
  + Invoice date
  + Document type (Invoice or Credit)

### Output

There will be a separated xml file for credit notes and for invoices.

### Process

The script, as shown in Figure 8, creates two separate xml files. One for all found credit note and one for all found invoices. These xml files can later be imported in the iCON system via the administrative upload function. To enhance the usability the script allows the user to adjust several settings (for example the tenanted, the currency, etc.). These settings can be accessed, if the “Advanced Mode” checkbox is selected upon pressing the “create”-button. The additional settings can be seen in Figure 7. If there are no values entered in the provided fields the script will use the displayed values.

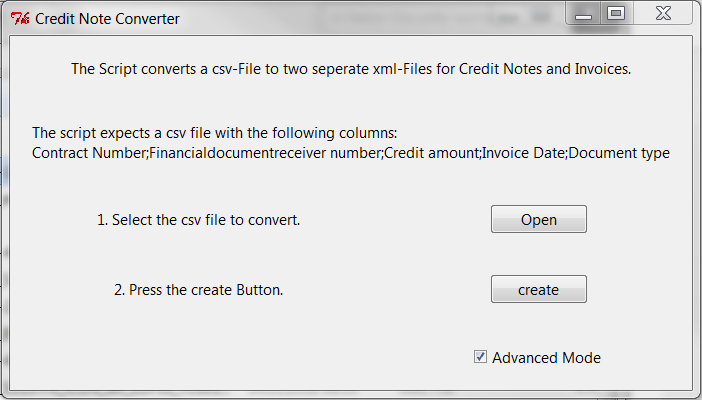
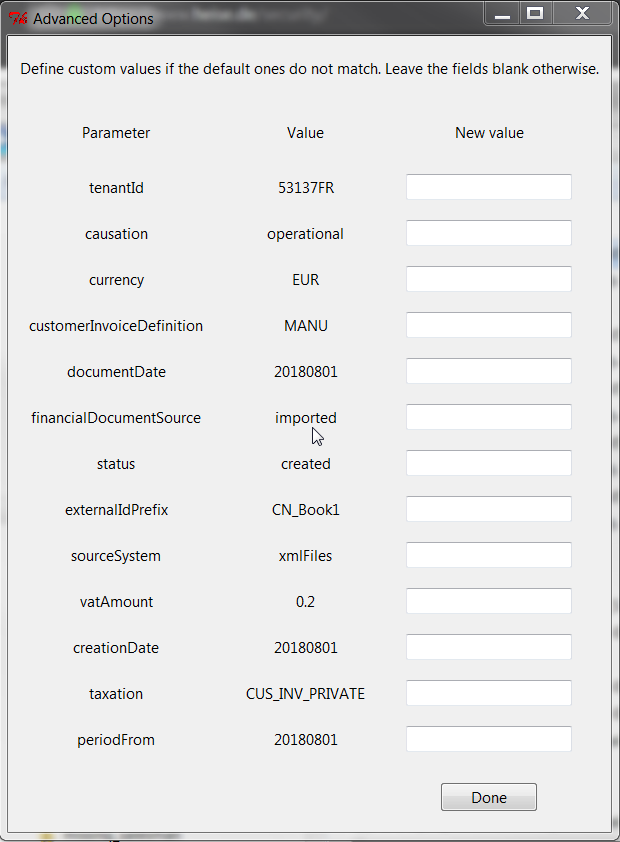


Figure 8Advanced Settings

Figure 9Credit Notes script

## CSV\_Changer

Input

Any CSV\_file

Output

Split csv files into smaller sizes

OR

CSV file witch changed delimiter

Process

Reads a csv file, asks the user if they want to change the delimiter or split the file into smaller files (to open them in excel), then provides the respective output file(s).

## Doos\_creator [Sebastian]

### Input

* Output from the Contingent Mapping script
* An empty VEGA list, which has to be provided by VEGA

### Output

* VEGA list filled with the contracts which have to be changed

### Process

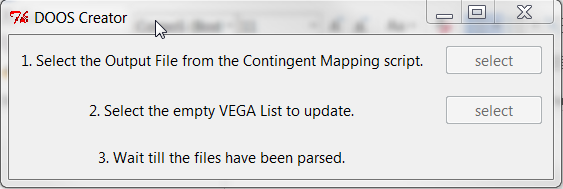
During the execution the script (see Figure 9) parses the contingent mapping list to find claims which have to be attached to a other contract. These claims are the crosschecked with the empty VEGA list. As a las step the VEGA list is populated with the claims. This list can then be sent to VEGA once again.

Figure 10Damage out of scope

## Excel\_to\_Utf8\_CSV

### Input

Any Xlsx-File

### Output

UTF-8 converted CSV file

### Process

Reads the xlsx file and converts it to csv format, formatted in utf-8.

## Gems\_compare\_user\_rights [Gudy]

Reads two Gems\_user\_rights files and provides a delta file between them. (See compare\_user\_rights documentation for AAT).

## Gems\_iQuote\_Generator [Daniel]

### Input

Nothing

### Output

Upload File for GEMS containing the information specified in the UI.

### Process

Start the script, follow the instructions in the interface and afterwards upload the generated file to iquote.

## Gems\_remove\_tenants [Gudy]

### Input

Missing\_organisations\_protocol

Gems Upload Files

### Ouptut

Protocol containing the missing organisations

OR

Gems Upload Files without the entries which don’t have a tenant specified in the protocol file

### Process

You can choose between creating a protocol for the missing users or removing the entries which have a missing tenant (in the missing\_organisations\_protocol).

The script then either creates the protocol or creates the Gems\_upload\_files without the missing-tenants-entries.

## Gems\_ user\_rights [Gudy]

See User\_Rights.

Functionality is exactly the same as in User\_Rights.

However, the script User\_Rights was user for AAT, Gems\_user\_Rights is used for GEMS Exports.

## Generate\_rule\_config [Tejmur]

### Important

* Dev server access via putty is needed to run this script properly

### Input

* The scripts expects a Excel file containing the possibly previous Rule configurations and a “overview sheet”
* Configuration file

### Output

* The script append the given Excel file with all chosen options and also updates the overview sheet
* List with parameters which are not present in the first Tab of the excel sheet

### Process

Once the script in Figure 10 is executed the user has to specify a dev environment from which the current available tenant list should be updated. Please specify a dev environment with as many tenants available as possible. In the next step the user has to provide his or her login credentials to crosscheck the deployed iCON version on the server. After this step the script shows a list with possible tenants and a selection. For each tenant which has to be updated the user has to specify a dev environment via the dropdown and check the checkbox. Before the scripts tries to fetch the rule configurations the user has to provide the rule configuration excel sheet by clicking on the “select button”. Once the “download”-button is pressed the script tries to fetch all selected rule configurations and therefore the user has to establish a putty session. The script will show a dialog which connection is currently expected.

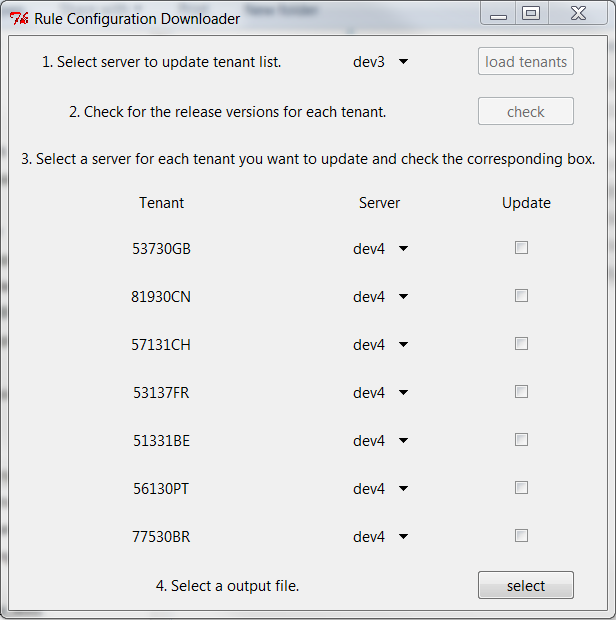


Figure 11Rule Configuration

### Configuration File

The script has a dedicated configuration file in the config-folder. There are all the necessary configurations stored. These include:

* Excluded parameters which are excluded when checking if a found parameter is present in the first excel sheet
* The database credentials for the dev servers and the last known deployed software version
* The known tenants

## iQuote\_gems\_mapper [Daniel]

### Input

iQuote CSV File

### Output

Gems Upload Fiels

### Process

Reads the iQuote CSV file and creates GEMS upload Files based on the iquote\_gems\_mapper.yml config file.

## iQuote\_Gems\_mapper\_scoped2 [Daniel]

Works like iQuote\_Gems\_mapper but has scoped roles in the config file

## Migrated\_data\_test [Tejmur]

### Important

* Dev server access via putty is needed to run this script properly

### Input

* Excel file containing a list of test variants in a sheet named “Test Variants”. This sheet must have the following columns (exact order)
  + #
  + V
  + Product
  + Division
  + Price Model
  + I
  + Invoicing
  + R
  + Contract Relationship
  + Contract Number
  + Applicable
  + Status
  + Remark

### Output

* Excel file with found contracts for each test variant

### Process

The script, shown in Figure 11, parses the given excel file for the test variants. It then tries to find the specified amount of contracts for each test variant and outputs the found contract numbers in the corresponding column. There are several optional options which can be specified, like a random pull for the contract numbers.

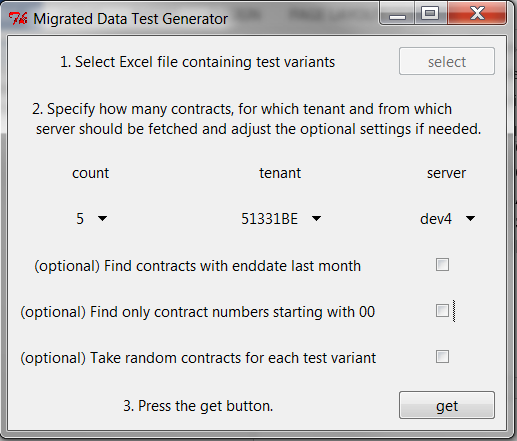


Figure 12Migrated Data Tests

### Configuration File

The configuration file is located in the config folder. It contains the following configurations:

* Database access credentials
* Tenant list
* Mappings between values for price models, invoicing types, etc. and the corresponding database values

## Parse\_I150\_logs [Gudy]

### Hint

This script may take a long time to execute depending of the provided filesize.

### Input

* I150 logfile

### Output

* Csv-file containing all found errors

### Process

The script in Figure 12 parses the provided I150 file for errors identified by the pattern “ICON-CLAIM”. These errors and their corresponding message are then written to the csv file with the following columns:

* claim number
* error code
* error message

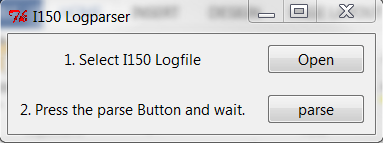


Figure 13I150 parser

## Parse\_I155\_logs [Gudy]

### Hint

This script may take a very long time to execute depending on the number of provided files and their respective size.

### Input

* Path to a folder containing only I155 logfiles

### Output

* Csv file containing all found errors
* Csv file containing all found registrations

### Process

The script in parses all files in the provided folder and looks for the patterns “REJECT”, “ERROR”, “Registration mark” and “FIN - OLD” to find errors and registrations inside these files. The found registrations and errors, as well as their corresponding codes and messages are the written to the two different output files.

## user\_rights [Gudy]

### Hint

* This script may take a long time to execute dependence on the provided size of the file
* This script only works for AAT role assignments

### Input

* Dd\_user report generated by AAT

### Output

* Excel list containing the users and their assigned roles in iCON
* Text file containing all encountered errors

### Process

The script extracts all users and their roles from the csv file, except those starting with one of the strings from the exclude list (see configuration file). The script then tries to determine the country bases on the associated outlet id in the configuration file and maps the AAT roles to a nicer display style. All this information is then displayed in an excel spreadsheet. If the script was unable to determine the country for a specific user there will be an error file with the unknown values.

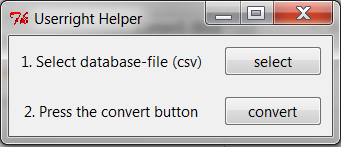


Figure 14 Convert User Rights

### Configuration File

The configuration file is located in the config folder. It contains the following configurations:

* Exclude list: User IDs starting with one of these strings are ignored
* Country mapping: Mapping between Outlet codes and countries
* Rights mapping: mapping between the roles as they are in the database and their appearance in the excel file

## Compare invoice runs [Tejmur]

### Important

This script currently only works for the Brazil market. If the columns are in the exact same order as in Brazil the script will work for other markets.

### Input

* Simulated Periodic or PAYD invoice run
* Invoice run (periodic or PAYD) from CMAN

### Output

* Excel spreadsheet

### Process

Starting the script as shown in Figure 14 enables the user to select which kind of invoice the script should compare. The script compares the two provided invoice files. It checks whether all VINs form CMAN run are present in the iCON run and the other way around. If a VIN is present in both runs the script checks if the following fields are the same in both invoice runs:

* Net Amount
* Document date [optional]
* Odometer [only PAYD]

The comparison, as well as the overview which VINs are present in which file is the written to a new excel spreadsheet.

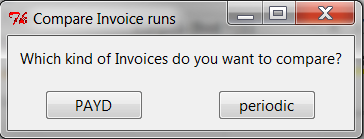


Figure 15Invoice Comparison

## Check Salesman [Steffen]

### Important

This script requires database access via putty.

### Input

* Csv file with users from AAT

### Output

* Excel file containing errors
* Xml file for loading the salesman into iCON
* Xml file for loading the dealers into iCON

### Process

The script in Figure 15 extracts all found users and their information from the provided csv file. To avoid unnecessary work the script then checks if the found salesman already exist in iCON. If not the script creates the necessary xml files to load the salesman and the dealers into iCON. If there are errors, because a dealer is missing in iCON the script creates a separate Excel file which has filled out and then used with the diff salesman script.

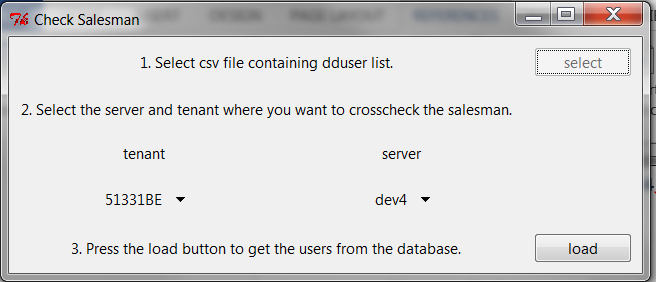


Figure 16Check Salesman script

### Configuration File

The configuration file is located in the config folder. It contains the following configurations:

* Database credentials
* List of known tenants

## Diff salesman [Steffen]

### Important

This script requires database access via putty.

### Input

* Excel spreadsheet form the check salesman script

### Output

* Xml file for loading the salesman into iCON
* Xml file for loading the dealers into iCON

### Process

This script in Figure 16 nearly works in the same way as the check salesman script. It parses the Excel file and checks whether the salesman or the dealers exist in iCON. Then it creates two xml files for salesman and dealers which can be uploaded into iCON.

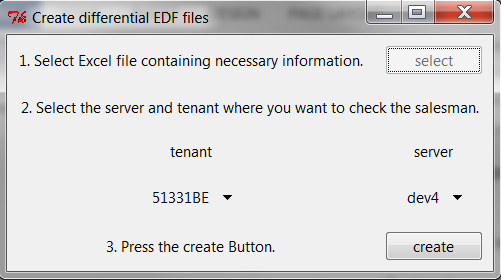


Figure 17DIff Salesman script

### Configuration File

The configuration file is located in the config folder. It is the same as the one for the check salesman script (ckeck\_salesman.yml). It contains the following configurations:

* Database credentials
* List of known tenants

## Gems\_Int\_To\_Prod\_Entitlements

### Input

Excel File (Doku AS22\_RechteRollen\_ICON\_GEMS\_PROD 5\_1 .. xlsx (see attached file in this directory)

### Output

Entitlement mapping between Int and Prod for GEMS

ProductClaimingSystemMapper

Input

Endresult.xlsx (from script contingent\_solution)

ProductToClaimingBereinigt.xlsx (see attached file in this directory)

Output

Changed endresult file with the productToClaimingSystem in it.

## Vega\_I156\_Dealer\_Comparison [Rajesh]

### Input

Combined File of 2 DB Extracts (See below SQL Statements)

Vega Dealers/Workshop File

#### SQL Statements.

Statement for all tenants except china: Select \* From icon.PARTNER\_PARTNER where PARTNERTYPE ='workshop' and STATE='active' and TENANTID in ('51331BE', '53137FR', '56130PT', '77530BR')

Statement for china: Select \* From icon.PARTNER\_PARTNER where PARTNERTYPE ='workshop' and STATE='active' and TENANTID = '81930CN'

After executing them and extracting the respective Excel file, combine them into one.

Output:  
Comparison Files between VEGA and ICON with missing dealers in iCON and missing dealers in VEGA (one combined and one for each market)

### Process:

Compares the iCON file with the VEGA file and creates output file with compared dealers in each fiel.

## XML\_Dealers / XML\_Salesman [Tejmur]

### Input

Excel file containing Salesman and Dealer Data in the following format:

Sheet 1: Salesman – Dealer Assignment:

|  |
| --- |
| UID |
| first name |
| name |
| phone |
| mobile |
| mail |
| homeorg |

Sheet 2: Dealer data

|  |
| --- |
| GSSNOUTLETID |
| COMPANYNAME |
| COMPANYNAME2 |
| VATID |
| GSSNCOMPANYOUTLETID |
| CITY |
| COUNTRY |
| STREET |
| ZIPCODE |
| PHONE |
| FAX |

Sheet 3: DB Dealer

|  |
| --- |
| NUMBER |
| TENANTID |
| OBJECTINSTANCETYPE |
| INSTANCEVERSION |
| CUSTOMERGLOBAL\_OBJECTID |
| LEGALADDRESS\_OBJECTID |
| COMMUNICATIONDATA\_OBJECTID |
| MAILADDRESS\_OBJECTID |
| INVOICEADDRESS\_OBJECTID |
| REVENUERECEIPT\_OBJECTID |
| STATE |
| PARTNERTYPE |
| DISPLAYNAME |
| FK\_BACKUP |
| LOCALIZEDPARTNERDETAILS\_OBJECTID |
| LASTMODIFICATIONDATE |
| EXTERNALID |
| SOURCESYSTEM |
| CREATEDBY |
| MIGRATIONDATE |
| MASTERDATARELEASEVERSION |
| COMPANYNAME |
| MARKETAREA |
| COMMERCIALREGISTRATION |
| COMPANYINTERNAL |
| VATID |
| ENDOFFISCALYEAR |
| COMPANYNAME2 |
| GSSNOUTLETOUTLETID |
| CLAIMINGSYSTEMID |
| FK\_PARENTORGANISATION |
| COMMERCIALCUSTOMERGLOBALS\_OBJECTID |
| COSTISSUER\_OBJECTID |
| FK\_LEGALFORM |
| GSSNOUTLETID |
| GSSNOUTLETCOMPANYID |
| GSSNOUTLETOUTLETCODE |
| MAINCONTACTPERSON\_NUMBER |
| FIRSTNAME |
| ISUSERLASTLOGIN |
| LASTNAME |
| DATEOFBIRTH |
| PLACEOFBIRTH |
| PERSONALFISCALCODE |
| DRIVINGLICENSEIDENTITYCARD |
| DEALERDIRECTORYUID |
| FK\_SALUTATION |
| DEFAULTGSSNOUTLETOUTLETID |
| SALESCHANNEL |
| ORGANISATIONALLEVEL |
| TECHCHANGETIME |
| TECHCHANGEUSER |
| TECHCREATETIME |
| TECHCREATEUSER |

Sheet 4: user data:

|  |
| --- |
| UID |
| first name |
| name |
| phone |
| mobile |
| mail |
| homeorg |

Output:

Upload files to upload in iCON as EDF (containing either Dealer data or Salesman data)

# Background Information

The following sections provide a quick introduction, installation guide and some useful links for python and pycharm. These sections are only necessary if you want to dig deeper and want to change/extend or debug some of the scripts.

## Python

Python is a script language just like Java. This means that a runtime environment is needed to execute a script file. This runtime is provided by the python framework. The framework can be found on the project webpage <https://www.python.org/downloads/> or inside the iCON-MagicScript-Collection. The script collection is designed to work with the python versions 2.7.x. This was done to provide compatibility of the scripts to the Linux servers which run the dev environment. Some of the scripts were initially designed to run on these servers but were never deployed. Depending on your access right you will not be able to install the python framework natively on your laptop. But for this case you can always use the provided python framework.

Python is a very modular language and heavily relies on modules. These modules can be imported into a python script to enhance its capability. For example the ibm\_db2 module provides support for interacting with db2 databases. If a module cannot be loaded the script will fail to run and will terminate. All necessary modules have been included in the provided python folder.

## Pycharm

Pycharm is an IDE (Integrated Development Environment) for python which allows a rapid deployment, testing and debugging of python scripts. The free community edition can be found under <https://www.jetbrains.com/pycharm/download/#section=windows> Just download the IDE and install it into your local folder. Once the IDE is installed the iCON-MagicScipt-Collection can be imported.