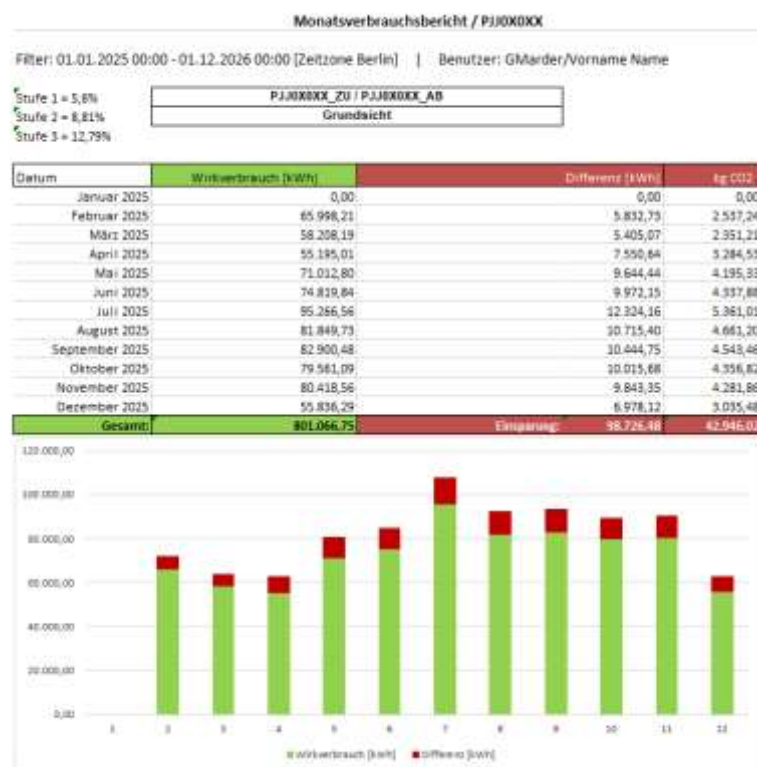


eSaver-Ausw_Erspar v1.02.2

Bedienungsanleitung



Allgemeine Hinweise

Um die einzelnen Stunden-Verbrauchs-Berichte pro Monat in die auszuführende Excel Datei zu kopieren, wurde dieses Tool erstellt.

Es soll helfen, die Übernahme der Daten in die Auszuwertende Datei zu erleichtern und Fehler beim Umsetzen zu vermeiden.

Inhaltsverzeichnis

1. Aufbau der Konfiguration Datei	4
1.1. Besonderheiten der Konfigurations-Datei	4
1.1.1. Voraussetzung damit das Tool arbeiten kann	4
2. Beschreibung der Konfigurations-Datei	5
2.1. Aufbau der Tabelle 1 [Export-VIDA Main]	5
2.2. Bedienung der Excel-Datei	6
2.2.1. Button zum Testen der Verbindung zu Python	6
2.2.2. Button zum Bearbeiten der VIDA-Daten	7
3. Beschreibung der Xlwings Konfigurations-Datei	8
3.1. Aufbau der Tabelle 2 [xlwings.conf]	8
3.2. Sonderfunktion von Xlwings	9
3.2.1. Sonderfunktion "Debug UDFs"	9
3.2.2. Sonderfunktion „Show Console“	9
3.2.3. Sonderfunktion "Use UDF Server"	10

Bedienungsanleitung

1. Aufbau der Konfiguration Datei

Als Konfigurations-Datei wurde eine Excel-Datei
„eSaver_Export_VIDA_Ausw_Erspar_SVB_PJJOX0XX.xlsm“ erstellt.

Kürzel in dem Namen:

- > SVB Stunden Verbrauchs Bericht
- > PJJOX0XXX Projekt Nummer

1.1. Besonderheiten der Konfigurations-Datei

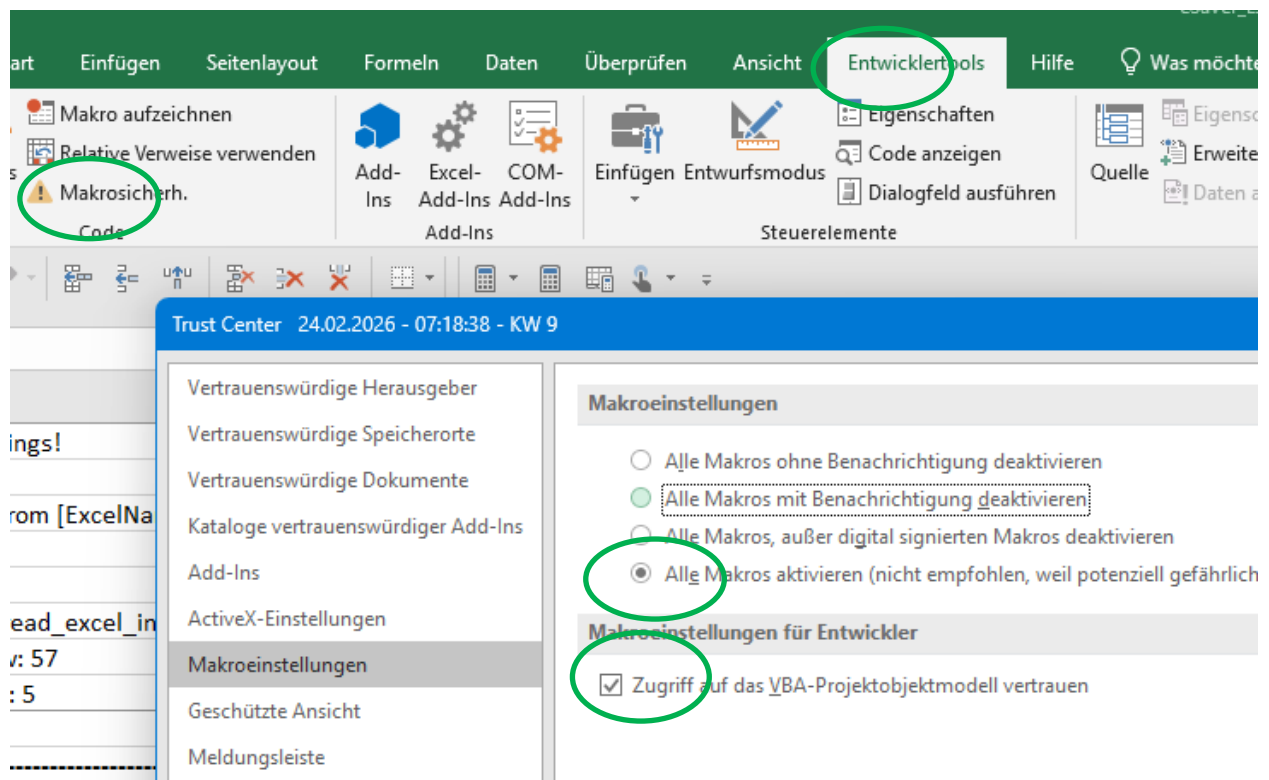
Die Excel-Datei sammelt die benötigten Informationen zu den Dateien und Verzeichnisse.

Die weitere Bedienung des Tools erfolgt ebenfalls über diese Excel-Datei.

Die Bedienung erfolgt über definierte Schaltflächen.

1.1.1. Voraussetzung damit das Tool arbeiten kann

Die Datei wurde für den Zugriff von VBA freigegeben; dies wird benötigt, damit über
Python - packages [xlwings] die Excel-Datei mit dem Tool unter Python zusammenarbeiten kann.



Bedienungsanleitung

2. Beschreibung der Konfigurations-Datei

2.1. Aufbau der Tabelle 1 [Export-VIDA Main]

Tabelle: [Export-VIDA Main]

Bye xlwings!	Initial-Werte		
Called from [ExcelName_py.xw_from_cell_req:]	Test Main		
call_2: read_excel_inits Fertig!			
last_row: 57			
last_col: 5			
eSaver Export VIDA Aus-Erspar INIT's	Name1-Name2		
eSaver:	G63 M / 1000 kVA		
Projektnummer:	PIJ0X0XX		
Datum	18.02.2026		
Jahresübersicht	2025		
Datum_Start	01.01.2025		
Datum_End	31.12.2025		
VIDA-Version	GridVis		
Formatvorlage	*.csv		
Export VIDA	Stundenverbrauch		
Path_Export_Files	d:\20_PROJ_ABWICKL\esaver_Ausw_Erspar\esaver_Ausw_Erspar_Py_v1.02.2.4_py31403\example\		
Path_Import_Files	d:\20_PROJ_ABWICKL\esaver_Ausw_Erspar\esaver_Ausw_Erspar_Py_v1.02.2.4_py31403\example\		
Export_File_01_Jan	PIJ0X0XX Export_VIDA-Data_2025_0101_2025_0201_StdVerBer_GM.csv		
Export_File_02_Feb	PIJ0X0XX Export_VIDA-Data_2025_0201_2025_0301_StdVerBer_GM.csv		
Export_File_03_Mrz	PIJ0X0XX Export_VIDA-Data_2025_0301_2025_0401_StdVerBer_GM.csv		
Export_File_04_Apr	PIJ0X0XX Export_VIDA-Data_2025_0401_2025_0501_StdVerBer_GM.csv		
Export_File_05_Mai	PIJ0X0XX Export_VIDA-Data_2025_0501_2025_0601_StdVerBer_GM.csv		
Export_File_06_Jun	PIJ0X0XX Export_VIDA-Data_2025_0601_2025_0701_StdVerBer_GM.csv		
Export_File_07_Jul	PIJ0X0XX Export_VIDA-Data_2025_0701_2025_0801_StdVerBer_GM.csv		
Export_File_08_Aug	PIJ0X0XX Export_VIDA-Data_2025_0801_2025_0901_StdVerBer_GM.csv		
Export_File_09_Sep	PIJ0X0XX Export_VIDA-Data_2025_0901_2025_1001_StdVerBer_GM.csv		
Export_File_10_Okt	PIJ0X0XX Export_VIDA-Data_2025_1001_2025_1101_StdVerBer_GM.csv		
Export_File_11_Nov	PIJ0X0XX Export_VIDA-Data_2025_1101_2025_1201_StdVerBer_GM.csv		
Export_File_12_Dez	PIJ0X0XX Export_VIDA-Data_2025_1201_2026_0101_StdVerBer_GM.csv		
Import_File_01_Jan	PIJ0X0XX Export_VIDA-Data_2025_0101_2025_0201_StdVerBer_imp.xlsx		
Import_File_02_Feb	PIJ0X0XX Export_VIDA-Data_2025_0201_2025_0301_StdVerBer_imp.xlsx		
Import_File_03_Mrz	PIJ0X0XX Export_VIDA-Data_2025_0301_2025_0401_StdVerBer_imp.xlsx		
Import_File_04_Apr	PIJ0X0XX Export_VIDA-Data_2025_0401_2025_0501_StdVerBer_imp.xlsx		
Import_File_05_Mai	PIJ0X0XX Export_VIDA-Data_2025_0501_2025_0601_StdVerBer_imp.xlsx		
Import_File_06_Jun	PIJ0X0XX Export_VIDA-Data_2025_0601_2025_0701_StdVerBer_imp.xlsx		
Import_File_07_Jul	PIJ0X0XX Export_VIDA-Data_2025_0701_2025_0801_StdVerBer_imp.xlsx		
Import_File_08_Aug	PIJ0X0XX Export_VIDA-Data_2025_0801_2025_0901_StdVerBer_imp.xlsx		
Import_File_09_Sep	PIJ0X0XX Export_VIDA-Data_2025_0901_2025_1001_StdVerBer_imp.xlsx		
Import_File_10_Okt	PIJ0X0XX Export_VIDA-Data_2025_1001_2025_1101_StdVerBer_imp.xlsx		
Import_File_11_Nov	PIJ0X0XX Export_VIDA-Data_2025_1101_2025_1201_StdVerBer_imp.xlsx		
Import_File_12_Dez	PIJ0X0XX Export_VIDA-Data_2025_1201_2026_0101_StdVerBer_imp.xlsx		
PIJ0X0XX_Ersparnis_File	PIJ0X0XX_Name1-Name2 Ersparnis 2025_01-12 - Stundenbasiert GMpk.xlsx		

In dieser Tabelle werden die

Bedienungsanleitung

2.2. Bedienung der Excel-Datei

Zur leichten Bedienung der Konvertierung der Dateien und die Übernahme der Werte in die Jahresauswertungs-Datei wurden Bedien-Buttons eingebunden.

Hinweis:

Die eingebundenen Buttons sind noch sehr technisch gebunden und werden bei nächster Gelegenheit in der Gruppe Diskutiert und besser festgelegt.

2.2.1. Button zum Testen der Verbindung zu Python



❖ Call_Py_xw_main_req()

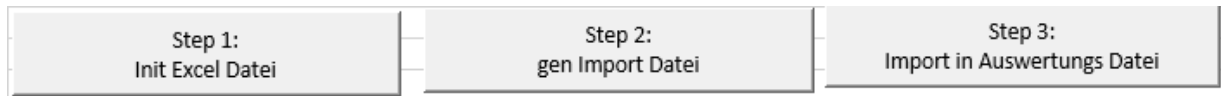
Damit wird die Funktion "xw_main_req" in Python aufgerufen, und eine Nachricht in die Zeile A1 geschrieben

-> Der Text in Zeile A1 wird abwechselnd mit 'Hello xlwings' / 'Bye xlwings' geschrieben, um eine Rückmeldung von Python zu erkennen

-> Der Text in Zeile A3 wird 'Called from [ExcelName_py.xw_main_req:]' eingetragen

Bedienungsanleitung

2.2.2. Button zum Bearbeiten der VIDA-Daten



❖ Schritt 1:
Init Excel Datei

Damit wird die Funktion (Initialisierung der Excel-Import einträge) in Python aufgerufen, mit dieser Funktion werden die Dateinamen aus en Spalten "Export-Files_MM_mmm" gelesen und die Erweiterungen gegen _Imp.xlsx getauscht. Diese Dateinamen werden dann in der Excel unter den "Import_Files_MM_mmm" eingetragen

-> Als Rückmeldung springt der Button wieder aus seiner gedrückten Position zurück.

❖ Schritt 2:
gen Import Datei

Damit wird die Funktion (generieren der Import Dateien) in Python aufgerufen, mit dieser Funktion werden die einzelnen Export-Dateien eingelesen und die gelesenen Werte überprüft. Die Werte werden auf eine genormte Einheit konvertiert und in das Excel-Format überführt. Danach werden die einzelnen Import-Files "Import_Files_MM_mmm" erstellt.

-> Als Rückmeldung springt der Button wieder aus seiner gedrückten Position zurück.

❖ Schritt 3:
Import in Auswertungs Datei

Damit wird die Funktion (Importiere generierte Files in die Auswertungs-Datei) in Python aufgerufen, mit dieser Funktion werden die einzelnen Import-Dateien eingelesen und die gelesenen Werte überprüft. Das Datum in den Import-Dateien bestimmen welche Tabelle in der "Auswertungs-Datei" geöffnet wird. Danach werden die einzelnen Werte in die "Auswertungs-Datei" übertragen.

-> Als Rückmeldung springt der Button wieder aus seiner gedrückten Position zurück.

3.1. Aufbau der Tabelle 2 [xlwings.conf]

3.2. Sonderfunktion von Xlwings

3.2.1. Sonderfunktion “Debug UDFs”

Debug UDFs	TRUE
------------	------

➤ **TRUE**

Die Debug unter VBA wird unterstützt

➤ **FALSE** Standard

Die Debug unter VBA wird nicht unterstützt

3.2.2. Sonderfunktion „Show Console“

Show Console	TRUE
--------------	------

➤ **TRUE**

Die Windows-Konsole mit den Python „print()“-Ausgaben“ wird angezeigt.

Info:

Der Ablauf der Funktionen sind langsamer,
durch die optische Rückmeldung können Fehler leichter diagnostiziert werden.

➤ **FALSE** Standard

Die Windows Konsole mit den Python „print()“-Ausgaben“ wird nicht angezeigt.

Info:

Der Ablauf der Funktionen ist dadurch geringfügig schneller,
bei Fehler bekommt man keine optische Rückmeldungen angezeigt.

Bedienungsanleitung

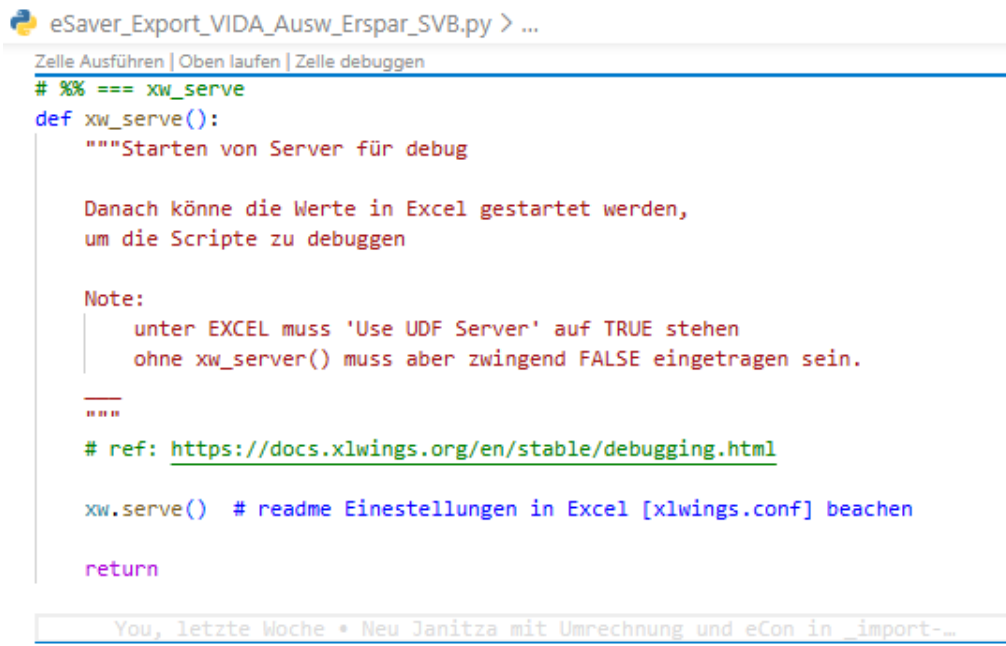
3.2.3. Sonderfunktion "Use UDF Server"

Use UDF Server	FALSE
----------------	-------

➤ **TRUE**

Es wird eine Abfrage aktiviert, die den Start des Debuggers von VSCode erkennt und eine Verbindung zum Debug-Mode von VSCode herstellt.

Nun kann unter VSCode die Master-Datei mit der Taste "F5" gestartet werden.



```
eSaver_Export_VIDA_Ausw_Erspar_SVB.py > ...
Zelle Ausführen | Oben laufen | Zelle debuggen
# %% === xw_serve
def xw_serve():
    """Starten von Server für debug

    Danach könne die Werte in Excel gestartet werden,
    um die Scripte zu debuggen

    Note:
    | unter EXCEL muss 'Use UDF Server' auf TRUE stehen
    | ohne xw_serve() muss aber zwingend FALSE eingetragen sein.
    """
    # ref: https://docs.xlwings.org/en/stable/debugging.html

    xw.serve() # readme Einstellungen in Excel [xlwings.conf] beachten

    return

You, letzte Woche • Neu Janitza mit Umrechnung und eCon in _import-...
```

➤ **FALSE** Standard

Es wird keine Abfrage an ein laufendes Python Programm gestartet.
Ein debuggen unter VSCode mit der Taste F5 startet den Debugger es kann aber keine Verbindung aufgebaut werden.