

# iG/Script Debugger

## Zusammenfassung

Diese Dokumentation beschreibt die Installation, Konfiguration und Funktionsweise des iG/Script Debuggers.

Die Dokumentation darf nur mit Erlaubnis der infoGrips GmbH vervielfältigt werden.

---

# Inhaltsverzeichnis

1. Einleitung .....	4
1.1. Überblick .....	4
1.2. Aufbau dieser Dokumentation .....	4
1.3. Konventionen .....	4
2. Installation Visual Studio Code .....	5
2.1. Erweiterung installieren .....	5
3. Konfiguration iG/Script Debugger .....	6
3.1. Einstellungen .....	6
3.2. Explorer .....	7
3.3. Testskript erstellen .....	7
3.4. Debug-Konfiguration erstellen .....	8
4. Funktionsweise des Debuggers .....	11
4.1. ICS Launch (32-Bit) / ICS Launch (64-Bit) .....	11
4.2. ICS Listen .....	11
4.3. Erläuterung Debugging .....	12
4.3.1. Variablen .....	15
4.3.2. Watch .....	15
4.3.3. Call Stack .....	15
4.3.4. Breakpoints .....	15
4.3.5. Steuerung .....	15

# 1. Einleitung

## 1.1. Überblick

Mit dem iG/Script Debugger lassen sich Anpassungen leichter bewerkstelligen, Fehler effizienter lokalisieren und beheben. Mithilfe des frei verfügbaren Visual Studio Code von Microsoft können sie ihre Skripte leichter implementieren.

## 1.2. Aufbau dieser Dokumentation

Diese Dokumentation ist wie folgt aufgebaut:

- Kapitel 2: Installation Visual Studio Code.
- Kapitel 3: Konfiguration iG/Script Debugger.
- Kapitel 4: Funktionsweise des Debuggers.

## 1.3. Konventionen

In dieser Dokumentation werden folgende Konventionen eingehalten:

kursiv	Namen von Dateien und URL's
<b>fett</b>	neue Begriffe, Namen von Funktionen oder Methoden
<code>courier</code>	Programmtext oder Eingaben im Betriebssystem

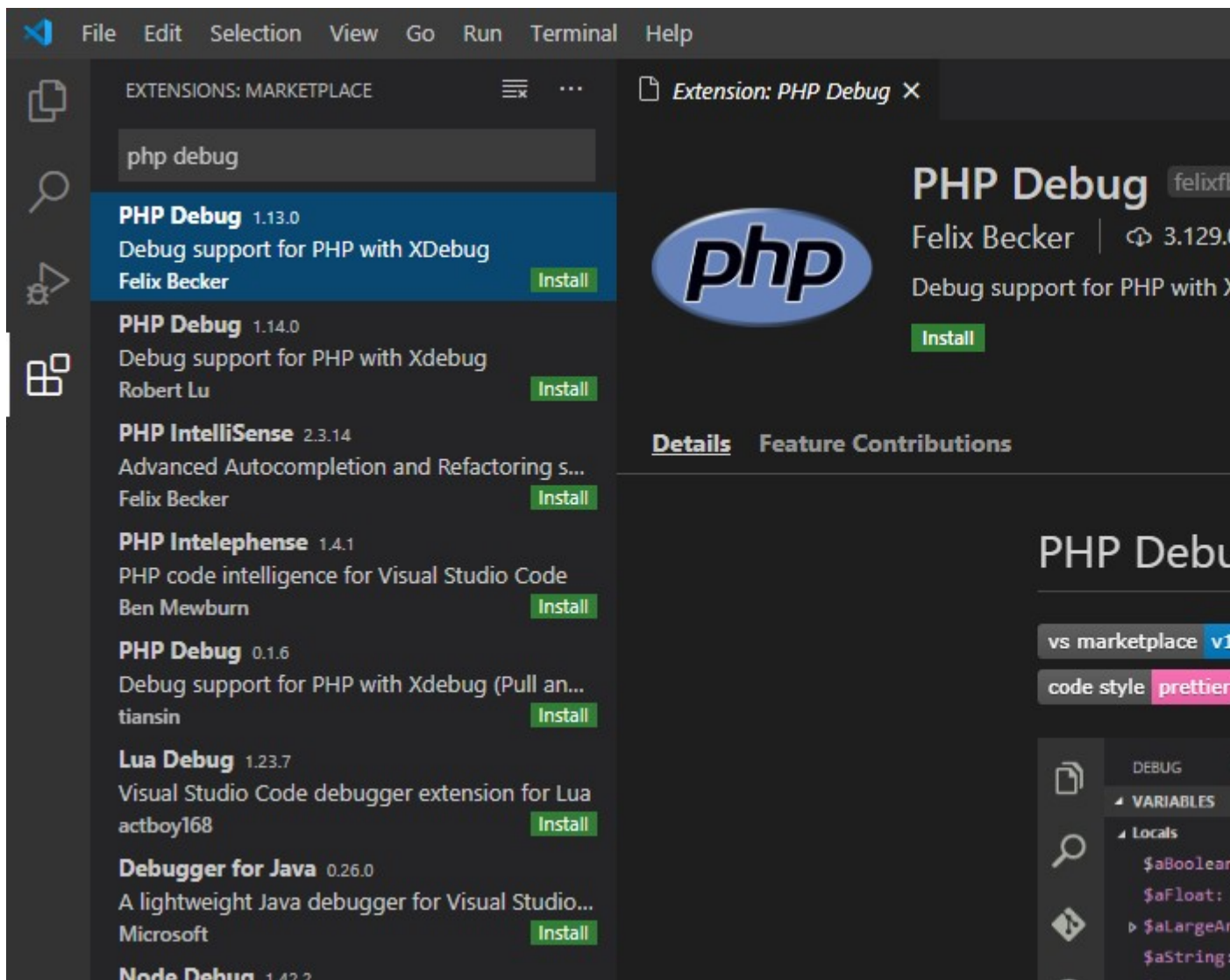
## 2. Installation Visual Studio Code

- Aktuelle Version von Visual Studio Code herunterladen: <https://code.visualstudio.com>
- Visual Studio Code installieren
- Visual Studio Code starten

### 2.1. Erweiterung installieren

- Wählen sie im linken Menü das Extension-Symbol aus (Erweiterungen)
- Geben sie den Suchbegriff «php debug» ein
- Wählen sie die Extension «PHP Debug» von Felix Becker aus und installieren sie die Erweiterung indem Sie den Knopf «Install» betätigen

Abbildung 1.

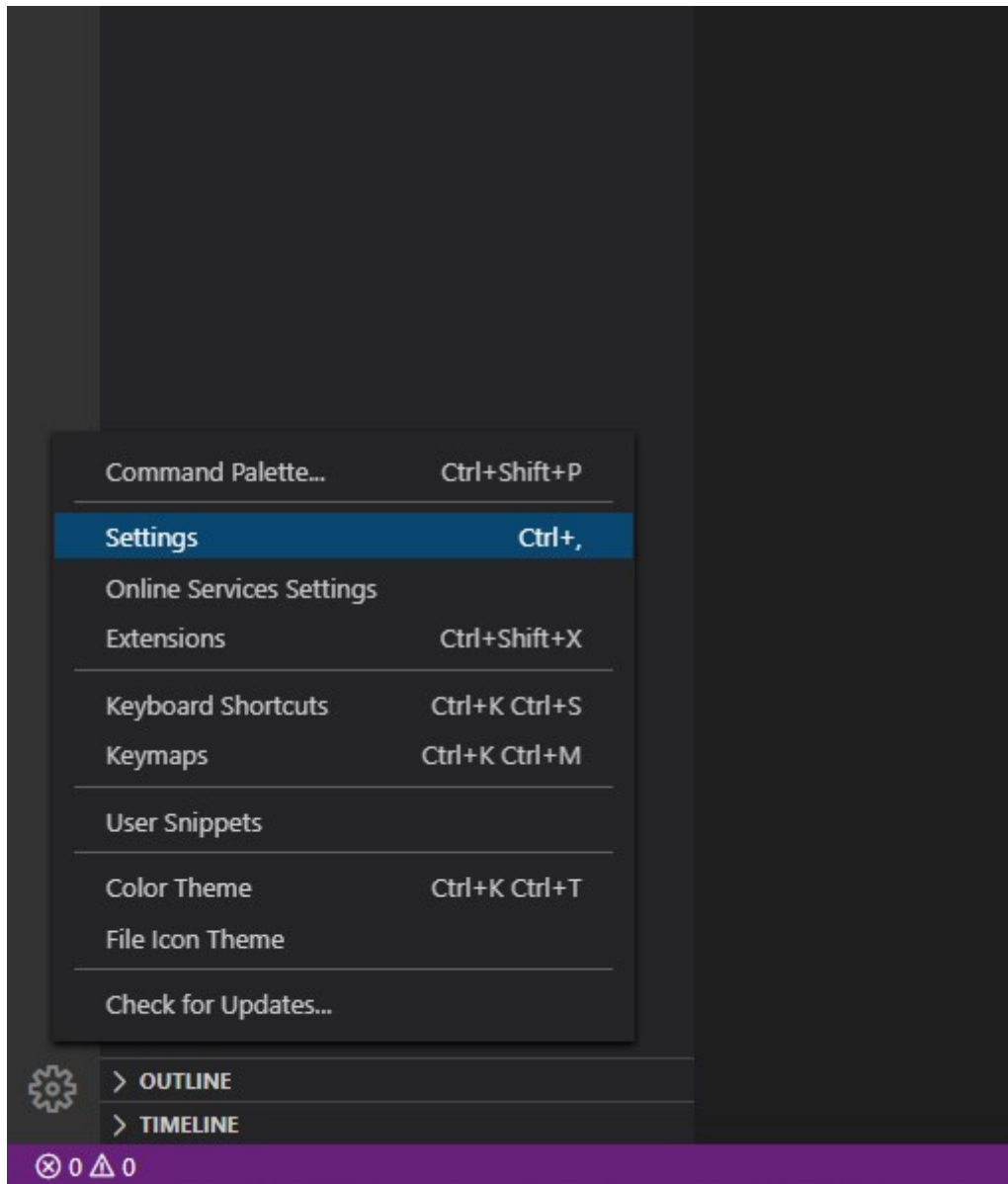


## 3. Konfiguration iG/Script Debugger

### 3.1. Einstellungen

Wählen sie unten links das Zahnradsymbol aus und wechseln zu den Settings (Einstellungen).

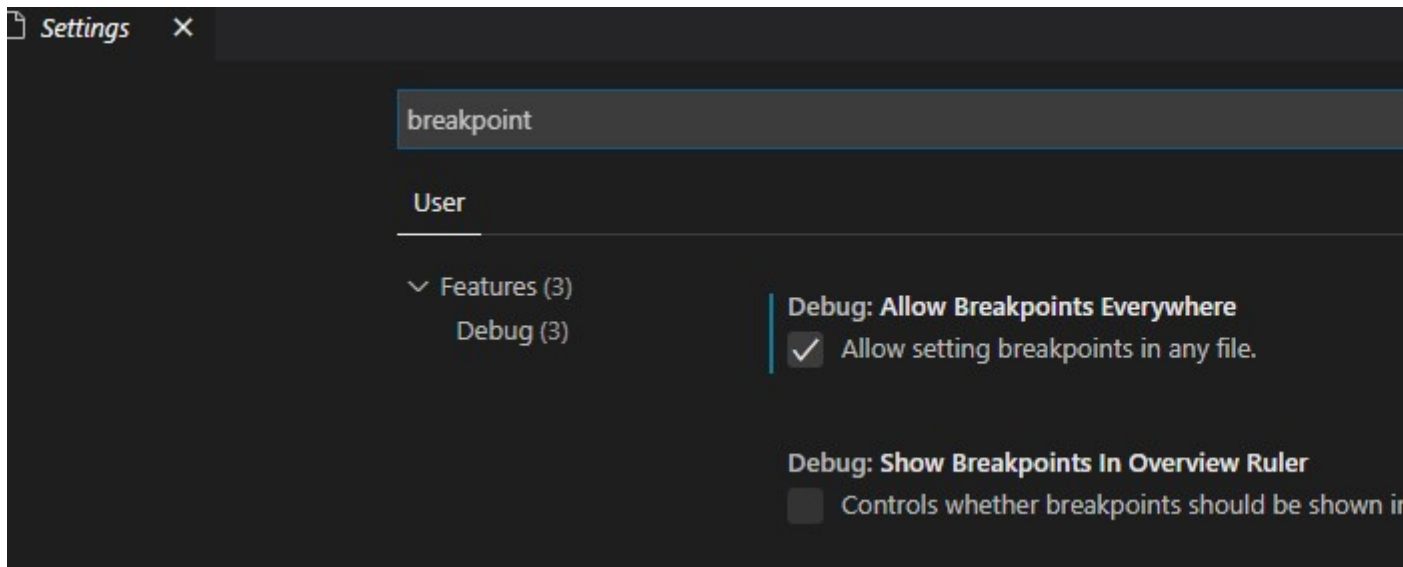
Abbildung 2.



Geben Sie den Suchbegriff: "breakpoint" ein.

Setzen Sie den Haken bei "Allow setting breakpoints in any file".

### Abbildung 3.

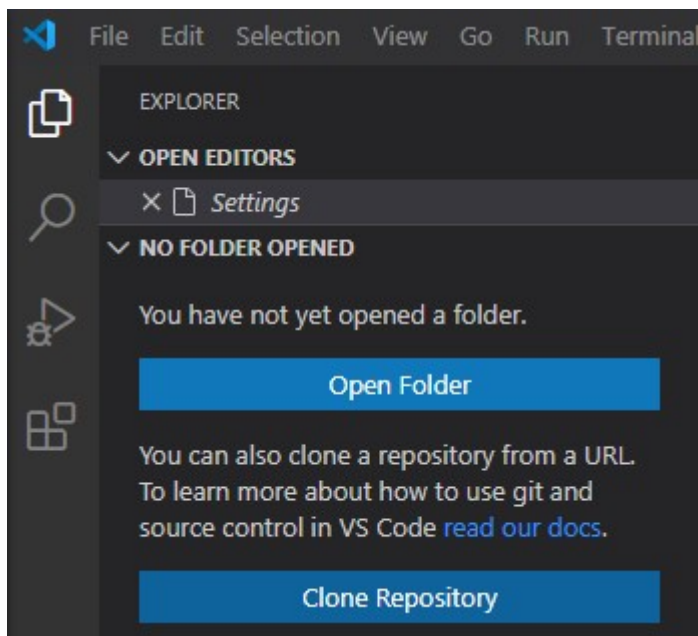


## 3.2. Explorer

Wählen sie im linken Menü das Explorer-Symbol aus.

Öffnen sie den Hauptordner der INTERLIS Tools oder des GeoShop über “Open Folder”.

### Abbildung 4.



## 3.3. Testskript erstellen

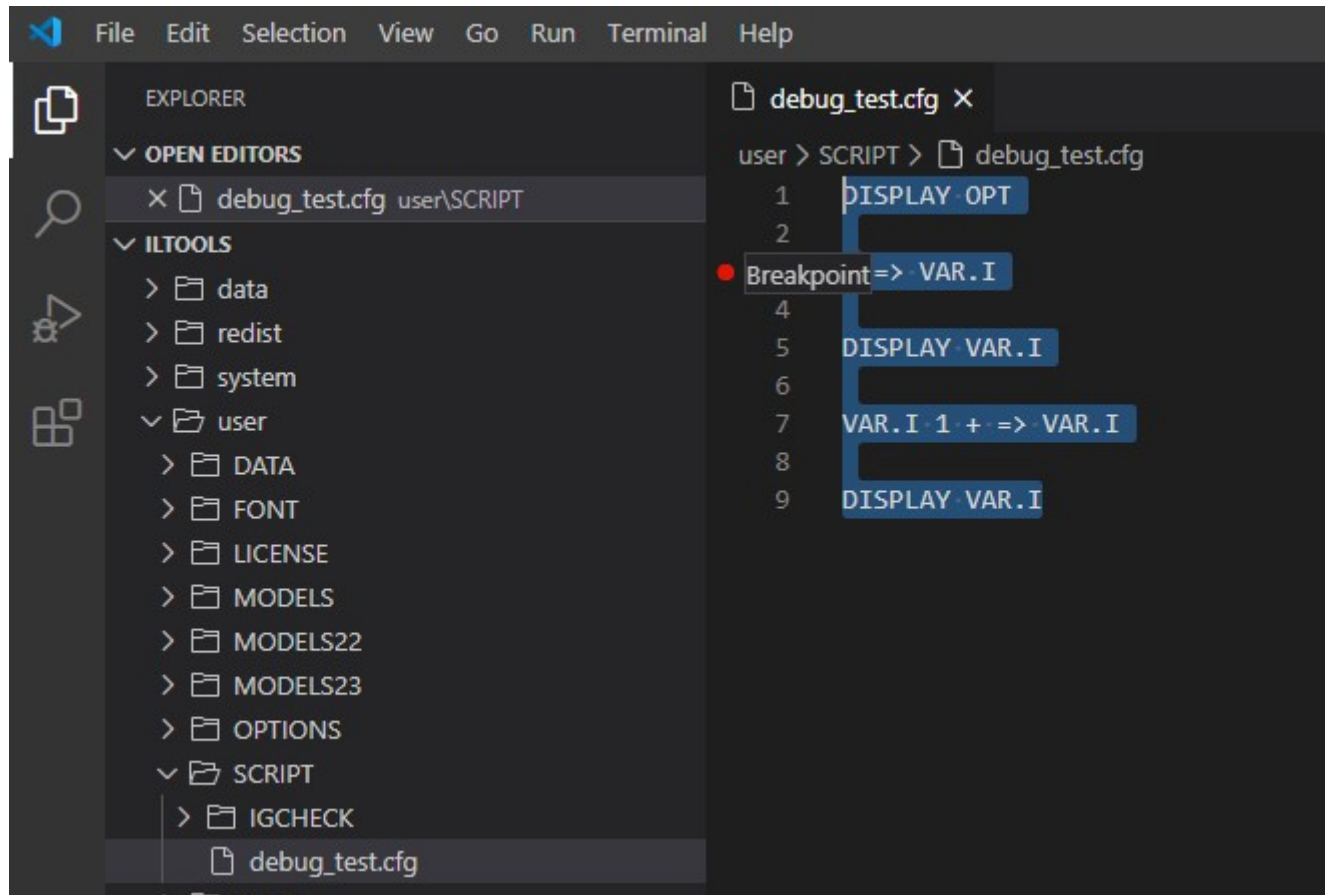
Erstellen sie unter \user\script ein Testskript (Zum Beispiel: debug\_test.cfg).

```
DISPLAY OPT
```

```
0 => VAR.I  
  
DISPLAY VAR.I  
  
VAR.I 1 + => VAR.I  
  
DISPLAY VAR.I
```

Setzen sie einen Breakpoint (zum Beispiel in Zeile 3), indem sie links neben der Zeilennummer den Klick ausführen.

**Abbildung 5.**



### 3.4. Debug-Konfiguration erstellen

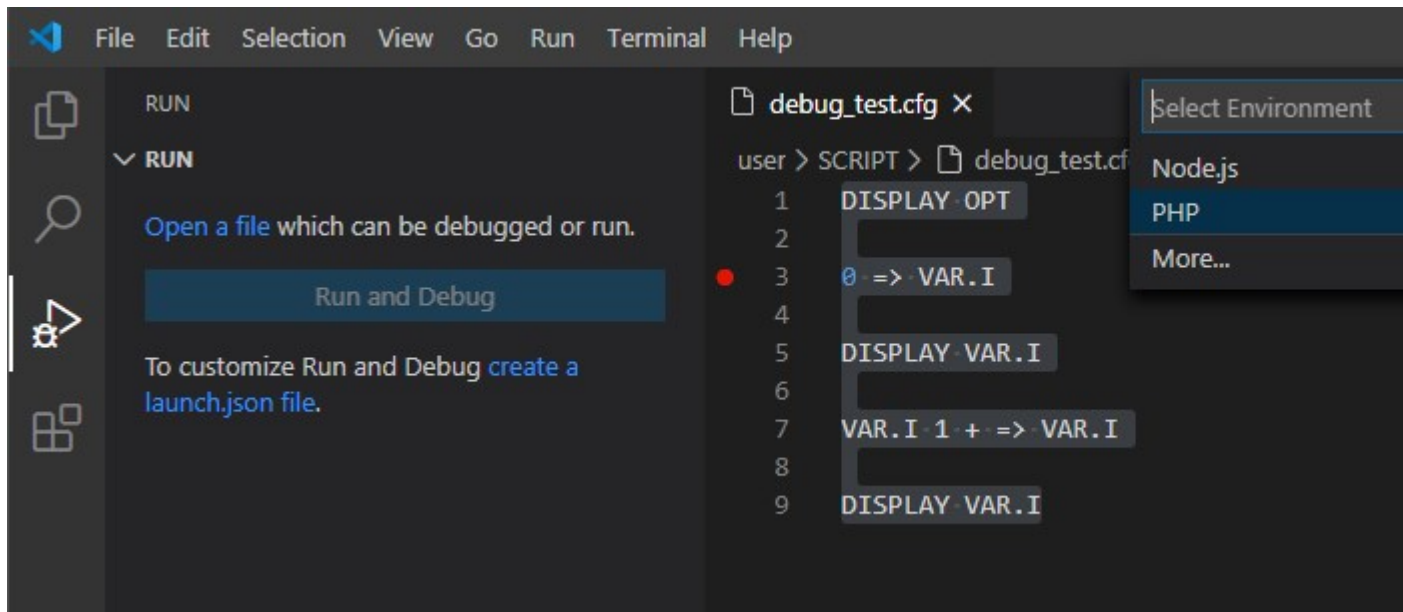
Wechseln sie im linken Menü zum Reiter “Run”.

Wählen sie “create a launch.json file” aus.

Bei “Select Environment” können sie PHP auswählen.



Abbildung 6.



Ändern sie die "launch.json" Datei ab, indem Sie die Konfiguration wie folgt überschreiben:

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "ICS Listen",
      "type": "php",
      "request": "launch",
      "port": 9000,
      "log": false,
      "stopOnEntry": false,
      "xdebugSettings": {
        "supports_async": 0
      }
    },
    {
      "name": "ICS Launch (32Bit)",
      "port": 9000,
      "request": "launch",
      "type": "php",
      "program": "${file}",
      "args": [
        "-xdebug",
        "localhost:9000",
        "-script",
        "${file}"
      ],
      "cwd": "${workspaceFolder}",
      "log": false,
      "externalConsole": false,
      "stopOnEntry": true,
      "runtimeExecutable": "${workspaceFolder}\\system\\BIN\\ics.exe",
    }
  ]
}
```

```
    "xdebugSettings": {
      "supports_async": 0
    }
  },
  {
    "name": "ICS Launch (64Bit)",
    "port": 9000,
    "request": "launch",
    "type": "php",
    "program": "${file}",
    "args": [
      "-xdebug",
      "localhost:9000",
      "-script",
      "${file}"
    ],
    "cwd": "${workspaceFolder}",
    "log": false,
    "externalConsole": false,
    "stopOnEntry": true,
    "runtimeExecutable": "${workspaceFolder}\\system\\BIN64\\ics.exe",
    "xdebugSettings": {
      "supports_async": 0
    }
  }
]
}
```

Speichern sie die Änderungen.

## 4. Funktionsweise des Debuggers

### 4.1. ICS Launch (32-Bit) / ICS Launch (64-Bit)

Mit dem Befehl “ICS-Launch” können sie ICS-Skripte direkt aufrufen und debuggen. Allerdings können im Vergleich zur ICS-Listen-Methode keine Optionen dynamisch übergeben werden.

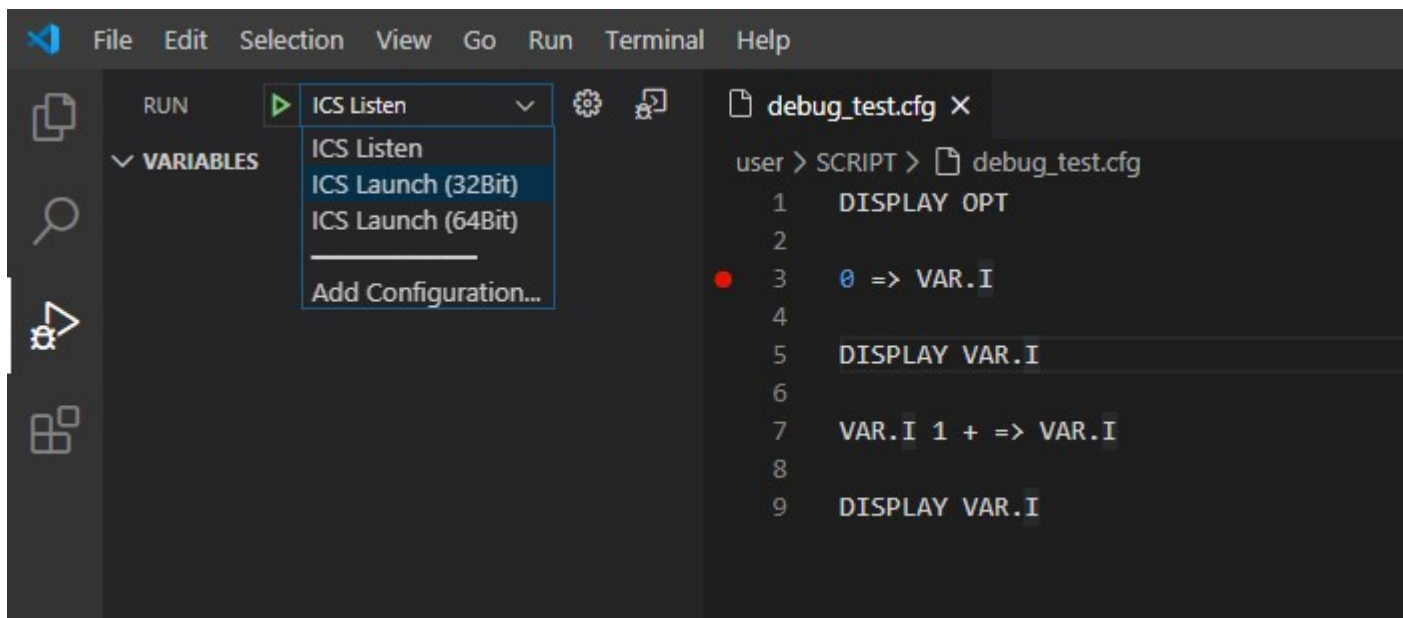
Wählen sie im Dropdown-Menü “ICS Launch (32Bit)” oder “ICS Launch (64Bit)” aus.

Sie können nun das aktuell ausgewählte Skript, in unserem Fall `debug_test.cfg`, debuggen.

Lassen sie das Skript laufen, indem Sie den grünen Run-Button neben der Auswahlliste betätigen.

Nun befinden wir uns im Debug-Modus.

#### Abbildung 7.



### 4.2. ICS Listen

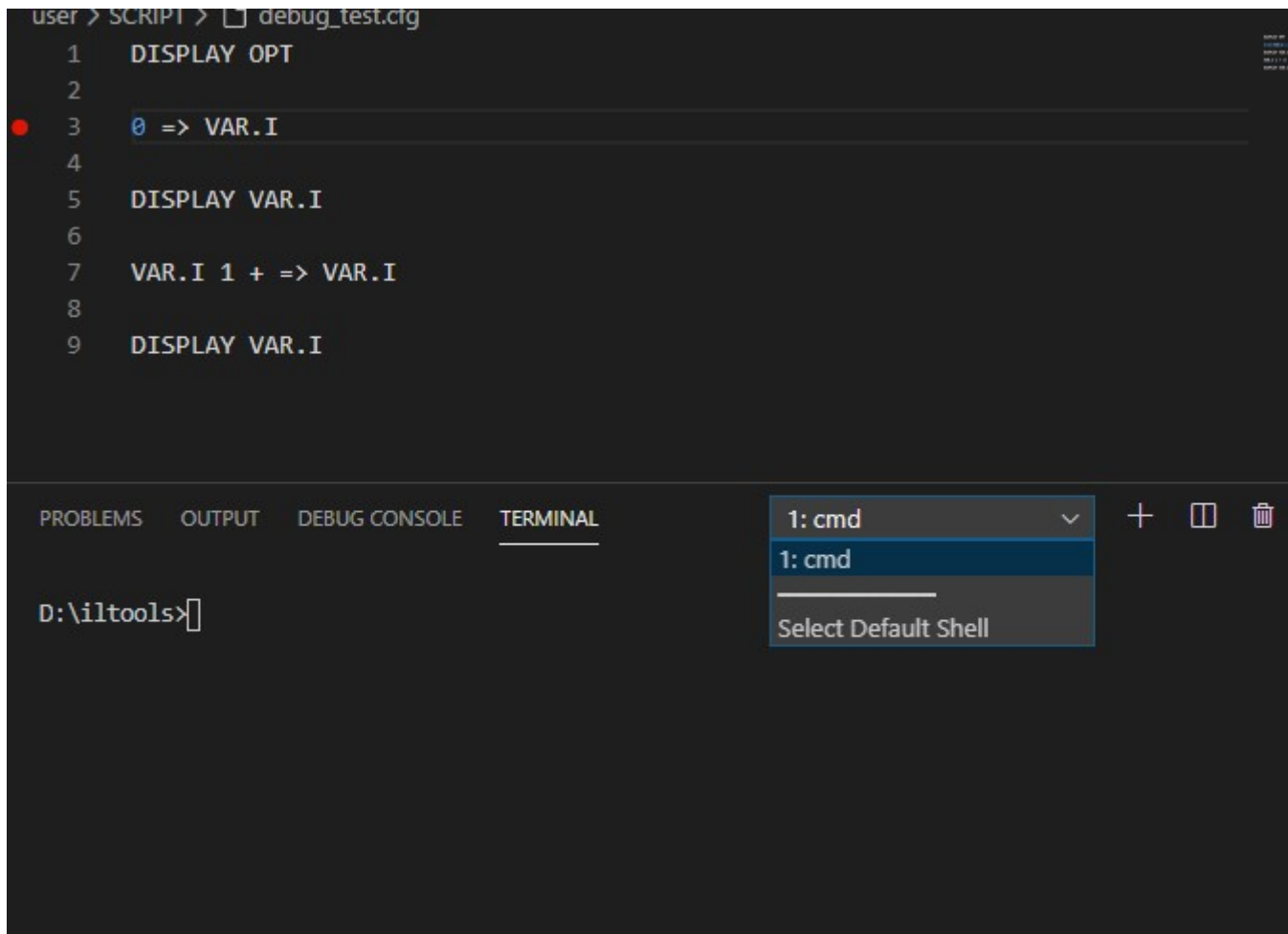
Mit dem Befehl “ICS-Listen” können wir uns an beliebige ICS-Skripte, welche wir über Command-Befehle laufen lassen, anhängen. Zudem können die ICS-Parameter bzw. Optionen frei gesetzt werden.

Wählen sie im Dropdown-Menü “ICS Listen” aus.

Lassen sie den ICS-Listener laufen, indem Sie den grünen Run-Button neben der Auswahlliste betätigen.

Nun wechseln sie zum Terminal und wählen, falls noch nicht der Fall, “Command Prompt” als Default-Shell aus.

## Abbildung 8.



Geben sie im Terminal den Befehl: `system\bin\ics.exe -script \script\debug_test.cfg -xdebug` ein.

Nun befinden wir uns im Debug-Modus.

## 4.3. Erläuterung Debugging

Im Debug-Modus sehen wir nun verschiedene Bereiche und die Steuerungseinheit des Debuggers oben rechts.

## **Abbildung 9.**

The screenshot displays the iG/Script Debugger interface with the following components:

- Top Bar:** RUN button, ICS Listen dropdown, and icons for settings and help.
- VARIABLES:**
  - Local: (empty)
  - Global:
    - CLASSES: MAP (14)
    - NATIVE: MAP (0)
    - PROCEDURES: MAP (0)
    - ICS: MAP (366)** (highlighted)
    - MESSAGE: MAP (13)
    - OPT: MAP (12)
    - ICSCPU: MAP (7)
      - IN: MAP (0)
      - OUT: MAP (0)
      - VAR: MAP (0)
    - SOCKET: MAP (0)
- WATCH:**
  - IN: MAP (0)
  - OUT: MAP (0)
  - 'IN' EXISTS: TRUE
- CALL STACK:**
  - PAUSED ON BREAKPOINT
  - MAIN | debug\_test.cfg | 3:1
- BREAKPOINTS:**
  - Notices
  - Warnings
  - Errors
  - Exceptions
  - Everything
  - debug\_test.cfg user\SCRIPT
- Code Editor:**

```

user > SCRIPT > debug_test.cfg
1  DISPLAY OPT
2
3  0 => VAR.I
4
5  DISPLAY VAR.I
6
7  VAR.I 1 + => VAR.I
8
9  DISPLAY VAR.I
                    
```
- Terminal:**

```

D:\iltools>system\bin\ics.exe -script \script\debu
reading script d:\iltools\user\script\debug_test.c

*****
*                                     *
* INTERLIS-Tools                       *
*                                     *
* Version 2020.0.861 (03.06.2020) *
*                                     *
* (c) infoGrips 1997-2020             *
*                                     *
*****

*****
* License Information:                 *
* infoGrips GmbH                       *
* FOR EVALUATION USE ONLY              *
* ICS Kernel Version 2020.0/32bit *
*****
                    
```

### 4.3.1. Variablen

Unter den Variablen sind die lokalen-, globalen- und falls vorhanden, die klassenspezifischen Variablen, zu finden.

### 4.3.2. Watch

Im Reiter Watch können Variablen und auch Ausdrücke ausgewertet werden.

Beispiele: "IN" oder "OUT" oder "IN EXISTS"

### 4.3.3. Call Stack

Im Call Stack ist die Aufrufhierarchie zu finden. Falls Prozeduren in eingebundenen Skripten (Bibliotheken, Modulen) aufgerufen werden, kann die Aufrufreihenfolge der Skripte sehr gut nachvollzogen werden.

### 4.3.4. Breakpoints

Hier sind alle gesetzten Haltepunkte aufgelistet. Sie können über Rechtsklick auch eine Kondition für den Breakpoint setzen. Die Kondition muss TRUE oder FALSE zurückliefern.

Beispiel: VAR.I = 0

### 4.3.5. Steuerung

Die Steuerung setzt sich aus den Befehlen "Run", "Step Over", "Step Into", "Step Out", "Restart" und "Stop" zusammen.

#### Abbildung 10.



Befehl "Run": Führt das Skript bis zum nächsten Haltepunkt, oder bis zum Ende des Skripts, aus.

Befehl "Step Over": Ausführung bis zur nächsten Zeile im selben Skript.

Befehl "Step Into": Ausführung bis zur nächsten Zeile. Spring in die Unterprozedur, falls verfügbar.

Befehl "Step Out": Springt aus der aktuellen Prozedur zurück in die aufzurufende Prozedur.

Befehl "Restart": Stoppt das Skript und startet den Debugger neu.

Befehl "Stop": Stoppt das Skript und beendet den Debugger.