Administration

idgard Sync



Table of Contents

Administration ---- 4
Installer
Systemvoraussetzungen ---- 5
Update-Richtlinie ---- 6
Bereitstellungsleitfaden ---- 8
Installer-Benutzervariablen (öffentliche MSI-Eigenschaften) ---- 13
Konfigurationsdateien ---- 15
MSI-Setup Screenshots ---- 17

Administration

Willkommen zu *Administration* of **idgard Sync**.

idgard Sync ist eine Windows-App, die Ihren idgard-Arbeitsbereich direkt in Ihren täglichen Desktop-Workflow integriert.

> **NOTE**
> Mit idgard Sync sind Ihre Dateien im Windows-Explorer wie in einem vertrauten Cloud-Laufwerk verfügbar, während die Sicherheitsstandards von idgard erhalten bleiben.

Installer

- Systemvoraussetzungen
- Update-Richtlinie
- Bereitstellungsleitfaden
- Installer-Variablen (Öffentliche MSI-Eigenschaften)
- Konfigurationsdateien
- Screenshots

Systemvoraussetzungen

Die folgenden Voraussetzungen müssen erfüllt sein, um idgard Sync zu installieren und zu betreiben.

Betriebssystem

Anforderung	Minimum
Betriebssystem	Windows 11 (64-Bit)

> **IMPORTANT**
>
> **Windows 10** wird offiziell **nicht** unterstützt. Die Anwendung kann unter Windows 10 22H2 (Build 19045) oder höher grundsätzlich ausgeführt werden — das zugrunde liegende Windows SDK adressiert Build 19041 (20H2), der dieselbe API-Oberfläche abdeckt wie 20H2, 21H1, 21H2 und 22H2. Eingeschränkte Funktionalität und Stabilitätsprobleme sind jedoch möglich und werden nicht behoben.

Laufzeitabhängigkeiten

Der Installer lädt fehlende Laufzeitkomponenten automatisch herunter und installiert sie.

Komponente	Version
.NET Desktop Runtime	10.0 (x64)
Windows App Runtime	2.5.1 (x64)

> **NOTE**
>
> Beide Laufzeitkomponenten sind im Exe-Installer enthalten und werden automatisch eingerichtet, sofern sie noch nicht auf dem System vorhanden sind. Sie sind nicht im MSI-Setup enthalten und müssen vorab separat installiert werden.

Update-Richtlinie

Dieses Dokument beschreibt, wie idgard Sync Updates seiner zentralen Abhängigkeiten verfolgt und übernimmt.

Komponente	Strategie	Wartezeit nach Support-/Releaseende
Windows OS (Feature-Update)	Minimum erhöhen nach MS-Supportende	3 – 6 Monate
Windows App SDK (Hauptversion)	Neueste Hauptversion ≥ 6 Monate alt	≥ 6 Monate
.NET Desktop Runtime (Haupt-/LTS)	Nur LTS-Versionen übernehmen	6 – 12 Monate nach GA
.NET Desktop Runtime (Patch)	Immer neuesten Patch der aktuellen LTS	Sofort

Windows-Betriebssystem

Wir orientieren uns am Supportzeitraum von Microsoft für Windows-Feature-Updates.

Regel: Wir erhöhen die mindestens unterstützte Windows-Version **3 bis 6 Monate nachdem Microsoft den Mainstream-Support** für die aktuell als Minimum gelistete Version beendet. Der genaue Zeitpunkt innerhalb dieses Fensters hängt von der Release-Reife und dem Testaufwand ab.

Ressource	Link
Windows 10 Releaseinformationen	learn.microsoft.com – Windows 10
Windows 11 Releaseinformationen	learn.microsoft.com – Windows 11

Windows App SDK

Regel: Wir setzen auf die **neueste Hauptversion, die zum Zeitpunkt eines Releases mindestens 6 Monate alt ist**. Eine brandneue Hauptversion wird nicht sofort übernommen; wir warten mindestens 6 Monate nach deren Erscheinen, bevor wir sie als Mindestanforderung festlegen. Innerhalb der gewählten Hauptversion liefern wir stets die neueste verfügbare Version aus.

Ressource	Link
Windows App SDK – Downloads & Versionshinweise	learn.microsoft.com – Windows App SDK

.NET Desktop Runtime

Regel: Wir setzen ausschließlich auf **Long-Term Support (LTS)-Versionen** (z. B. .NET 10, .NET 12). Eine neue LTS-Version wird **6 bis 12 Monate nach ihrer GA-Veröffentlichung** übernommen, sobald sich das Ökosystem stabilisiert hat. Innerhalb der übernommenen Hauptversion liefern wir stets den **neuesten Minor- oder Patch-Release** aus (aktuell z. B. 10.0).

Nicht-LTS-Versionen (Standard-Term Support, STS) werden nicht als Pflichtlaufzeit eingesetzt.

Ressource	Link
.NET – Downloads & Releaseplan	dotnet.microsoft.com

Bereitstellungsleitfaden

Dieser Leitfaden erklärt gängige Methoden zur Bereitstellung eines MSI-Pakets, von der interaktiven Installation bis hin zum vollständig automatisierten Rollout.

`<PRODUCTNAME>` ist "idgard Sync". `<USERNAME>` ist der Windows-Benutzername des Zielbenutzers.

Best Practices

- Halten Sie Bereitstellungsschritte vorhersehbar und wiederholbar
- Nutzen Sie Protokollierung für unbeaufsichtigte Installationen
- Prüfen Sie das Installationsergebnis nach dem Rollout

Installation

Installation mit vollständiger UI (Weiter → Weiter → Fertigstellen)

Verwenden Sie diese Option, wenn ein Benutzer die Installation manuell auf einer Workstation durchführt (der Benutzer benötigt Administratorrechte).

Schritte

1. Doppelklicken Sie auf die `.msi`-Datei.
2. Folgen Sie dem Installationsassistenten.
3. Klicken Sie sich mit **Weiter** durch die Dialoge.
4. Klicken Sie am Ende auf **Fertigstellen**.

> **NOTE**
>
> **Am besten geeignet für**
>
> - manuelle Installationen
> - kleine Unternehmen
> - IT-begleitete geführte Einrichtung

Unbeaufsichtigte Installation (`msiexec` ohne UI)

Verwenden Sie diese Option für die unbeaufsichtigte Bereitstellung per Skript, Intune, SCCM, GPO oder RMM-Tools.

```
msiexec /i "setup.msi" /qn /norestart
```

Nützliche Optionen

- `/i` = Paket installieren
- `/qn` = keine UI (silent)
- `/norestart` = nicht automatisch neu starten
- `/l*v "C:\Temp\install.log"` = ausführliche Protokollierung

Beispiel mit Protokollierung:

```
msiexec /i "setup.msi" /qn /norestart /l*v "C:\Temp\install.log"
```

> ⓘ **NOTE**
>
> **Am besten geeignet für**
>
> - Installationen mit detaillierter Protokollierung
> - mittelgroße Unternehmen
> - zentralisierte IT-Bereitstellung

Installation mit benutzerdefinierten Eigenschaften

Verwenden Sie öffentliche MSI-Eigenschaften (müssen **GROSSGESCHRIEBEN** sein), um das Verhalten des Installers zu steuern.

Weitere Informationen zu allen verfügbaren Eigenschaften von **idgard Sync** finden Sie unter Installer-Benutzervariablen (öffentliche MSI-Eigenschaften).

Beispiel: Silent-Installation mit booleschem Wert `MY_BOOLEAN_PROPERTY=1` – 1 für Eigenschaft aktivieren (ja, true)

```
msiexec /i "setup.msi" /qn MY_BOOLEAN_PROPERTY=1
```

Beispiel: Silent-Installation mit booleschem Wert `MY_BOOLEAN_PROPERTY=0` – o für Eigenschaft deaktivieren (nein, false)

```
msiexec /i "setup.msi" /qn MY_BOOLEAN_PROPERTY=0
```

Beispiel: Silent-Installation mit Zeichenfolge `MY_STRING_PROPERTY` – das gesamte `NAME=Wert`-Paar in Anführungszeichen setzen `"NAME=Mein Wert"`, wenn der Wert Leerzeichen enthält

```
msiexec /i "setup.msi" /qn "MY_STRING_PROPERTY=Lorem Ipsum"
```

Gängige Muster

- `MY_BOOLEAN_PROPERTY=1` oder `MY_BOOLEAN_PROPERTY=0` (boolesch)
- `"MY_STRING_PROPERTY=Lorem Ipsum"` (Zeichenfolge mit Leerzeichen – gesamtes Paar in Anführungszeichen)

Konfiguration nach der Bereitstellung über Konfigurationsdateien

Wenden Sie nach der MSI-Bereitstellung umgebungsspezifische Einstellungen mithilfe von Konfigurationsdateien an.

Es gibt einen speziellen Konfigurationsordner: `C:\Users\<USERNAME>\AppData\Local\<PRODUCTNAME>\config`.

Typischer Ablauf

1. MSI-Paket bereitstellen.
2. Konfigurationsdatei(en) (zum Beispiel: .json, .config, .xml, .ini) in den Konfigurationsordner **config** kopieren.
3. **Sichere Werte** (Zertifikate, Tokens, Endpunkte) über **Secret-Management/Tooling** setzen.
4. Anwendung neustarten.

Beispielhafter Skriptablauf

```
msiexec /i "setup.msi" /qn /norestart
Copy-Item "C:\IT\configs\my-config.json" "C:\Users\<USERNAME>\AppData\Local\
<PRODUCTNAME>\configs\my-config.json" -Force

# Anwendung neu starten, z.B. über das Startmenü
```

Deinstallation

Interaktive Deinstallation durch den Anwendungsbenutzer

Wenn der angemeldete Benutzer idgard Sync für sich selbst deinstalliert – über **Einstellungen → Apps** oder die **Systemsteuerung** – werden alle benutzerspezifischen Daten automatisch bereinigt:

- Sync-Root-Registrierung
- Registrierungseinträge
- idgard Sync-Laufwerk
- Kontextmenü-Erweiterung

Dies ist der empfohlene Deinstallationsweg für Endbenutzer.

Silent-Deinstallation oder Deinstallation durch einen anderen Benutzer

Wenn die Anwendung unbeaufsichtigt deinstalliert wird (z. B. über Intune, SCCM, GPO oder `msiexec /qn`) oder durch einen Administrator, der nicht der idgard Sync-Benutzer ist, läuft das MSI ohne Zugriff auf die Benutzersitzung und **kann benutzerspezifische Daten nicht bereinigen**.

Die folgenden benutzerspezifischen Daten verbleiben nach der Deinstallation für jeden Benutzer, der idgard Sync auf diesem Rechner aktiv genutzt hat, und müssen für jeden dieser Benutzer separat entfernt werden:

- Sync-Root-Registrierung
- Registrierungseinträge unter `HKCU`
- idgard Sync-Laufwerk
- Kontextmenü-Erweiterung

Silent-Deinstallationsbefehl

```
msiexec /x "setup.msi" /qn /norestart
```

Manuelle Bereinigung pro Benutzerkonto

Führen Sie die folgenden Schritte für jeden betroffenen Benutzer in einer **erhöhten PowerShell-Sitzung** aus.

> **NOTE**
> Ersetzen Sie `<USERNAME>` durch den Windows-Benutzernamen und `<SID>` durch die entsprechende Sicherheitskennung (SID).
> Alle lokalen Benutzer mit SIDs auflisten: `Get-LocalUser | Select-Object Name, SID`

1. Anwendungsdaten löschen

```
Remove-Item "C:\Users\<USERNAME>\AppData\Local\<PRODUCTNAME>" -Recurse -Force
```

2. Registrierungseinträge entfernen

Falls der Benutzer nicht angemeldet ist, zuerst die Registry-Struktur laden:

```
& "C:\Windows\System32\reg.exe" load "HKU\<SID>" "C:\Users\<USERNAME>\NTUSER.DAT"
```

> **NOTE**
> Der beim `reg.exe load` verwendete Name (hier `<SID>`) muss in allen nachfolgenden Befehlen konsistent verwendet werden. Es kann ein beliebiger eindeutiger Name gewählt werden — die Verwendung der SID vermeidet Konflikte.

Anwendungseinstellungen und Autostart-Eintrag löschen:

```
& "C:\Windows\System32\reg.exe" delete "HKU\<SID>\Software\<PRODUCTNAME>" /f
& "C:\Windows\System32\reg.exe" delete "HKU\
<SID>\Software\Microsoft\Windows\CurrentVersion\Run" /v "<PRODUCTNAME>" /f
& "C:\Windows\System32\reg.exe" delete "HKU\
<SID>\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run" /v "
<PRODUCTNAME>" /f
```

Registry-Struktur anschließend wieder entladen, falls sie manuell geladen wurde:

```
& "C:\Windows\System32\reg.exe" unload "HKU\<SID>"
```

3. Kontextmenü-Erweiterung entfernen

```
Get-AppxPackage -Name "Idgard.Sync.WinShell" -User <SID> | Remove-AppxPackage -
User <SID>
```

4. Sync-Root-Registrierung aufheben

Die Sync-Root-Registrierung ist in HKLM gespeichert und kann direkt durch einen Administrator gelöscht werden. Der Schlüsselname folgt dem Muster `idgard!<SID>!<USERNAME>`:

```
& "C:\Windows\System32\reg.exe" delete
"HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SyncRootManager\idgard!
<SID>!<USERNAME>" /f
```

5. Veralteten Sync-Root-Namespace-Eintrag entfernen

Nach der Deinstallation kann ein veralteter Eintrag in der Shell-Namespace-Registrierung des Benutzers verbleiben, der dazu führt, dass Explorer einen verwaisten Eintrag in der Navigationsleiste anzeigt.

Zuerst alle Einträge unter dem NameSpace-Schlüssel auflisten, um den idgard-Eintrag anhand seines Anzeigenamens zu identifizieren:

```
New-PSDrive -Name HKU -PSProvider Registry -Root HKEY_USERS -ErrorAction
SilentlyContinue | Out-Null
Get-ChildItem "HKU:\
<SID>\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace" |
    ForEach-Object {
        [PSCustomObject]@{
            GUID        = $_.PSChildName
            DisplayName = (Get-ItemProperty $_.PSPath).'(default)'
        }
    }
```

Die GUID des idgard-Eintrags aus der Ausgabe notieren und anschließend löschen:

```
& "C:\Windows\System32\reg.exe" delete "HKU\
<SID>\Software\Microsoft\Windows\CurrentVersion\Explorer\Desktop\NameSpace\
<GUID>" /f
```

Installer-Benutzervariablen (öffentliche MSI-Eigenschaften)

Die folgenden **benutzerdefinierten Eigenschaften** (in GROSSBUCHSTABEN) sind öffentliche MSI-Eigenschaften und können zur Installationszeit über das Befehlszeilentool `msiexec` gesetzt werden.

Weitere Beispiele finden Sie im Bereitstellungsleitfaden.

> **NOTE**
> Öffentliche MSI-Eigenschaften erfordern ein `NAME=WERT`-Paar.
>
> **Beispiele:**
> `MY_BOOLEAN_PROPERTY=1` oder `MY_BOOLEAN_PROPERTY=0` (boolesch)
> `"MY_STRING_PROPERTY=Lorem Ipsum"` (Zeichenfolge mit Leerzeichen – gesamtes Paar in Anführungszeichen)

> **WARNING**
> Der Name allein ist nicht gültig – die Eigenschaft wird **ignoriert**.
> Eigenschaftsnamen (in Großbuchstaben) müssen zu dem passen, was das MSI unterstützt.
> Unbekannte Eigenschaften werden **ohne** Warnung oder Fehler ignoriert.

Verwendungsbeispiele für benutzerdefinierte (überschreibbare) Eigenschaften

Benutzerdefinierte Eigenschaften steuern sowohl das **Verhalten des Installers** als auch das **Laufzeitverhalten** der installierten Anwendung. Eigenschaften, die das Laufzeitverhalten beeinflussen, werden während der Installation in der Windows-Registrierung unter `HKLM\SOFTWARE\`**idgard Sync** gespeichert. Die Anwendung liest diese Werte beim Start oder bei Bedarf.

In den folgenden Beispielen werden optionale `msiexec`-Parameter **weggelassen**, um den Fokus auf den jeweiligen Anwendungsfall der benutzerdefinierten Eigenschaft zu legen.

Anwendungsupdate-Funktionalität steuern – auf `1` setzen, um automatische Updates zu `deaktivieren` (Standardwert ist `0`, Update ist aktiv):

```
msiexec /i "setup.msi" /qn DISABLE_APP_UPDATE="1"
```

EIGENSCHAFT	Registrierungsschlüssel	Werte	Anwendungsfall
DISABLE_APP_UPDATE	DisableAppUpdate	**0**, 1	Anwendungsupdate steuern

> **NOTE**
> Nur Eigenschaften mit einem vollständig großgeschriebenen `PROPERTY_NAME` im WiX-Quellcode sind öffentliche MSI-Eigenschaften.
> Eigenschaften mit `Secure="yes"` werden intern befüllt und können **nicht** über die Befehlszeile überschrieben werden.

Standard-Eigenschaften

> **WARNING**
> **Standard-Eigenschaften** sind zwar zugänglich, sollten aber **nicht** vom Benutzer geändert werden.

Diese **ARP-Eigenschaften** steuern die Darstellung des Produkts in **Software** (Programme und Features).

Eigenschaft	Beschreibung
`ARPCONTACT`	Support-Kontakt in der Softwareverwaltung.
`ARPHELPLINK`	Hilfe-URL in der Softwareverwaltung.
`ARPHELPTELEPHONE`	Support-Telefonnummer in der Softwareverwaltung.
`ARPPRODUCTICON`	Symbol in der Softwareverwaltung.

Interne Eigenschaften

Eigenschaft	Beschreibung
`EXENAME`	Name der ausführbaren Hauptdatei.
`TENANT_PRODUCT_NAME`	Anzeigename des Tenant-Produkts.

Sichere / **Schreibgeschützte** Eigenschaften

Alle sicheren Eigenschaften werden intern gesetzt (`Secure="yes"`) und können **nicht** über die Befehlszeile gesetzt werden!

Eigenschaft	Beschreibung
`PREVIOUSVERSIONSINSTALLED`	Wird durch die Upgrade-Tabelle gesetzt.

Konfigurationsdateien

`PRODUCT_NAME` ist "idgard Sync". `USER_NAME` ist der Windows-Benutzername des aktuellen Benutzers.

Es gibt einen speziellen Konfigurationsordner: `C:\Users\USER_NAME\AppData\Local\PRODUCT_NAME\config`, Konfigurationsdatei(en) (zum Beispiel: .json, .config, .xml, .ini) in den Konfigurationsordner kopieren und Anwendung neustarten.

Konfigurationsdateien sind ein Teil des Bereitstellungsleitfaden.

Allgemeine Verwendung von Konfigurationsdateien:

1. Navigieren Sie im Windows-Explorer zu `%LocalAppData%/PRODUCT_NAME/`.
2. Erstellen Sie einen Ordner `config`, falls dieser noch nicht existiert.
3. Erstellen Sie eine neue JSON-Datei mit dem Namen der Konfiguration, z. B. `my-config.json`.

Verhalten (`_behavior` Eigenschaft)

Alle JSON-Konfigurationsdateien erfordern eine `_behavior`-Einstellung, die steuert, wie die Konfiguration beim Start angewendet wird. Nicht jede Einstellung unterstützt alle `_behavior`-Varianten.

Wenn keine `_behavior` gesetzt wird, oder der Wert nicht vorhanden ist, wird `once` verwendet.

Wert	Beschreibung
`once` *(Standard)*	Liest die Konfigurationsdatei, importiert die Werte und löscht anschließend die Datei. Der Benutzer kann die Werte danach in den App-Einstellungen ändern.
`always`	Liest die Konfigurationsdatei, importiert die Werte und behält die Datei. Die Werte überschreiben bei jedem Start die Benutzereinstellungen.
`OnceIfNotSet`	Wie `once`, jedoch wird ein Wert nur importiert, wenn noch keine App-Einstellung dafür vorhanden ist.

Web-Proxy

Konfiguration eines Web-Proxy via Konfigurationsdatei.

- **Dateiname** der Konfigurationsdatei: `proxy.json`
- Unterstützte **_behavior**: `once`, `always` und `OnceIfNotSet`.

Beispiel für `proxy.json`.

```
{
    "_behavior": "once",
    "ProxyAddress": "my.proxy-server.tld",
    "ProxyPort": 8080
}
```

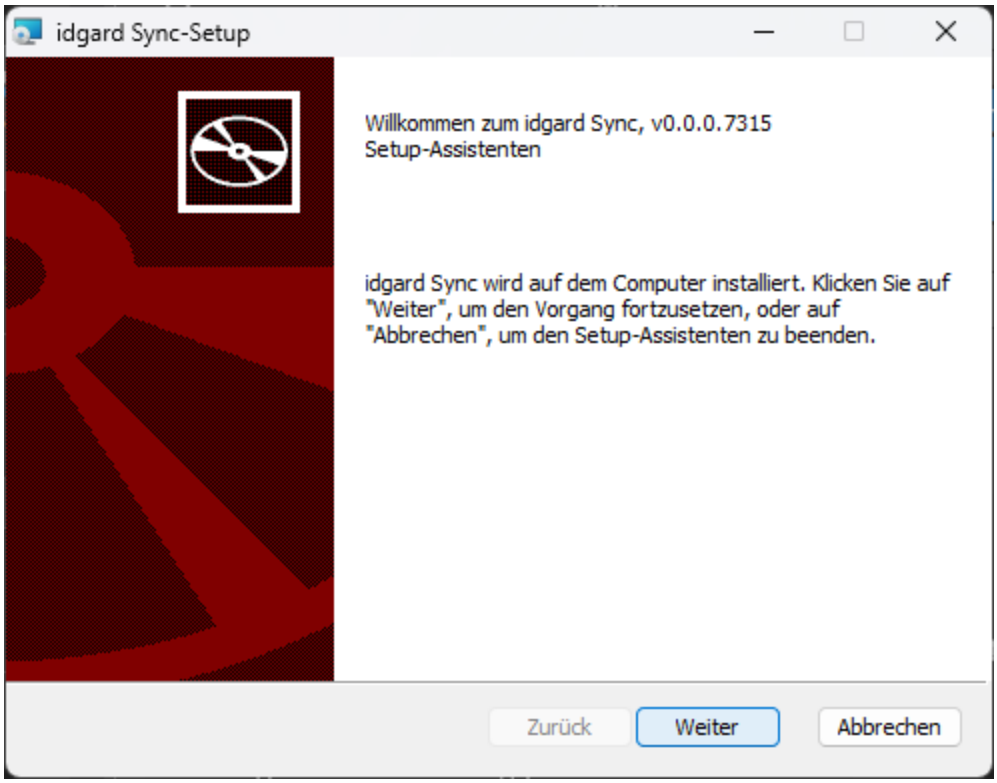
Konfigurieren Sie `ProxyAddress` (kann eine IP-Adresse oder eine URL sein) und `ProxyPort` entsprechend Ihren Anforderungen.

Die Verwendung des Proxys ist außerdem in den Log-Dateien sichtbar:

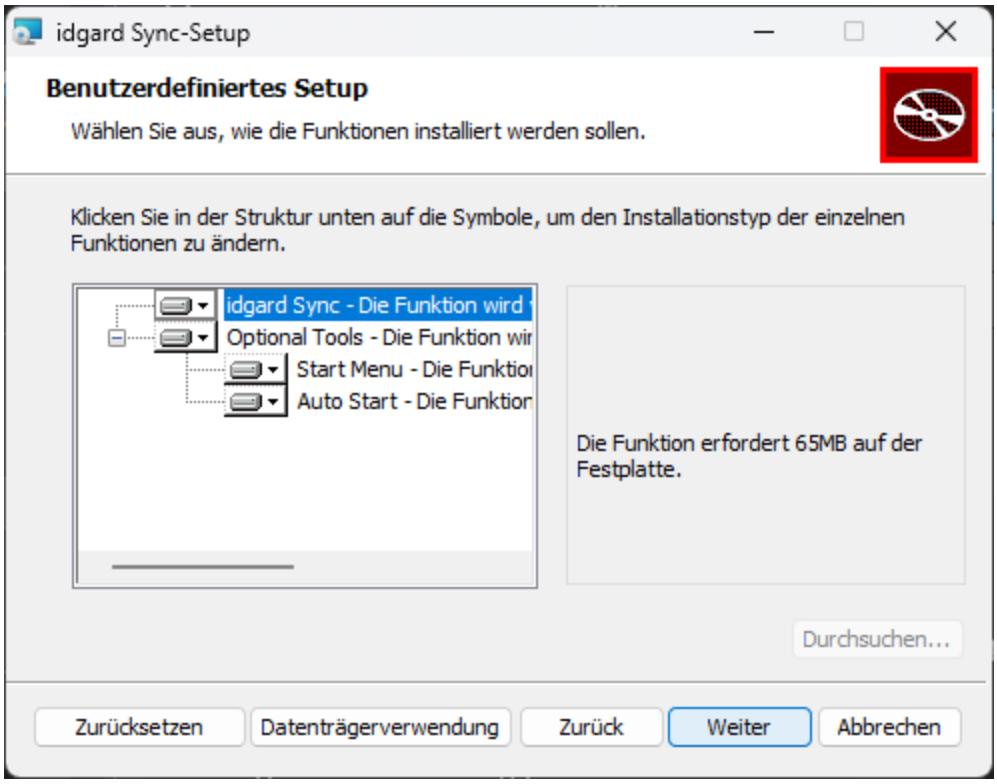
```
[YYYY-mm-dd HH:MMM:ss.fff INF] Parsing the proxy config file
'C:\Users\USER_NAME\AppData\Local\PRODUCT_NAME\config\proxy.json' was successful. Address:
my.proxy-server.tld, Port: 8080
```

MSI-Setup Screenshots

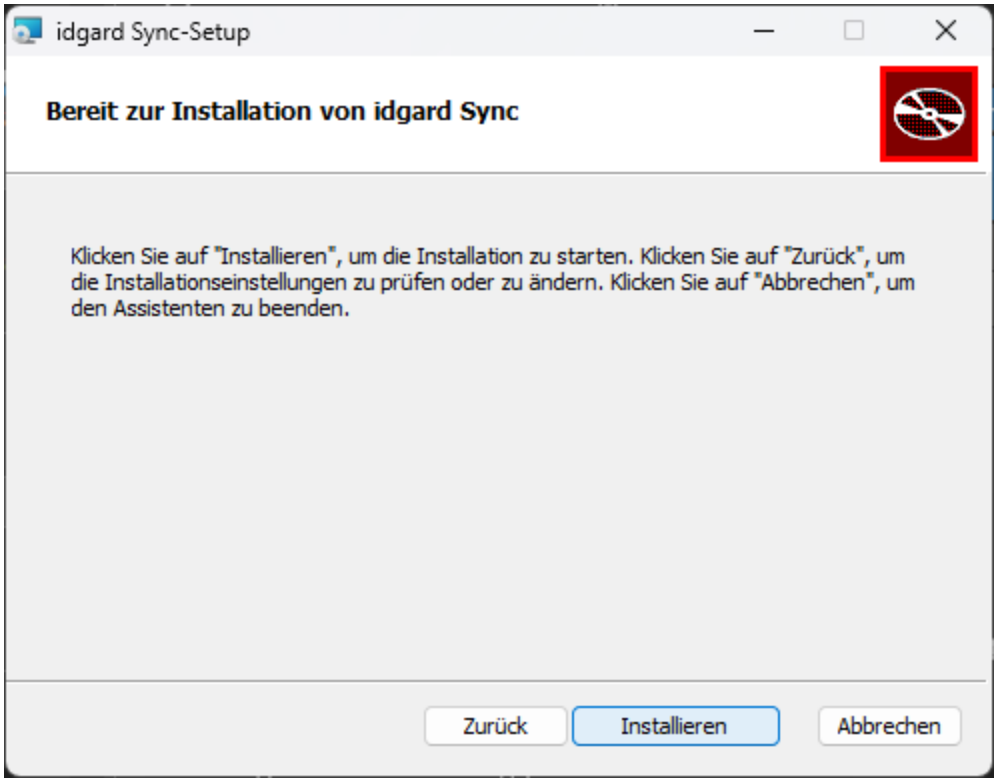
Wir nutzen einen Weiter-Weiter-Ende Ansatz für das Setup



Willkommen Dialog

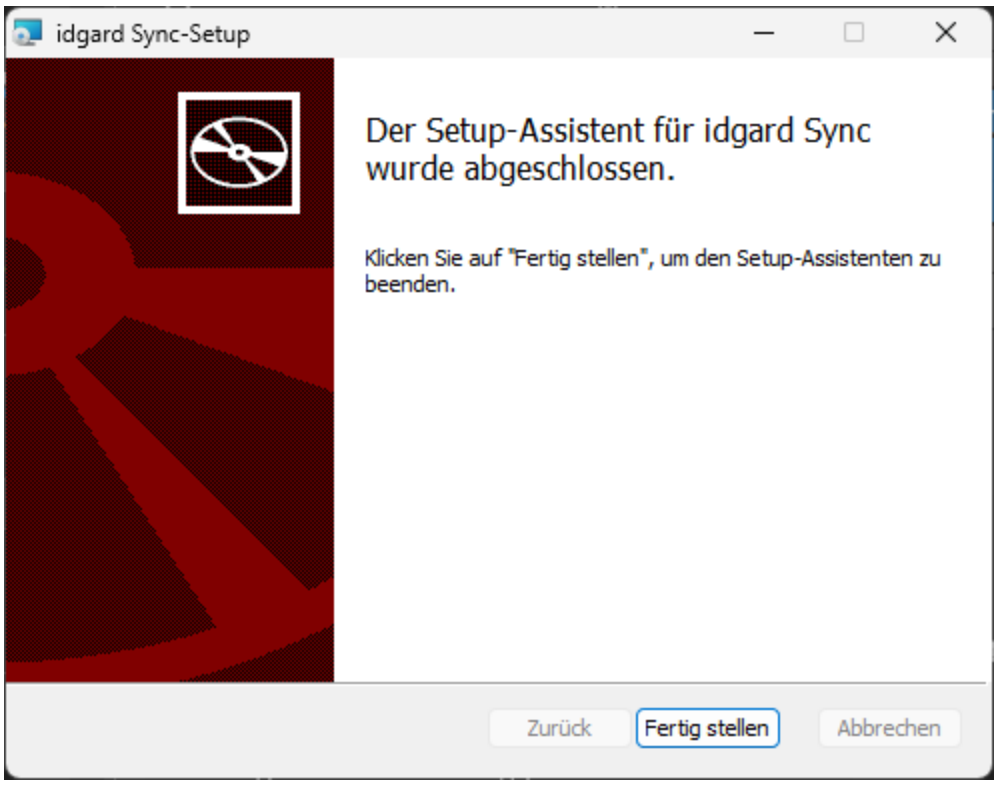


Feature Tree



Bestätigung Dialog

> UAC – User Access Control um für Admin-Privilegien zu fragen.



Ende Dialog