One Identity Manager 9.1.1

# API-Entwicklungshandbuch

**Legende**

**WARNUNG: Das Symbol WARNUNG weist auf ein potenzielles Risiko von Körperverletzungen oder Sachschäden hin, für das Sicherheitsvorkehrungen nach Industriestandard empfohlen werden. Dieses Symbol ist oft verbunden mit elektrischen Gefahren bezüglich Hardware.**

**VORSICHT: Das Symbol VORSICHT weist auf eine mögliche Beschädigung von Hardware oder den möglichen Verlust von Daten hin, wenn die Anweisungen nicht befolgt werden.**

One Identity Manager API-Entwicklungshandbuch
Aktualisiert - 28. März 2023, 21:25 Uhr

Die aktuellsten Versionen der Produktdokumentation finden Sie unter One Identity Manager Dokumentation.

# Inhalt

# Über dieses Handbuch

Dieses Handbuch erklärt Ihnen die Funktionsweise des API Servers, wie Sie API-Aufrufe programmieren und One Identity Manager um eigene API-Methoden erweitern.

## Verfügbare Dokumentation

Die Online Version der One Identity Manager Dokumentation finden Sie im Support-Portal unter Online-Dokumentation. Videos mit zusätzlichen Informationen finden Sie unter www.YouTube.com/OneIdentity.

# Grundlagen der API-Entwicklung

HTML-Anwendungen nutzen für die Kommunikation mit der One Identity Manager-API den API-Client. Der API-Client regelt den kompletten Netzwerkzugriff auf den API Server.

Die wichtigsten Bestandteile für die Entwicklung eigener APIs sind:

- API-Projekte: Ein API-Projekt stellt die eigentliche Anwendung dar und stellt API-Methoden bereit. Mit One Identity Manager werden verschiedenste API-Projekte ausgeliefert, beispielsweise das Web Portal (**portal**).
- API-Plugins: Ein API-Plugin dient als Container für benutzerdefinierte Erweiterungen. Mit einem API-Plugin können Sie eigene API-Projekte bereitstellen und/oder bestehenden API-Projekten weitere API-Methoden hinzufügen.
- API-Provider: Ein API-Provider ist eine einzelne Klasse in einer DLL-Datei, die API-Methoden deklariert.

### Detaillierte Informationen zum Thema


- Grundlagen des API Servers auf Seite 6
- Richtlinien und Konventionen auf Seite 8


# Grundlagen des API Servers

In diesem Kapitel finden Sie grundsätzliche Informationen zur Architektur des API Servers, die für die Anprogrammierung mit eigenen API-Methoden wichtig sind.

### Detaillierte Informationen zum Thema


- Allgemeine Informationen zum API Server auf Seite 7
- API Server-Web-Oberfläche aufrufen auf Seite 7
- Verschlüsselung auf Seite 7
- Allgemeine Hinweise zur Programmierung eigener API-Methoden auf Seite 7

# Allgemeine Informationen zum API Server

- Der API Server stellt die API zur Verfügung.
- Der API Server ist mithilfe der Owin-Plattform implementiert (siehe http://owin.org/).
- Bei URLs wird die Groß- und Kleinschreibung beachtet.

# API Server-Web-Oberfläche aufrufen

Von der Web-Oberfläche des API Servers aus können Sie:

- den API Server konfigurieren
- die Swagger-Dokumentation Ihrer API aufrufen
- das Web Portal für Betriebsunterstützung öffnen
- alle installierten Webanwendungen aufrufen

***Um die API Server Web-Oberfläche aufzurufen***

- Rufen Sie die Web-Adressse (URL) Ihres API Servers in einem Browser auf.

# Verschlüsselung

Der API Server legt Daten auf fälschungssichere Weise verschlüsselt auf dem Client ab.

Das Zertifikat wird bei der Installation des API Servers auf dem IIS konfiguriert. Weitere Informationen zur Installation des API Servers finden Sie im *One Identity Manager Installationshandbuch*.

Weitere Informationen zur Konfiguration der Verschlüsselung finden Sie im *One Identity Manager Konfigurationshandbuch für Webanwendungen*.

# Allgemeine Hinweise zur Programmierung eigener API-Methoden

- Da der API Server zustandslos ist (stateless), speichern Sie API-Methoden ohne Client-spezifischen Zustand.
  Sie dürfen daher beispielsweise keine globalen Variablen definieren oder am Session-Objekt Zustandsdaten hinterlegen. Beim Neustart des API Server-Prozesses werden diese Werte nicht wiederhergestellt.

- Nach dem Aktivieren der Routen dürfen Sie die Definitionsobjekte nicht mehr verändern.
- Verwenden Sie asynchronen Code beim Definieren von API-Methoden. Damit wird die effiziente Nutzung der Server-Ressourcen unterstützt sowie die System-Performance unter Last verbessert. Die Methoden der API und des zugrundeliegenden Objektmodells setzen diese Asynchronität mithilfe des Task-based Asynchronous Pattern (TAP) um. Weitere Informationen zu TAP finden Sie unter https://docs.microsoft.com/de-de/dotnet/standard/asynchronous-programming-patterns/task-based-asynchronous-pattern-tap.
- Verwenden Sie beim Definieren von API-Methoden NICHT die Methode **HttpContext.Current**. Die aktuelle HTTP-Anforderung können Sie mit der statischen Methode **QBM.CompositionApi.OwinRequestScopeContext.Context.Current** abfragen.
- Verwenden Sie beim Definieren von API-Methoden, die Daten verändern, NICHT die Methode **GET**.

# Richtlinien und Konventionen

In diesem Kapitel finden Sie allgemeine Richtlinien und Konvention, die Sie beim Erstellen einer API beachten müssen.

### Detaillierte Informationen zum Thema

# Abarbeitung einer Anfrage an den API Server

In diesem Kapitel finden Sie Informationen zur Abarbeitung einer Anfrage an den API Server.

## Authentifizierung

Bei einer Anfrage an den API Server wird geprüft, ob in der Sitzung für das jeweilige API-Projekt die primäre und gegebenenfalls sekundäre Anmeldung erfolgt ist (siehe Authentifizierung auf Seite 10).

> HINWEIS: Diese Prüfung erfolgt nicht, wenn die API-Methode der Anfrage als **AllowUnauthenticated** markiert ist.

Für die Zuordnung der aktuellen Sitzung wird der vom Client übermittelte Cookie **imx-session-<Name des API-Projektes>** ausgewertet.

Wenn ein Cookie übermittelt wird, der aber keiner im aktuellen Prozess aktiven Sitzung zugeordnet werden kann, wird mit dem im Cookie enthaltenen Sicherheitstoken eine neue Sitzung hergestellt (siehe Sitzungsstatus und Sicherheitstoken auf Seite 12).

Liegt keine primäre Anmeldung vor, versucht der API Server über eines der aktivierten Single-Sign-On-Authentifizierungsmodule eine Datenbankverbindung herzustellen.

Kann die Anmeldung auch dann nicht hergestellt werden, wird die Bearbeitung der Anforderung abgebrochen und der HTTP-Fehlercode **500** wird an den Client übermittelt (siehe Antwortcodes auf Seite 21).

## Autorisierung des Methodenzugriffs

Der API Server prüft, ob der aktuell angemeldete Benutzer berechtigt ist, die Methode auszuführen. Verfügt der Benutzer nicht über die benötigten Berechtigungen, wird die Bearbeitung abgebrochen und der HTTP-Fehlercode **500** wird an den Client übermittelt (siehe Antwortcodes auf Seite 21).

## Validierung der Anfrage

Der API Server ruft die an der API-Methode hinterlegten Validatoren nacheinander auf. Wenn ein Fehler zurückgemeldet wird, wird die Bearbeitung abgebrochen und der HTTP-Fehlercode **400** wird an den Client übermittelt (siehe Antwortcodes auf Seite 21).

## Abarbeitung der Anfrage (für Entity-Methoden)

- GET (Laden einer Entity)
  - Ermittlung der WHERE-Klausel mit internen und externen Filtern
  - Laden der Daten aus der Datenbank
  - Anreichern einer Entity mit berechneten Spalten

- Eine Entity im delayed-logic-Modus kann mit einer POST-Anfrage verändert oder mit einer DELETE-Anfrage gelöscht werden. Eine Entity in diesem Modus ist zustandslos (stateless) und belegt auf dem Server nach der Abarbeitung einer Anfrage keine Ressourcen mehr.
  Unterstützte HTTP-Methoden:
  - GET (Lesen einer Entity)
  - POST (Ändern einer Entity)
  - DELETE (Löschen einer Entity)
- Eine interaktive Entity muss einmalig mit einer PUT-Anfrage erzeugt werden, und erhält danach eine eigene ID. Benutzen Sie bei nachfolgenden Anforderungen (POST oder DELETE) diese ID.

  Unterstützte HTTP-Methoden:
  - GET (Laden einer Entity)
  - PUT (Erzeugen einer interaktiven Entity)
  - POST (Ändern einer interaktiven Entity)
  - DELETE (Löschen einer interaktiven Entity)

# Authentifizierung

Die Authentifizierung von Benutzern am API Server erfolgt pro API-Projekt.

Die Ausführung einer API-Methode erfordert die vorherige Authentifizierung an einem API-Projekt. Ist die API-Methode als `AllowUnauthenticated` markiert (Beispiele finden Sie im SDK), ist keine Authentifizierung nötig.

Die Authentifizierung erfolgt in zwei Schritten:

1. Erforderliche primäre Authentifizierung: Standard-Authentifizierung über ein Authentifizierungsmodul
2. Optionale sekundäre Authentifizierung: Multifaktor-Authentifizierung (über OneLogin)

Weitere Informationen zur Konfiguration der Authentifizierung finden Sie im *One Identity Manager Konfigurationshandbuch für Webanwendungen*.

### Detaillierte Informationen zum Thema

- Authentifizierung (primär) auf Seite 11
- Abmeldung auf Seite 12

### Verwandte Themen

- Abarbeitung einer Anfrage an den API Server auf Seite 9

# Authentifizierung konfigurieren

Sie können festlegen, wie sich Benutzer an Ihrer API authentifizieren. Sie konfigurieren die Authentifizierung am API-Projekt.

***Um die Authentifizierung zu konfigurieren***

1. Bearbeiten Sie Ihr API-Projekt.
2. Erstellen Sie die Klasse **SessionAuthDbConfig** und geben Sie dabei folgende Eigenschaften an:
    a. **Product**: Legen Sie die Anwendung fest, deren Authentifizierungsmodule Sie verwenden möchten (beispielsweise **WebDesigner** oder **Manager**).
    b. **SsoAuthentifiers**: Legen Sie die Single-Sign-on-Authentifizierungsmodule fest, die verwendet werden sollen.
    c. **ExcludedAuthentifiers**: Legen Sie Authentifizierungsmodule fest, die nicht verwendet werden sollen.

# Authentifizierung (primär)

Die primäre Authentifizierung am API-Projekt wird mithilfe der API-Methode **imx/login/<Name des API-Projektes>** ermöglicht.

Senden Sie dazu eine Anfrage mit der HTTP-Methode **POST** mit folgendem Inhalt:

```
{ "Module": "RoleBasedPerson", "User": "<Benutzername>", "Password":
"<Passwort>" }
```

TIPP: Beispiele dazu finden Sie im SDK.

## Sicherheitsmechanismen

Der API Server verwendet einen Sicherheitsmechanismus, um Cross-Site-Request-Forgery-Angriffe (CSRF) zu unterbinden. Dafür wird bei der Anmeldung ein zufällig generiertes Token in einem Cookie (**XSRF-TOKEN**) an den Client gesendet. Der Client muss danach in jeder Anfrage an den Server den Wert dieses Tokens in einem HTTP-Header (**X-XSRF-TOKEN**) übermitteln. Fehlt dieser Header, wird die Anfrage mit dem Fehlercode **400** beendet.

HINWEIS: Wenn eine API-Anfrage mit einem Fehler und Hinweis auf einen falschen CSRF-Schutz-Cookie abgebrochen wird, prüfen Sie, ob Ihr Browser die vom Browser gesendeten Cookies akzeptiert.

TIPP: Sie können den Namen und den Pfad des Cookies und den Namen des HTTP-Headers über das Administrationsportal anpassen. Verwenden Sie dazu die Konfigurationsschlüssel **Name des Cookies, der das vom Server ausgestellte CSRF-Schutz-Token enthält** (**XsrfProtectionCookieName**) und **Pfad für den CSRF-Schutz-Cookie** (**XsrfProtectionCookiePath**).

Sie können zudem den CSRF-Schutz im Administrationsportal deaktivieren (Konfigurationsschlüssel **CSRF-Schutz-Token global deaktivieren** (**XsrfProtectionDisabled**)). One Identity empfiehlt jedoch, dies nicht zu tun.

Wie Sie Konfigurationsschlüssel bearbeiten, erfahren Sie im *One Identity Manager Konfigurationshandbuch für Webanwendungen*.

## Abmeldung

Die Abmeldung vom API-Projekt wird mithilfe der API-Methode **imx/logout/<Name des API-Projektes>** ermöglicht.

Senden Sie dazu eine Anfrage mit der HTTP-Methode **POST** ohne Inhalt.

# Sitzungsstatus und Sicherheitstoken

Der Status der Sitzung wird in einem Cookie gespeichert. Dieses Cookie enthält ein verschlüsseltes Sicherheitstoken, mit dem die Anmeldung am API Server auch wiederhergestellt werden kann, wenn der API Server zwischenzeitlich neu gestartet wird. Das Sicherheitstoken wird mit dem bei der Installation ausgewählten Zertifikat kryptographisch signiert.

HINWEIS: Startet ein am API Server angemeldeter Benutzer den verwendeten Browser neu, werden das Cookie und die darin enthaltenen Sitzungsinformationen zurückgesetzt.

### Verwandte Themen

- Sitzungsstatus abfragen auf Seite 12

## Sitzungsstatus abfragen

Mithilfe der API-Methode **imx/sessions/<Name des API-Projektes>** können Sie den Status der Sitzung abfragen. In der Antwort erhalten Sie folgende Informationen:

- Zulässige Authentifizierungsmodule und zugehörige Parameter das jeweilige API-Projekt
- Art der sekundären Anmeldung

# API-Methoden

Sie können die folgenden Arten von API-Methoden definieren.

- Entity-Methoden
- Benutzerdefinierte Methoden
- SQL-Methoden

### Detaillierte Informationen zum Thema


- Entity-Methoden auf Seite 13
- Benutzerdefinierte Methoden auf Seite 19
- SQL-Methoden auf Seite 19


# Entity-Methoden

Entity-Methoden arbeiten mit kleinen Teilen des Objektmodells, um Daten aus der Datenbank zu lesen beziehungsweise in diese zu schreiben. Wenn Sie eine Entity-Methode erstellen müssen Sie nur Tabellen- und Spaltennamen sowie gegebenenfalls eine Filterbedingung (WHERE-Klausel) angeben. Die interne Abarbeitung wird durch den API Server übernommen. Das Schema der Eingabe- und Ausgabedaten ist ebenfalls fest vorgegeben.

Beispiele zur Definition von Entity-Methoden finden Sie im SDK unter `Sdk01_Basics\01-BasicQueryMethod.cs`.

### Detaillierte Informationen zum Thema


- Ergebnisse einschränken auf Seite 13
- Sortierung auf Seite 14
- Filterung auf Seite 14
- Gruppierung auf Seite 15
- Hierarchische Datenstrukturen auf Seite 16
- Abfrage zusätzlicher Parameter auf Seite 17
- Typsichere Klassen


## Ergebnisse einschränken

HINWEIS: Entity-basierte Methoden arbeiten standardmäßig mit einer Limitierung, um unabsichtliches Laden extrem großer Datenmengen zu vermeiden.

Mithilfe der folgenden Abfrage-Parameter können Sie die Anzahl der zurückgegebenen Datensätze begrenzen, um mehrere Datensätze in aufeinanderfolgenden Antworten zu erhalten:

| Abfrage-Parameter | Standardwert | Beschreibung |
|---|---|---|
| PageSize | 20 | Legen Sie fest, wie viele Datensätze in der Antwort maximal enthalten sein dürfen.<br>Wenn Sie nur die Gesamtanzahl ermitteln, aber keine einzelnen Datensätze erhalten möchten, verwenden Sie den Wert **-1**. |
| StartIndex | 0 | Legen Sie fest, ab welchem Datensatz die Ergebnisse in der Antwort zurückgegeben werden.<br>Dieser Parameter ist nullbasiert (das erste Element wird mit dem Wert **0** angesprochen). |

**Beispiel**

Die folgende Anfrage gibt 50 Identitäten zurück und beginnt dabei bei der 101. Identität:

```
https://<Host-Name>/ApiServer/portal/person?PageSize=50&StartIndex=100
```

## Sortierung

Mithilfe des Abfrage-Parameters **OrderBy** können Sie die Ergebnisse einer zurückgegebenen Antwort sortieren. Mit diesem Parameter können Sie nach den Spaltennamen der zugrundeliegenden Datenbanktabelle sortieren.

**Beispiele**

Die folgende Anfrage gibt Identitäten aufsteigend sortiert nach Vorname zurück:

```
https://<Host-Name>/ApiServer/portal/person?OrderBy=FirstName
```

Die folgende Anfrage gibt Identitäten absteigend sortiert nach Vorname zurück:

```
https://<Host-Name>/ApiServer/portal/person?OrderBy=FirstName%20DESC
```

## Filterung

Mithilfe des Abfrage-Parameters **filter** können Sie die Ergebnisse einer zurückgegebenen Antwort filtern. Ein solcher Filter besteht aus einem JSON-formatierten String, der die folgenden Filterbedingungen enthalten muss:

- **ColumnName**: Name der Spalte, auf die gefiltert werden soll
- **CompareOp**: der Vergleichsoperator, der den Inhalt der gewählten Spalte mit einem Sollwert vergleicht

  Die folgenden Vergleichsoperatoren sind zulässig:

  - **Equal**: Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte mit dem Vergleichswert übereinstimmen.
  - **NotEqual**: Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte NICHT mit dem Vergleichswert übereinstimmen.
  - **LowerThan**: Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte kleiner sind als der Vergleichswert.
  - **LowerOrEqual**: Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte kleiner oder gleich dem Vergleichswert sind.
  - **GreaterOrEqual**: Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte größer oder gleich dem Vergleichswert sind.
  - **Like**: Erfordert die Verwendung eines Prozentzeichens (%) als Platzhalterzeichen. Sie können bis zu zwei Prozentzeichen in dem Wert angeben. Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte mit dem Pattern-Vergleichswert übereinstimmen.
  - **NotLike**: Erfordert die Verwendung eines Prozentzeichens (%) als Platzhalterzeichen. Sie können bis zu zwei Prozentzeichen in dem Wert angeben. Die Ergebnisse beinhalten nur Datensätze, bei denen die Daten in der Spalte NICHT mit dem Pattern-Vergleichswert übereinstimmen.
  - **BitsSet**: Der Wert wird mit einem Vergleichswert über den logischen Operator & verglichen. Das Ergebnis muss ungleich 0 sein.
  - **BitsNotSet**: Der Wert wird Wert mit einem Vergleichswert über den logischen Operator & verglichen. Das Ergebnis muss gleich 0 sein.
- **Value1**: Vergleichswert der mit dem Inhalt der Spalte verglichen werden soll
- **Value2**: Wird dieser zweite Vergleichswert übergeben, so wird der Wert für **CompareOp** ignoriert und alle Werte, die größer gleich **Value1** und kleiner gleich **Value2** sind, werden ermittelt.

**Beispiel**

Die folgende Anfrage gibt aller Identitäten mit dem Nachnamen "Smith" zurück:

```
https://<Host-Name>/ApiServer/portal/person/all?filter=[{ColumnName:
'LastName', CompareOp: 'Equal', Value1: 'Smith'}]
```

## Gruppierung

Mithilfe des Pfad-Parameters **group** können Sie die Ergebnisse einer zurückgegebenen Antwort gruppieren. Dabei können Sie mithilfe des Abfrage-Parameters **by** angeben, nach welchem Attribut gruppiert werden soll. Zusätzlich können Sie mithilfe des Abfrage-

Parameters **withcount** festlegen (Werte: **true** oder **false**), ob die Anzahl der Objekte für jede Gruppe berechnet werden soll. Dies kann die Ausführungszeit verlängern.

HINWEIS: Die API-Methode muss Gruppierungen unterstützen (mithilfe des Parameters **EnableGrouping**).

Das Ergebnis der Abfrage enthält eine Filterbedingung, die Sie als URL-Parameter als Filter übergeben können.

**Beispiel**

Die folgende Anfrage ermittelt die Menge aller Identitäten gruppiert nach primärem Standort:

```
https://<Host-Name>/ApiServer/portal/person/all/group?by=UID_
Locality&withcount=true
```

Antwort:

```
{
    {
        "Display": "(No value: Primary location)",
        "Filters": [
            {
                "ColumnName": "UID_Locality",
                "CompareOp": 0
            }
        ],
        "Count": 42
    },
    {
        "Display": "Berlin",
        "Filters": [
            {
                "ColumnName": "UID_Locality",
                "CompareOp": 0,
                "Value1": "c644f672-566b-4ab0-bac0-2ad07b6cf457"
            }
        ],
        "Count": 12
    }
}
```

## Hierarchische Datenstrukturen

Einige Tabellen des Datenmodells sind als Baumstruktur definiert (beispielsweise **Department**). Daten aus solchen Tabellen können aus einer bestimmten Hierarchie-Ebene geladen werden.

Mithilfe des Abfrage-Parameters **parentKey** des übergeordneten Objekts können Sie die Hierarchie-Ebene bestimmen.

**Beispiel**

Die folgende Anfrage ermittelt die Servicekategorien direkt unterhalb der Servicekategorie **Access Lifecycle**:

```
https://<Host-Name>/ApiServer/portal/servicecategories?parentKey=QER-f33d9f6ec3e744a3ab69a474c10f6ff4
```

Die folgende Anfrage ermittelt die Servicekategorien, die keine übergeordnete Servicekategorie haben:

```
https://<Host-Name>/ApiServer/portal/servicecategories?parentKey=
```

Die folgende Anfrage ermittelt alle Servicekategorien unabhängig von ihrer Hierarchie:

```
https://<Host-Name>/ApiServer/portal/servicecategories
```

Mithilfe des Pfad-Parameters **noRecursive** können Sie festlegen, ob die Daten als flache Liste abgefragt wird (Werte: **true** oder **false**).

**Beispiel**

```
https://<Host-Name>/ApiServer/portal/servicecategories?noRecursive=true
```

## Abfrage zusätzlicher Parameter

Mithilfe des Abfrage-Parameters **withProperties** können Sie festlegen, welche zusätzlichen Informationen aus bestimmten Tabellenspalten in der zurückgegebenen Antwort enthalten sein sollen.

HINWEIS: Um Tabellenspalten für diese Abfragen freizugeben, aktivieren Sie im Designer für die entsprechenden Spalten in den Spalteneigenschaften die Option **Anzeige in Assistenten**.

TIPP: Sie können die Namen mehrerer Spalten kommasepariert angeben.

**Beispiel**

Die folgende Anfrage ermittelt die Menge aller Identitäten und gibt zusätzlich deren bevorzugten Namen und Titel zurück:

```
https://<Host-Name>/ApiServer/portal/person/all?withProperties=PreferredName,Title
```

Antwort:

```
{
    "TotalCount": 105950,
    "TableName": "Person",
    "Entities": [
        {
            "Display": "100, User (USER1)",
            "LongDisplay": "100, User (USER1)",
            "Keys": [
                "bbf3f8e6-b719-4ec7-be35-cbd6383ef370"
            ],
            "Columns": {
                "DefaultEmailAddress": {
                    "Value": "USER1@qs.ber",
                    "IsReadOnly": true
                },
                "IdentityType": {
                    "Value": "Primary",
                    "IsReadOnly": true,
                    "DisplayValue": "Primary identity"
                },
                "PreferredName": {
                    "Value": "Johnny",
                    "IsReadOnly": true
                },
                "Title": {
                    "Value": "Dr.",
                    "IsReadOnly": true
                },
                "XObjectKey": {
                    "Value": "<Key><T>Person</T><P>bbf3f8e6-b719-4ec7-be35-
cbd6383ef370</P></Key>",
                    "IsReadOnly": true
                }
            }
}
```


## Typsichere Klassen

Verwenden Sie die typsichere Klassen, um das Datenbankmodell typsicher zu verwenden. Daraus ergeben sich folgende Vorteile:

- Beim Kompilieren der Skripte wird geprüft, ob die verwendeten Klassen korrekt sind. Dadurch erkennen Sie bereits sehr früh Tippfehler in Tabellen- und Spaltennamen.
- Die Entwicklungsumgebung kann eine automatische Vervollständigung anbieten.
- Der Datentyp der Spalten ist bekannt, wodurch Typkonvertierungsfehler vermieden werden.

***Um typsichere Klassen zu verwenden***

1. Bearbeiten Sie das entsprechende API-Plugin (siehe API-Plugins bearbeiten auf Seite 26) und nehmen Sie dabei folgende Aktion vor:
    - Fügen Sie eine Referenz auf die Bibliothek für typsichere Klassen des entsprechenden Datenbankmoduls hinzu (zum Beispiel `AOB.TypedWrappers.dll`).

      Die Klassen für dieses Modul sind daraufhin im Namensraum `<Modulname>.TypedWrappers` verfügbar (zum Beispiel `AOB.TypedWrappers`).

# Benutzerdefinierte Methoden

Benutzerdefinierte Methoden sind Methoden, deren Abarbeitung, Eingabe- und Ausgabedaten Sie im Code vollständig definieren. Dieser Typ bietet daher die größte Flexibilität.

Beispiele zur Definition von benutzerdefinierten Methoden finden Sie im SDK unter `Sdk01_Basics\03-CustomMethod.cs`.

# SQL-Methoden

SQL-Methoden sind Methoden, die Daten aus einer vordefinierten SQL-Abfrage über die API zur Verfügung stellen. Die Parameter einer Anfrage legen Sie als SQL-Parameter an.

Beispiele zur Definition von SQL-Methoden finden Sie im SDK unter `Sdk01_Basics\02-BasicSqlMethod.cs`.

### Detaillierte Informationen zum Thema

- Entity-Methoden auf Seite 13

# HTTP-Methoden

HTTP-Anfragen können die folgenden HTTP-Methoden verwenden:

- **GET**: Diese Methode ruft Daten vom Anwendungsserver ab.
- **PUT**: Diese Methode ändert Daten auf dem Anwendungsserver.
- **POST**: Diese Methode erstellt Daten auf dem Anwendungsserver.
- **DELETE**: Diese Methode löscht Daten auf dem Anwendungsserver.

# Datumsformate

Datumswerte müssen in Anfragen zum Ändern oder Hinzufügen von Objekten im Format ISO 8601 in der lokalen Zeitzone des Clients angegeben werden.

**Beispiel**

```
2016-03-19T13:09:08.123Z
```

### Verwandte Themen

- Parameterformate auf Seite 20

# Parameterformate

HTTP-Anfragen verwenden zwei Arten von Parametern:

- Pfad-Parameter
- Abfrage-Parameter

### Verwandte Themen

- Datumsformate auf Seite 20

# Pfad-Parameter

Pfad-Parameter setzen den URL-Pfad fort. Als Trennzeichen wird hierbei ein Schrägstrich verwendet.

Wenn eine Anfrage Pfad-Parameter verwendet, werden diese im URI-Format für die Anfrage angegeben.

**Beispiel**

```
https://<Host-Name>/ApiServer/imx/sessions/exampleparameter
```

# Abfrage-Parameter

Abfrage-Parameter werden der URL über ein Frage- oder das &-Zeichen angehangen.

Dem ersten Abfrage-Parameter muss ein Fragezeichen vorangestellt werden. Dabei muss folgendes Format verwendet werden:

`?Parametername=Parameterwert` (beispielsweise `?orderBy=LastName`)

Den nachfolgenden Abfrage-Parametern muss ein & vorangestellt werden. Dabei muss folgendes Format verwendet werden:

`&Parametername=Parameterwert` (beispielsweise `?sortOrder=ascending`)

HINWEIS: Unbekannte Abfrage-Parameter werden vom Server mit dem Fehlercode **400** abgelehnt. Das betrifft auch Abfrage-Parameter mit einer fehlerhaften Groß-/Kleinschreibung.

**Beispiel**

`https://<Host-Name>/AppServer/portal/person?orderBy=LastName`

# Antwortformate

Die meisten API-Methoden liefern Ergebnisse im JSON-Format (application/json) zurück. Des Weiteren wird die Rückgabe von Ergebnissen in den Formaten CSV und PDF unterstützt, soweit das Ergebnis der entsprechenden API-Methode als exportierbar deklariert ist (über das Flag **AllowExport**). Grundsätzlich kann eine API-Methode Ergebnisse in beliebigen Formaten zurückgeben, die mit HTTP kompatibel sind.

***Um Ergebnisse im CSV-Format zu erhalten***

- Setzen Sie den **Accept header** in der Anfrage auf **text/csv**.

***Um Ergebnisse im PDF-Format zu erhalten***

- Setzen Sie den **Accept header** in der Anfrage auf **application/pdf**.

  HINWEIS: Um Ergebnisse im PDF-Format zu erhalten, muss in Ihrem System das Modul **RPS** installiert sein.

### Verwandte Themen

- Antwortcodes auf Seite 21

# Antwortcodes

Antworten, die von der REST-API versendet werden, verwenden die nachfolgenden Codes. Wenn Anfragen fehlschlagen, wird eine erklärende Fehlermeldung angezeigt.

| Antwortcodes | Beschreibung |
|---|---|
| 200 | Die Anfrage war erfolgreich. |
| 204 | Die Anfrage war erfolgreich. Die Antwort enthält keinen Inhalt. |
| 401 | Der Zugriff ist unautorisiert. Die Sitzung muss zuerst authentifiziert werden. |
| 404 | Die angegebene Ressource konnte nicht gefunden werden. |
| 405 | Die verwendete HTTP-Methode ist für diese Anfrage nicht erlaubt. |
| 500 | Ein Server-Fehler ist aufgetreten. Die Fehlermeldung wird in der Antwort mitgesendet. Aus Sicherheitsgründen ist keine detaillierte Fehlermeldung in der Antwort enthalten. Weitere Informationen können Sie der Anwendungsprotokolldatei auf dem Server entnehmen. |

### Verwandte Themen

- Antwortformate auf Seite 21

# Deadlocks vermeiden

Bei der API-Entwicklung wird viel asynchroner Code mit async/await-Konstrukt geschrieben. Um sogenannte Deadlocks (Verklemmungen) zu verhindern, verwenden Sie für jedes **await**-Schlüsselwort die Methode **ConfigureAwait(false)**.

Weitere Informationen finden Sie unter https://blog.stephencleary.com/2012/07/dont-block-on-async-code.html und https://devblogs.microsoft.com/dotnet/configureawait-faq/.

# Beispiele und Hilfe – Software Development Kit

Um Ihnen die Entwicklung Ihrer API einfacher zu gestalten, stellt Ihnen One Identity ein Software Development Kit (SDK) mit vielen kommentierten Code-Beispielen zur Verfügung.

Das SDK finden Sie unter auf dem Installationsmedium im Verzeichnis `QBM\dvd\AddOn\ApiSamples`.

# Eigene APIs implementieren

Um eigene APIs zu implementieren, können Sie API-Plugins erstellen.

Der API Server lädt alle DLLs, auf die das Namensschema `*.CompositionApi.Server.PlugIn.dll` passen, und stellt die darin enthaltenen API-Definitionen zur Verfügung.

Um Ihre eigene APIs zu implementieren, stehen Ihnen folgende Möglichkeiten zur Verfügung:

- Sie können eine API zu einem One Identity-API-Projekt hinzufügen (siehe APIs zu One Identity-API-Projekten hinzufügen).
- Sie können ein eigenes API-Projekt erstellen (API-Projekte erstellen auf Seite 29).
- Sie können eine API zu einem Ihrer eigenen bestehenden API-Pojekte hinzufügen (siehe APIs zu eigenen API-Projekte hinzufügen auf Seite 29).

### Detaillierte Informationen zum Thema

- API-Plugins verwalten auf Seite 24
- APIs zu One Identity-API-Projekten hinzufügen auf Seite 28
- API-Projekte erstellen auf Seite 29
- APIs zu eigenen API-Projekte hinzufügen auf Seite 29

# API-Plugins verwalten

Mithilfe von API-Plugins können Sie eigens entwickelte APIs und API-Projekte implementieren und verwenden.

Voraussetzungen:

- Sie verwenden eine Versionsverwaltung (beispielsweise Git).
- Sie verwenden eine integrierte Entwicklungsumgebung (IDE).

### Detaillierte Informationen zum Thema

- API-Plugins erstellen
- API-Plugins bearbeiten
- TypeScript API-Clients kompilieren auf Seite 26

# API-Plugins erstellen

Um eigens entwickelte APIs und API-Projekte zu implementieren, können Sie API-Plugins erstellen.

***Um ein API-Plugin zu erstellen***

1. Starten Sie Ihre IDE (beispielsweise Visual Studio).
2. Erstellen Sie ein neues .NET Framework 4.8-Projekt mit einem Namen nach dem folgenden Format: `<Name des Projekts>.CompositionApi.Server.Plugin`.
3. Fügen Sie Referenzen auf die folgenden DLL-Dateien aus dem One Identity Manager-Installationsverzeichnis hinzu:
   - `QBM.CompositionApi.Server.dll`
   - `VI.Base.dll`
   - `VI.DB.dll`
4. Erstellen Sie den API-Code.
5. Kompilieren Sie die DLL-Datei in Ihrer IDE.
6. Kopieren Sie die DLL-Datei in den Unterordner `bin` Ihrer IIS-Installation.
7. Importieren Sie die DLL-Datei mithilfe des Software Loaders in Ihre One Identity Manager-Datenbank und weisen Sie sie der Maschinenrolle **Business API Server** zu. Weitere Informationen zum Importieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
8. Kopieren Sie die DLL-Datei in das One Identity Manager-Installationsverzeichnis.
9. Importieren Sie die DLL-Datei mithilfe des Software Loaders in Ihre One Identity Manager-Datenbank und weisen Sie sie der Maschinenrolle **Development and Testing** zu. Weitere Informationen zum Importieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
10. Starten Sie den API Server neu und stellen Sie sicher, dass die Datei `<Name des Projekts>.CompositionApi.Server.Plugin` im Ordner `bin` des API Server-Installationsverzeichnisses vorhanden ist.
11. Kompilieren Sie den entsprechenden TypeScript API-Client (siehe TypeScript API-Clients kompilieren auf Seite 26).

# API-Plugins bearbeiten

Sie können bestehende API-Plugins bearbeiten.

***Um ein bestehendes API-Plugin zu bearbeiten***

1. Starten Sie Ihre IDE (beispielsweise Visual Studio).
2. Öffnen Sie ein bestehendes .NET Framework 4.8-Projekt.
3. Bearbeiten Sie den API-Code.
4. Kompilieren Sie die DLL-Datei in Ihrer IDE.
5. Kopieren Sie die DLL-Datei in den Unterordner `bin` Ihrer IIS-Installation.
6. Importieren Sie die DLL-Datei mithilfe des Software Loaders in Ihre One Identity Manager-Datenbank. Weitere Informationen zum Importieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
7. Kopieren Sie die DLL-Datei in das One Identity Manager-Installationsverzeichnis.
8. Importieren Sie die DLL-Datei mithilfe des Software Loaders in Ihre One Identity Manager-Datenbank. Weitere Informationen zum Importieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
9. Starten Sie den API Server neu und stellen Sie sicher, dass die Datei `<Name des Projekts>.CompositionApi.Server.Plugin` im Ordner `bin` des API Server-Installationsverzeichnisses vorhanden ist.
10. Kompilieren Sie den entsprechenden TypeScript API-Client (siehe TypeScript API-Clients kompilieren auf Seite 26).

# TypeScript API-Clients kompilieren

Nachdem Sie ein API-Plugin erstellt haben, müssen Sie einen entsprechenden TypeScript API-Client kompilieren.

***Um einen TypeScript API-Client zu kompilieren***

1. Öffnen Sie ein Kommandozeilenprogramm (zum Beispiel Windows Powershell).
2. Im Kommandozeilenprogramm wechseln Sie in das Installationsverzeichnis von One Identity Manager.
3. Führen Sie das ImxClient-Kommando **start-update** aus (siehe start-update auf Seite 45).

**Beispiel**

```
imxclient start-update
```

4. Führen Sie das ImxClient-Kommando **compile-api** aus (siehe compile-api auf Seite 33).

   **Beispiel**

   ```
   imxclient compile-api /copyapi imx-api-ccc.tgz /packagename imx-api-ccc
   ```

   Das Dialogfenster zum Auswählen der Datenbankverbindung öffnet sich.

5. Im Dialogfenster nehmen Sie eine der folgenden Aktionen vor:
   - Um eine bestehende Verbindung zur One Identity Manager-Datenbank zu verwenden, wählen Sie aus der Auswahlliste **Datenbankverbindung auswählen** die entsprechende Verbindung aus.

     - ODER -

   - Um eine neue Verbindung zur One Identity Manager-Datenbank zu verwenden, klicken Sie **Neue Verbindung erstellen** und geben Sie eine neue Verbindung an.
6. Unter **Authentifizierungsverfahren** geben Sie das Verfahren und die Anmeldedaten an, mit denen Sie sich an der Datenbank anmelden möchten.
7. Klicken Sie **Anmelden**.
8. Importieren Sie das npm-Paket `imx-api-ccc` in Ihre TypeScript-Anwendung.

   TIPP: (Optional) Falls Sie einen anderen Paketnamen als `imx-api-ccc` verwenden möchten, erweitern Sie die Datei `remove-local-packages.js` und fügen Sie eine Zeile für das Paket in die Liste ein. Dies sorgt dafür, dass Ihr Paket nicht in das Package Locking aufgenommen wird und immer aus der lokalen Quelle aktualisiert wird.

## Verwandte Themen

- compile-api auf Seite 33

# APIs zu One Identity-API-Projekten hinzufügen

Um One Identity-HTML-Anwendungen um angepasste Funktionen zu ergänzen, können Sie eigene APIs zu One Identity-API-Projekten hinzufügen. Dazu erstellen Sie ein API-Plugin, definieren darin die API und weisen diesem API-Plugin das entsprechende One Identity-API-Projekt zu.

***Um eine eigene API zu einem One Identity-API-Projekt hinzuzufügen***

1. Erstellen oder bearbeiten Sie ein API-Plugin (siehe API-Plugins erstellen auf Seite 25 oder API-Plugins bearbeiten auf Seite 26) und nehmen Sie dabei folgende Aktionen vor:
    a. Erstellen Sie in dem API-Plugin-Projekt eine neue Klasse. Diese Klasse stellt den sogenannten API-Provider dar.
    b. Deklarieren Sie die Klasse mit dem Interface, das zu dem API-Projekt gehört, zu dem Sie Ihre API hinzufügen möchten.

       Die folgenden One Identity-API-Projekte können ergänzt werden:

       **Tabelle 1: Ausgelieferte API-Projekte**

| Name der HTML-Anwendung | Name des API-Projekts | Zu implementierendes Interface |
|---|---|---|
| Web Portal | portal | `IApiProviderFor<QER.CompositionApi.Portal.PortalApiProject>` |
| Web Portal für Betriebsunterstützung | opsupport | `IApiProviderFor<QBM.CompositionApi.Operations.OperationsApiProject>` |
| Administrationsportal | admin | `IApiProviderFor<QBM.CompositionApi.AdminApi.AdminApiProject>` |
| Kennwortrücksetzungsportal | passwordreset | `IApiProviderFor<QER.CompositionApi.Password.PasswordPortalApiProject>` |

    c. Implementieren Sie die Methode **Build** des Interfaces **IApiProviderFor** mit

der gewünschten API-Funktionalität.

**Beispiel**

```
    public class ExampleApi : IApiProviderFor<QER.CompositionApi.Portal.PortalApiProject>

    {
        public void Build(IApiBuilder builder)
        {
            builder.AddMethod(Method.Define("example")
                .AllowUnauthenticated()
                .HandleGet(qr => new DataObject { Message = "Hello world!" }));
        }
    }
```

# API-Projekte erstellen

Um One Identity-HTML-Anwendungen um angepasste Funktionen zu ergänzen, können Sie eigene API-Projekte erstellen. Dazu kopieren Sie das Beispiel-API-Projekt **CustomApiProject** (siehe Beispiele und Hilfe – Software Development Kit auf Seite 23), passen es entsprechend an und weisen es einem API-Plugin zu.

***Um ein eigenes API-Projekt zu erstellen***

1. Kopieren Sie das Beispiel-API-Projekt **CustomApiProject**.
2. Passen Sie das kopierte API-Projekt entsprechend an.
3. Erstellen Sie ein API-Plugin (siehe API-Plugins erstellen auf Seite 25) und nehmen Sie dabei folgende Aktion vor:
    - Erstellen Sie in dem API-Plugin-Projekt eine neue Klasse, die das Interface **IApiProviderFor<Name Ihres API-Projekts>** implementiert. Diese Klasse stellt den sogenannten API-Provider dar.

# APIs zu eigenen API-Projekte hinzufügen

Sie können weitere APIs zu eigenen API-Projekten hinzufügen.

***Um eine API zu Ihrem API-Projekt hinzuzufügen***

1. Bearbeiten Sie das API-Plugin (siehe API-Plugins bearbeiten auf Seite 26), das zu dem API-Projekt gehört, und nehmen Sie dabei folgende Aktion vor:
    - Erstellen Sie in dem API-Plugin-Projekt eine neue Klasse, die das Interface **IApiProviderFor<Name Ihres API-Projekts>** implementiert. Diese Klasse stellt den sogenannten API-Provider dar.

# ImxClient-Kommandozeilenprogramm

Mithilfe des ImxClient-Kommandozeilenprogramms können Sie verschiedene Funktionen zum Verwalten des API Servers und von Dateien in der Kommandozeile ausführen.

### Detaillierte Informationen zum Thema

- ImxClient-Kommandozeilenprogramm starten auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# ImxClient-Kommandozeilenprogramm starten

Sie können das ImxClient-Kommandozeilenprogramm jederzeit über ein beliebiges Kommandozeilenprogramm starten.

***Um das ImxClient-Kommandozeilenprogramm zu starten***

1. Öffnen Sie ein Kommandozeilenprogramm (zum Beispiel Windows Powershell).
2. Im Kommandozeilenprogramm wechseln Sie in das Installationsverzeichnis von One Identity Manager.
3. Starten Sie die Anwendung `ImxClient.exe`.

# ImxClient-Kommando-Übersicht

In den nachfolgenden Kapiteln erhalten Sie eine Liste aller ImxClient-Kommandos, die Sie ausführen können.

### Detaillierte Informationen zum Thema

- check-translations auf Seite 32
- compile-api auf Seite 33
- compile-app auf Seite 34
- connect auf Seite 35
- edit-config auf Seite 36

# check-translations

Sucht nach Beschriftungen (mehrsprachige Texte) mit fehlenden Übersetzungen in einem bestimmten Ordner und dessen Unterordnern.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/path <Pfad zum Ordner>`: Legt fest, welcher Ordner geprüft werden soll.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver

aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# compile-api

Kompiliert die API und speichert das Ergebnis in der Datenbank.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `-N`: Verhindert das Speichern in der Datenbank.
- `/modules <Modul1,Modul2,...>`: Legt fest, welche Module aufgenommen werden sollen. Wenn Sie hier nichts angeben, werden alle Module aufgenommen. Geben Sie die Namen der Module kommasepariert ein.
- `/clientmodules <Modul1,Modul2,...>`: Legt fest, für welche Module API-Code generiert werden soll. Wenn Sie hier nichts angeben, wird für alle Module API-Code generiert. Geben Sie die Namen der Module kommasepariert ein.

- `/packagename <Name>`: Legt den API-Client-Paketnamen fest. Der Standardwert ist `imx-api`.
- `/copyapi <Pfad zum Ordner>`: Legt fest, wohin die Datei `imx-api.tgz` kopiert werden soll.
- `/nowarn <Fehler1,Fehler2,...>`: Legt fest, welche Fehler beim Kompilieren ignoriert werden sollen. Geben Sie die Codes der Warnungen kommasepariert ein.
- `/warnaserror <Fehler1,Fehler2,...>`: Legt fest, welche Warnungen beim Kompilieren als Fehler angezeigt werden sollen. Geben Sie die Codes der Warnungen kommasepariert ein.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# compile-app

Führt die HTML-Paketkompilierung aus.

Das Kommando führt die folgenden Schritte aus:

1. Ausführen des Kommandos **npm install** im Anwendungsordner
2. Ausführen des Kommandos **npm run build** im Packet-Ordner
3. Erstellen des Ergebnisses im Unterordner `dist`
   Das Ergebnis wird in einer ZIP-Datei in der Datenbank gespeichert.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.

    - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `/workspace <Pfad zum Arbeitsverzeichnis>`: Legt das Arbeitsverzeichnis fest. In diesem Ordner liegt die Anwendung, die kompiliert werden soll. Dieser Ordner enthält normalerweise die Datei `package.json` der Anwendung. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.
- `/app <Name der Anwendungsprojekts>`: Legt fest, welches Anwendungsprojekt kompiliert werden soll. Wenn Sie hier nichts angeben, werden alle Anwendungsprojekte kompiliert.
- `-D`: Führt die Debug-Kompilierung aus.
- `-S`: Überspringt die Ausführung des Kommandos **npm install** im Anwendungsordner.
- `-P`: Verhindert das Bauen von Bibliotheken im Anwendungsordner.
- `/copyto <Dateipfad>`: Speichert das Ergebnis der Kompilierung als ZIP-Dateien in einem Ordner.
- `/exclude <Modulname>`: Lässt Pakete eines Moduls bei der Kompilierung aus (beispielsweise **AOB**).

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# connect

Baut eine Datenbankverbindung auf.

Sollte bereits eine Datenbankverbindung bestehen, wird diese geschlossen und anschließend eine neue aufgebaut.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# edit-config

Konfiguriert einen Trusted Source Key für eine Anwendung.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/path <Pfad zum Ordner>`: Legt fest, welche Konfigurationsdatei geladen werden soll (zum Beispiel die Datei `web.config` einer Webanwendung). Über die Einstellung **BaseURL** dieser Konfigurationsdatei wird festgestellt, für welche Anwendung der Trusted Source Key erstellt werden soll.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.

    - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
    - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `-T`: Konfiguriert einen zufällig generierten Trusted Source Key für die Anwendung.
- `/trustedsourcekey <Trusted Source Key>`: Konfiguriert den angegebenen Trusted Source Key für die Anwendung.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# fetch-files

Lädt alle Dateien einer bestimmten Maschinenrolle aus der Datenbank und speichert sie in einem lokalen Ordner.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/targets <Ziel1;Ziel2;...>`: Legt fest, welche Maschinenrollen verwendet werden sollen.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
    - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
    - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
    - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver

aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`

- `/workspace <Pfad zum Arbeitsverzeichnis>`: Legt das Arbeitsverzeichnis fest, in dem die Dateien abgelegt werden sollen. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# get-apistate

Fragt den Kompilierungsstatus der API in der Datenbank ab.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `/branch <ID des Kompilierungszweiges>`: Fragt den Kompilierungsstatus der API ab, die unter dem angegebenen Kompilierungszweig gespeichert wurde.
- `/htmlapp <Name des HTML-Pakets>`: Liefert Daten für das angegebene HTML-Paket.
- `-D`: Liefert Daten für Debug-Assemblies.
- `-R`: Liefert Daten für Release-Assemblies.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# get-filestate

Vergleicht die lokale Dateistruktur mit der Dateistruktur in der Datenbank.

Mithilfe des Konfigurationsparameters **QBM | ImxClient | get-filestate | NewFilesExcludePatterns** können Sie festlegen, welche Dateien vom Abgleich ausgeschlossen werden sollen. So verhindern Sie eine zu große Last während des Abgleichs. Standardmäßig werden die Ordner `node_modules` und `imx-modules` vom Abgleich ausgeschlossen.
Den Konfigurationsparameter können Sie im Designer anpassen. Verwenden Sie beim Festlegen der Regeln, die hier festgelegten Formate:
https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.filesystemglobbing.matcher
Trennen Sie mehrere Einträge mit dem Zeichen **|**.

HINWEIS: Mit dem Konfigurationsparameter schließen Sie grundsätzlich nur neue Dateien vom Abgleich aus. Bereits in der Datenbank vorhandene Dateien werden nicht berücksichtigt.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/targets <Ziel1;Ziel2;...>`: Legt fest, welche Maschinenrollen verwendet werden sollen.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird

diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.

- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `/workspace <Pfad zum Verzeichnis>`: Legt das Arbeitsverzeichnis fest, in dem die Dateien liegen, die Sie abgleichen möchten. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# help

Zeigt die Liste der zulässigen Befehle an.

### Parameter

Um Hilfe zu bestimmten Befehlen anzuzeigen, übergeben Sie zusätzlich den entsprechenden Befehl als Parameter.

Beispiel: `help fetch-files`

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# inject-package

Installiert Pakete aus einer tgz-Datei in das Verzeichnis `node_modules` des Arbeitsverzeichnisses.

Verwenden Sie dieses Kommando nur für lokale und abhängigkeitslose Pakete, die keine vollständige NPM-Installation erfordern.

### Parameter

Pflichtparameter:

- `/inject <Paket1>,<Paket2>,...`: Legt fest, welche Pakete installiert werden sollen. Geben Sie die Namen der Pakete kommasepariert ein.

Optionale Parameter:

- `/workspace <Pfad zum Arbeitsverzeichnis>`: Legt das Arbeitsverzeichnis fest, in dessen `node_modules`-Verzeichnis die Pakete installiert werden sollen. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# install-apiserver

Installiert einen API Server auf dem lokalen Internet Information Services (IIS).

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/app <Anwendungsname>`: Legt fest, welcher Name für die Anwendung verwendet werden soll (zum Beispiel in der Titelzeile des Browsers).
- `/sessioncert <Zertifikatsfingerabdruck>`: Legt fest, welches (installierte) Zertifikat für die Erzeugung und Überprüfung von Session-Token verwendet werden soll.

  TIPP: Um den Fingerabdruck eines Zertifikats zu erhalten, können Sie beispielsweise die Windows-Funktion **Computerzertifikate verwalten** verwenden und den Fingerabdruck über die Detailinformationen des Zertifikats ermitteln.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.

    - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `-u`: Erlaubt unsichere HTTP-Verbindungen zur API Server-Webseite. Standardmäßig kann die API Server-Webseite nur über eine verschlüsselte Verbindung aufgerufen werden.
- `/site <Seitenname>`: Legt die Website auf dem IIS fest, unter der die Webanwendung installiert werden soll. Wenn Sie hier nichts angeben, wird die Website automatisch ermittelt (normalerweise **Default Web Site**).
- `/searchservice <URL>`: Legt die URL des Anwendungsservers fest, auf dem der Suchdienst gehostet wird, der verwendet werden soll.

  HINWEIS: Wenn Sie die Volltextsuche verwenden möchten, müssen Sie einen Anwendungsserver angeben. Sie können den Anwendungsserver auch zu einem späteren Zeitpunkt in die Konfigurationsdatei eintragen.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# push-files

Speichert Dateien, die Sie lokal geändert haben, wieder in der Datenbank.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Pflichtparameter:

- `/targets <Ziel1;Ziel2;...>`: Legt fest, welche Maschinenrollen verwendet werden sollen.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
    - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.

    - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
    - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `/workspace <Pfad zum Ordner>`: Legt das Arbeitsverzeichnis fest, in dem die Dateien liegen, die geändert wurden und jetzt in der Datenbank gespeichert werden sollen. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.
- `/tag <uid>`: Legt fest, auf welches Änderungskennzeichen die Änderungen gebucht werden. Weitere Informationen zu Änderungskennzeichen finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
- `/add <Datei1;Datei2;...>`: Legt fest, welche neuen Dateien der Datenbank hinzugefügt werden sollen. Verwenden Sie relative Pfade.
- `/del <Datei1;Datei2;...>`: Legt fest, welche Dateien aus der Datenbank gelöscht werden sollen. Verwenden Sie relative Pfade.
- `-C`: Verhindert das Speichern von geänderten Dateien und speichert nur neue Dateien und löscht Dateien in der Datenbank.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

## repl

Startet das ImxClient-Kommandozeilen-Programm im sogenannten REPL-Modus.

In diesem Modus werden in einer Endlosschleife folgende Aktionen ausgeführt:

- Lesen von Kommandos von **stdin**
- Weiterleiten der Befehle an das passende Plugin
- Ausgeben der Ergebnisse der Verarbeitung nach **stdout**

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# run-apiserver

Startet oder stoppt einen "self-hosted" API Server.

Diese Kommando benötigt eine Datenbankverbindung.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.
- `/dialog <Name der Dialog-Authentifikation>`: Legt die Dialog-Authentifikation fest.

Optionale Parameter:

- `/conndialog <Option>`: Legt fest, ob ein Anmeldedialog für die Datenbankverbindung angezeigt werden soll. Die folgenden Optionen sind möglich:
  - `off`: Es wird kein Anmeldedialog angezeigt. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
  - `show`: Ein Anmeldedialog wird angezeigt (auch wenn bereits eine Verbindung zur Datenbank besteht) und die Verbindung wird durch die neue ersetzt.
  - `fallback`: (Standardwert) Wenn eine Verbindung zur Datenbank besteht, wird diese verwendet. Wenn keine Verbindung zur Datenbank besteht, wird versucht eine Verbindung herzustellen.
- `/factory <Name des Zielsystems>`: Legt das Zielsystem für die Verbindung fest. Geben Sie diesen Parameter an, wenn Sie eine Verbindung zum Anwendungsserver aufbauen möchten. Beispiel: `QBM.AppServer.Client.ServiceClientFactory, QBM.AppServer.Client`
- `-S`: Stoppt den API Server.
- `/baseaddress <Basis-URL mit Port-Angabe>`: Legt die Basis-URL und den Port der Webanwendung fest, unter der der API Server Verbindungen entgegennimmt.

  **Beispiel**

  ```
  /baseaddress http://localhost:8184
  ```

- `/baseurl <Basis-URL>`: Legt die Basis-URL der Webanwendung fest.

  **Beispiel**

```
/baseaddress http://localhost
```

- `/plugin <Dateiname1,Dateiname2>`: Lädt zusätzliche Plugins aus den angegebenen Dateien.
- `/htmldir <Verzeichnis>`: Legt fest, aus welchem Verzeichnis zusätzliche HTML-Anwendungsdateien und Plugins geladen werden sollen. Diese Einstellung ist für Entwicklungsszenarien gedacht.

**Beispiel**

```
/htmldir C:example\imxweb\cpl
```

Das Plugin **cpl** wird aus dem Ordner `C:example\imxweb\cpl` geladen statt aus den Standardquellen.

- `-T`: Fragt den Status des laufenden API Servers ab.
- `-B`: Sperrt die Konsole.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# start-update

Prüft, ob Software-Aktualisierungen verfügbar sind. Falls Software-Aktualisierungen gefunden werden, wird der Software-Aktualisierungsprozess gestartet.

### Parameter

Anmeldeparameter:

- `/conn <Name der Datenbankverbindung>`: Legt die Datenbank fest, mit der Sie sich verbinden möchten.

Optionale Parameter:

- `/target <Pfad des zu aktualisierenden Verzeichnisses>`: Legt das Verzeichnis fest, in dem die Software installiert ist, die aktualisiert werden soll. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.

- `-C`: Prüft nur, ob Software-Aktualisierungen verfügbar sind. Der Software-Aktualisierungsprozess wird nicht gestartet.
- `-G`: Blendet die Benutzeroberfläche der Software-Aktualisierung aus.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# workspace-info

Fragt den Zustand des Angular-Arbeitsverzeichnisses ab (vorhandene Anwendungen und letzte Aktualisierung des API Clients).

### Parameter

Optionale Parameter:

- `/workspace`: Legt fest, welches Arbeitsverzeichnis abgefragt werden soll. Wenn Sie hier nichts angeben, wird das aktuelle Verzeichnis verwendet.

### Verwandte Themen

- ImxClient-Kommandozeilenprogramm auf Seite 31
- ImxClient-Kommando-Übersicht auf Seite 31

# Über uns

One Identity Lösungen eliminieren die Komplexität und die zeitaufwendigen Prozesse, die häufig bei der Identity Governance, der Verwaltung privilegierter Konten und dem Zugriffsmanagement aufkommen. Unsere Lösungen fördern die Geschäftsagilität und bieten durch lokale, hybride und Cloud-Umgebungen eine Möglichkeit zur Bewältigung Ihrer Herausforderungen beim Identitäts- und Zugriffsmanagement.

## Kontaktieren Sie uns

Bei Fragen zum Kauf oder anderen Anfragen, wie Lizenzierungen, Support oder Support-Erneuerungen, besuchen Sie https://www.oneidentity.com/company/contact-us.aspx.

## Technische Supportressourcen

Technische Unterstützung steht für Kunden von One Identity mit einem gültigen Wartungsvertrag und Kunden mit Testversionen zur Verfügung. Sie können auf das Support Portal unter https://support.oneidentity.com/ zugreifen.

Das Support Portal bietet Selbsthilfe-Tools, die Sie verwenden können, um Probleme schnell und unabhängig zu lösen, 24 Stunden am Tag, 365 Tage im Jahr. Das Support Portal ermöglicht Ihnen:

- Senden und Verwalten von Serviceanfragen
- Anzeigen von Knowledge-Base-Artikeln
- Anmeldung für Produktbenachrichtigungen
- Herunterladen von Software und technischer Dokumentation
- Anzeigen von Videos unter www.YouTube.com/OneIdentity
- Engagement in der One Identity-Community
- Chat mit Support-Ingenieuren
- Anzeigen von Diensten, die Sie bei Ihrem Produkt unterstützen

# Index

## I



## K



## L



## M



## P



## R



## S



## T



## U



## V

## Z