



One Identity Manager 9.3

HTML5-Entwicklungshandbuch

Copyright 2025 One Identity LLC.

ALLE RECHTE VORBEHALTEN.

Diese Anleitung enthält urheberrechtlich geschützte Informationen. Die in dieser Anleitung beschriebene Software wird unter einer Softwarelizenz oder einer Geheimhaltungsvereinbarung bereitgestellt. Diese Software darf nur in Übereinstimmung mit den Bestimmungen der geltenden Vereinbarung verwendet oder kopiert werden. Kein Teil dieser Anleitung darf ohne die schriftliche Erlaubnis von One Identity LLC in irgendeiner Form oder mit irgendwelchen Mitteln, elektronisch oder mechanisch reproduziert oder übertragen werden, einschließlich Fotokopien und Aufzeichnungen für irgendeinen anderen Zweck als den persönlichen Gebrauch des Erwerbers.

Die Informationen in diesem Dokument werden in Verbindung mit One Identity Produkten bereitgestellt. Durch dieses Dokument oder im Zusammenhang mit dem Verkauf von One Identity LLC Produkten wird keine Lizenz, weder ausdrücklich oder stillschweigend, noch durch Duldung oder anderweitig, an jeglichem geistigen Eigentumsrecht eingeräumt. MIT AUSNAHME DER IN DER LIZENZVEREINBARUNG FÜR DIESES PRODUKT GENANNTEN BEDINGUNGEN ÜBERNIMMT ONE IDENTITY KEINERLEI HAFTUNG UND SCHLIESST JEGLICHE AUSDRÜCKLICHE, IMPLIZIERTE ODER GESETZLICHE GEWÄHRLEISTUNG ODER GARANTIE IN BEZUG AUF IHRE PRODUKTE AUS, EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF DIE IMPLIZITE GEWÄHRLEISTUNG DER ALLGEMEINEN GEBRAUCHSTAUGLICHKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ODER NICHTVERLETZUNG VON RECHTEN. IN KEINEM FALL HAFTET ONE IDENTITY FÜR JEGLICHE DIREKTE, INDIREKTE, FOLGE-, STÖRUNGS-, SPEZIELLE ODER ZUFÄLLIGE SCHÄDEN (EINSCHLIESSLICH, OHNE EINSCHRÄNKUNG, SCHÄDEN FÜR VERLUST VON GEWINNEN, GESCHÄFTSUNTERBRECHUNGEN ODER VERLUST VON INFORMATIONEN), DIE AUS DER NUTZUNG ODER UNMÖGLICHKEIT DER NUTZUNG DIESES DOKUMENTS RESULTIEREN, SELBST WENN ONE IDENTITY AUF DIE MÖGLICHKEIT SOLCHER SCHÄDEN HINGEWIESEN HAT. One Identity übernimmt keinerlei Zusicherungen oder Garantien hinsichtlich der Richtigkeit und Vollständigkeit des Inhalts dieses Dokuments und behält sich das Recht vor, Änderungen an Spezifikationen und Produktbeschreibungen jederzeit ohne vorherige Ankündigung vorzunehmen. One Identity verpflichtet sich nicht, die in diesem Dokument enthaltenen Informationen zu aktualisieren.

Wenn Sie Fragen zu Ihrer potenziellen Nutzung dieses Materials haben, wenden Sie sich an:

One Identity LLC.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Besuchen Sie unsere Website (<http://www.OneIdentity.com>) für regionale und internationale Büro-Adressen.



Patente

One Identity ist stolz auf seine fortschrittliche Technologie. Für dieses Produkt können Patente und anhängige Patente gelten. Für die aktuellsten Informationen über die geltenden Patente für dieses Produkt besuchen Sie bitte unsere Website unter <http://www.OneIdentity.com/legal/patents.aspx>.

Marken

One Identity und das One Identity Logo sind Marken und eingetragene Marken von One Identity LLC. in den USA und anderen Ländern. Für eine vollständige Liste der One Identity Marken, besuchen Sie bitte unsere Website unter www.OneIdentity.com/legal/trademark-information.aspx. Alle anderen Marken sind Eigentum der jeweiligen Besitzer.

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 HTML5-Entwicklungshandbuch
Aktualisiert - 06. Januar 2025, 10:03 Uhr

Die aktuellsten Versionen der Produktdokumentation finden Sie unter [Dokumentation](#).

Inhalt

Über dieses Handbuch	4
Architektur der One Identity Manager-HTML-Anwendungen	5
HTML-Anwendungsentwicklung mit dem GitHub-Repository	6
Architektur des Angular-Workspace	6
Arbeitsumgebung für Nutzung des GitHub-Repositorys einrichten	8
Zugriff auf die API	9
Eigene HTML-Anwendungen erstellen und bearbeiten	10
One Identity Manager-Aktualisierungen übernehmen	10
Bibliotheken anpassen	11
Plugins hinzufügen	11
Fehlende Übersetzungen prüfen	13
HTML-Anwendungen registrieren	13
HTML-Anwendungen kompilieren und bereitstellen	14
Debugging	16
API Server lokal hosten	17
Debugging mit Plugins	17
Über uns	19
Kontaktieren Sie uns	19
Technische Supportressourcen	19

Über dieses Handbuch

Dieses Handbuch zeigt Ihnen, wie sie die HTML-Anwendungen des One Identity Manager im Quelltext ansehen und deren interne Funktionsweise nachvollziehen können. Zusätzlich erfahren Sie, wie Sie eigene HTML-Anwendungen erstellen und implementieren können.

Dafür können Sie bestehende HTML-Anwendungen aus einem GitHub-Repository als Vorlage verwenden (siehe [HTML-Anwendungsentwicklung mit dem GitHub-Repository](#) auf Seite 6).

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.

Architektur der One Identity Manager-HTML- Anwendungen

Die HTML-Anwendungen sind als nodeJS-Anwendungen aufgebaut, die das Framework **Angular** einsetzen. Grundsätzlich werden alle HTML-Anwendungen unterstützt, die sich als nodeJS-Anwendungen kompilieren lassen.

Die HTML-Anwendungen nutzen für die Kommunikation mit der One Identity Manager-API den API-Client. Der API-Client ist eine npm-Bibliothek, die automatisch bei der Kompilierung der API erzeugt und in der Datenbank abgelegt wird. Der API-Client regelt den kompletten Netzwerkzugriff auf den API Server.

Weitere Informationen zur API-Entwicklung finden Sie im *One Identity Manager API-Entwicklungshandbuch*.

HTML-Anwendungsentwicklung mit dem GitHub-Repository

Sie können eigene HTML-Anwendungen entwickeln, indem Sie die Quelltexte der Standard-HTML-Anwendungen als Vorlage verwenden.

Die Quelltexte der Standard-HTML-Anwendungen stehen Ihnen in einem [GitHub-Repository](#) zur Verfügung.

Architektur des Angular-Workspace

Das GitHub-Repository enthält den Quellcode für die in One Identity Manager enthaltenen HTML-Anwendungen. Es ist ein Monorepo, das den Angular-[Workspace](#) enthält, der aus Anwendungen und [Bibliotheken](#) besteht.

Der Angular-Workspace integriert das Werkzeug Nx. Dieses Werkzeug erleichtert den Umgang mit Monorepo-Umgebungen. Nx wird automatisch installiert. One Identity empfiehlt Ihnen Nx zu verwenden, um eine bessere Build-Geschwindigkeit zu erreichen und die Verwaltung von Projektabhängigkeiten zu vereinfachen. Weitere Informationen zu finden Sie [hier](#).

Beachten Sie beim Einsatz von Nx Folgendes:

- Der Nx-Workspace wird in der Datei `nx.json` definiert.
- Jedes Projekt besitzt jeweils eine eigene Deklarationsdatei `project.json`. Nx benutzt diese Dateien, um Abhängigkeiten zwischen den Projekten zu verwalten und die Kompilierungsreihenfolge festzulegen.

TIPP: Um die Abhängigkeiten zwischen Projekten zu visualisieren, verwenden Sie in einem Kommandozeilenprogramm den Befehl `nx graph`.

Jede Angular-Bibliothek und -Anwendung gehört zu einem Ordner im Verzeichnis `projects`. Der Angular-Workspace wird in der Datei `angular.json` definiert.

Tabelle 1: Angular-Bibliotheken

Name	Typ	Abhängigkeiten innerhalb des Workspace
qbm	Angular-Bibliothek	keine
qer	Angular-Bibliothek	qbm
dpr	Angular-Bibliothek	qbm
apc	Angular-Plugin-Bibliothek	qbm, qer
hds	Angular-Plugin-Bibliothek	qbm, qer
olg	Angular-Plugin-Bibliothek	qbm, qer
rmb	Angular-Plugin-Bibliothek	qbm, qer
rps	Angular-Plugin-Bibliothek	qbm, qer
sac	Angular-Plugin-Bibliothek	qbm, qer
qam	Angular-Plugin-Bibliothek	qbm, qer
tsb	Angular-Plugin-Bibliothek	qbm, qer
att	Angular-Plugin-Bibliothek	qbm, qer
rms	Angular-Plugin-Bibliothek	qbm, qer
aad	Angular-Plugin-Bibliothek	qbm, qer, tsb
aob	Angular-Plugin-Bibliothek	qbm, qer
uci	Angular-Plugin-Bibliothek	qbm, qer
cpl	Angular-Plugin-Bibliothek	qbm, qer

Name	Typ	Abhängigkeiten innerhalb des Workspace
o3t	Angular-Plugin-Bibliothek	qbm, qer, tsb
pol	Angular-Plugin-Bibliothek	qbm, qer

Jede Angular-Bibliothek gehört zum gleichnamigen One Identity Manager-Modul. Eine Angular-Bibliothek verhält sich wie eine reguläre Kompilierzeitabhängigkeit. Eine Plugin-Bibliothek wird zur Laufzeit dynamisch geladen. Das ist in den `imx-plugin-config.json`-Dateien der Plugins festgelegt.

Tabelle 2: Angular-Anwendungen

Name	Beschreibung	Projekttyp	Statische Abhängigkeiten
qbm-app-landingpage	API Server-Landing-Page und Server-Verwaltung	Angular-Anwendung	qbm
qer-app-portal	Web Portal	Angular-Anwendung	qbm, qer
qer-app-operationsupport	Web Portal für Betriebsunterstützung	Angular-Anwendung	qbm, qer
qer-app-pwdportal	Kennworrücksetzungsportal	Angular-Anwendung	qbm, qer

Arbeitsumgebung für Nutzung des GitHub-Repositorys einrichten

In diesem Kapitel erfahren Sie, wie Sie Ihre Arbeitsumgebung für die Nutzung des GitHub-Repositorys der Standard-HTML-Anwendungen einrichten, sodass Sie anschließend eigene Webanwendungen entwickeln können.

Voraussetzungen:

- Sie besitzen ein gültiges GitHub-Konto (siehe <https://github.com/>).

Um Ihre Arbeitsumgebung einzurichten

1. Beantragen Sie Zugriff auf das [GitHub-Repository](#).
2. Erstellen Sie einen Fork des entsprechenden Branches des GitHub-Repositorys (siehe <https://docs.github.com/en/get-started/quickstart/fork-a-repo>).

3. Installieren Sie npm (siehe <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>).
4. In Ihrem lokalen Repository, wechseln Sie in das Verzeichnis `imxweb` und führen Sie folgenden Befehl in einem Kommandozeilenprogramm aus:

```
npm install
```

Zugriff auf die API

Für den Zugriff auf die API wird der Typescript-API-Client verwendet.

Für jedes One Identity Manager-Modul, das API-Dienste bereitstellt, existiert eine Angular-Bibliothek `imx-api-<Modulname>.tgz` als NPM-Paket im Ordner `imx-modules` des Repositorys.

HINWEIS: Die API-Client-Bibliotheken sind nicht von Angular abhängig und können somit aus allen Javascript-Programmen verwendet werden.

Die TypeScript-API-Clients bestehen aus den folgenden Teilen:

- **Endpoint-basierte Methoden:**

Die Klasse **V2Client** des API-Clients enthält für jeden API-Endpoint eine Methode. Der Name der Methode wird nach dem folgenden Schema generiert:

```
<URL-Pfad des API-Endpoints>_<HTTP-Methode>
```

Beispiel

Die Methode **GET portal/serviceitems** wird zur Typescript-Methode **portal_serviceitems_get**.

- **Entity-basierte Methoden:**

Die Klasse **TypedClient** des API-Clients enthält für jede entity-basierte API-Methode eine Wrapper-Klasse, die es ermöglicht, Entities zu laden und zu speichern.

Beispiel für Methoden für interaktive Entities

Für die Methode **portal/serviceitems/interactive** gibt es die Eigenschaft **TypedClient.PortalServiceItemsInteractive** vom Typ der Wrapper-Klasse **PortalServiceItemsInteractiveWrapper**.

Je nach Umfang der unterstützten Operationen einer API-Methode gibt es die folgenden Methoden an der Wrapper-Klasse:

- Die Methode **createEntity** wird zum Erstellen einer neuen Entity verwendet.
- Die Methode **Get_byid** wird zum Laden einer interaktiven Entity vom API Server verwendet. Der API Server unterstützt pro Anfrage nur das Laden eines einzelnen Objekts aus der Datenbank als interaktive Entity. Dabei müssen die Primärschlüsselwerte des Objekts als Methodenparameter angegeben werden.
- Die Methoden **Put** und **Post** werden zum Speichern von Entities mit der PUT-Operation verwendet. Diese Methoden dürfen nicht direkt aufgerufen werden, sondern werden über die Methode **commit()** des Interfaces IEntity angesteuert.

Kapselung als Service

Der Zugriff auf die API ist pro Angular-Bibliothek in einem eigenen Angular-Service gekapselt, der sich in eigene Klassen importieren lässt:

- In der Angular-Bibliothek **qbm** heißt der Service **imx_SessionService**.
- In der Angular-Bibliothek **qer** heißt der Service **QerApiService**.
- In allen anderen Angular-Bibliotheken heißt der Service **ApiService**.

Eigene HTML-Anwendungen erstellen und bearbeiten

Um eigene HTML-Anwendungen zu erstellen und zu bearbeiten, können Sie Angular-Bibliotheken ändern und Plugins zum API Server hinzufügen.

TIPP: Zur Veranschaulichung stellt Ihnen One Identity im Angular-Workspace die beispielhafte HTML5-Anwendung **custom-app** zur Verfügung.

One Identity Manager-Aktualisierungen übernehmen

Sobald One Identity Produktaktualisierungen für One Identity Manager veröffentlicht, können Sie die entsprechenden Änderungen in Ihre angepassten HTML-Anwendungen übernehmen.

HINWEIS: Ihre angepassten HTML-Anwendungen werden nicht automatisch aktualisiert. Sie sind dafür verantwortlich, die Kompilierung und Bereitstellung aktueller Versionen einer HTML-Anwendung durchzuführen.

Um Änderungen aus einer One Identity Manager-Aktualisierung in Ihre HTML-Anwendungen zu übernehmen

1. Übernehmen Sie die Änderungen am Quelltext aus dem GitHub-Repository der Standard-HTML-Anwendungen in Ihre HTML-Anwendung (siehe <https://docs.github.com/pull-requests/collaborating-with-pull-requests/working-with-forks/merging-an-upstream-repository-into-your-fork>).
2. Kompilieren Sie Ihre HTML-Anwendung (siehe [HTML-Anwendungen kompilieren und bereitstellen](#) auf Seite 14) und beheben sie eventuell auftretende Kompilierfehler.
3. Prüfen Sie die One Identity Manager-Versionshinweise auf Änderungen der Standard-API oder Datenformate und prüfen Sie, ob Ihre HTML-Anwendungen noch ordnungsgemäß funktionieren.
4. Stellen Sie die neue Version Ihrer HTML-Anwendung bereit (siehe [HTML-Anwendungen kompilieren und bereitstellen](#) auf Seite 14).

Bibliotheken anpassen

Wenn Sie den Code einer Angular-Bibliothek ändern, müssen Sie alle Angular-Bibliotheken kompilieren und eigene Versionen aller Angular-Anwendungen, die die geänderte Angular-Bibliothek verwenden sollen, erstellen und bereitstellen. Dies gilt unabhängig davon, welche Angular-Bibliotheken geänderten Code enthalten.

Wenn Sie beispielsweise die Angular-Bibliothek **qer** ändern, müssen Sie auch alle Angular-Bibliotheken und die Angular-Anwendungen **qer-app-portal**, **qer-app-operationsupport** und **qer-app-pwdportal** kompilieren, da all diese Anwendungen die Angular-Bibliothek **qer** enthalten.

Falls Sie Nx einsetzen, übernimmt Nx die Verwaltung der Abhängigkeiten zwischen Bibliotheken. Verwenden Sie den Befehl `npm run nx:build-all`, um alle Projekte in der festgelegten Reihenfolge zu kompilieren.

Plugins hinzufügen

Plugins sind Angular-Bibliotheken, die zur Laufzeit dynamisch geladen werden. Die Plugins werden vom API Server verwaltet. Plugins werden automatisch vom API Server erkannt, indem im Programmverzeichnis nach Dateien mit dem Namen `imx-plugin-config.json` gesucht wird.

Die folgende Beispieldatei legt fest, dass die Angular-Plugin-Bibliothek `ccc` in die **qer-app-portal**-Anwendung geladen werden soll. Der Name des Angular-Moduls, das instanziiert werden soll, ist **CustomConfigModule**.

```
{
  "qer-app-portal": [
    {
      "Container": "ccc",
      "Name": "CustomConfigModule"
    }
  ]
}
```

Um ein Plugin hinzuzufügen

1. Erstellen Sie auf dem API Server die Datei `imxweb\<Name der Angular-Plugin-Bibliothek>\imx-plugin-config.json` mit folgendem Inhalt:

```
{
  "<Name of the HTML application>": [
    {
      "Container": "<Name of the Angular plugin library>",
      "Name": "<Name of the Angular module>"
    }
  ]
}
```

2. Importieren Sie die Datei mithilfe des Software Loaders in Ihre One Identity Manager-Datenbank und weisen Sie sie der Maschinenrolle **API Server** zu. Weitere Informationen zum Importieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
3. (Optional) Um zu prüfen, ob die HTML-Anwendung das Plugin richtig lädt, rufen Sie die URL `<URL des API Servers>/imx/applications` auf und prüfen Sie, dass an der HTML-Anwendung das entsprechende Plugin in der Liste erscheint.

Um ein Plugin mithilfe von Nx hinzuzufügen

1. Erstellen Sie mithilfe des Nx-Kommandozeilenprogramms ein neues Projekt (`nx generate lib <Projektname>`).
2. Öffnen Sie im neu erstellten Projekt die Datei `projects/<Projektname>/project.json` und fügen Sie alle Abhängigkeiten hinzu.
3. Nehmen Sie für jede HTML-Anwendung, die dieses Plugin verwenden soll, folgende Aktionen vor:
 - a. Öffnen Sie mithilfe eines Texteditors die Datei `project.json` der Anwendung (zum Beispiel `projects/qer-app-portal/project.json`).

- b. Fügen Sie unter `prebuild > dependsOn > projects` eine Bibliothek mit dem Namen Ihres neuen Projekts hinzu.

Heißt Ihr Projekt beispielsweise **CCC**, müssen Sie die Zeile entsprechend folgendermaßen ändern:

```
"projects": ["qer", "aad", "aob", "apc", "att", "CCC"]
```

- c. Speichern Sie die Datei.

Fehlende Übersetzungen prüfen

Sie können HTML-Anwendungen mithilfe des `ImxClient`-Kommandozeilenprogramms auf fehlende Übersetzungen der Benutzeroberfläche überprüfen. Weitere Informationen zum `ImxClient`-Kommandozeilenprogramm finden Sie im *One Identity Manager API-Entwicklungshandbuch*.

TIPP: Übersetzbare Texte werden im Quelltext mit dem Prefix `#LDS#` gekennzeichnet.

Um eine HTML-Anwendung auf fehlende Übersetzungen zu überprüfen

1. Starten Sie das `ImxClient`-Kommandozeilenprogramm.
2. Führen Sie im Ordner, den Sie auf fehlende Übersetzungen prüfen möchten, den Befehl **check-translations** aus.

Ein Bericht wird erstellt. Dieser Bericht zeigt Ihnen alle Dateien, in denen Texte gefunden wurden, die noch nicht oder nur teilweise übersetzt wurden.


3. (Optional) Um die Übersetzungsschlüssel und Übersetzungen anzulegen, verwenden Sie das Programm `Designer`. Weitere Informationen zu Übersetzungen finden Sie im *One Identity Manager Konfigurationshandbuch*.

HTML-Anwendungen registrieren

Um neue HTML-Anwendungen für die Nutzung zur Verfügung zu stellen und somit auf der Startseite des API Servers anzuzeigen, müssen Sie die HTML-Anwendungen in der Datenbank anlegen.

Um eine HTML-Anwendung in der Datenbank anzulegen

1. Starten Sie das Programm `Designer`.
2. Verbinden Sie sich mit der entsprechenden Datenbank.

3. Klicken Sie in der Navigation die Kategorie **Basisdaten** > **Sicherheitseinstellungen** > **HTML-Anwendungen**.
4. In der Menüleiste klicken Sie  (**Ein neues Objekt erstellen**).
5. In der Liste klicken Sie den neuen Eintrag.
6. Im Bereich **Eigenschaften** geben Sie in den entsprechenden Feldern die Daten der HTML-Anwendung an. Geben Sie mindestens die folgenden Informationen an:
 - **Anzeigename:** Geben Sie einen Namen für die HTML-Anwendung ein.
 - **HTML-Anwendung:** Geben Sie den Pfad CCC/<Name Ihrer HTML-Anwendung> ein.
 - **Vorkompiliert:** Setzen Sie den Wert auf **True**.

HTML-Anwendungen kompilieren und bereitstellen

Um eine HTML-Anwendung zur Verfügung zu stellen, müssen Sie die HTML-Anwendung kompilieren und das Paket als ZIP-Datei verfügbar machen.

Um eine HTML-Anwendung über den API Server zu kompilieren und bereitzustellen

1. Starten Sie ein Kommandozeilenprogramm.
2. Wechseln Sie in das Verzeichnis des Angular-Workspace.
3. Kompilieren Sie alle Bibliotheken, die von der HTML-Anwendung geladen werden. Falls Sie Nx einsetzen, führen Sie dazu den folgenden Befehl aus:

```
npx build <Anwendungsname>
```

HINWEIS: Angular-Bibliotheken müssen nicht gesondert bereitgestellt werden. HTML-Anwendungen enthalten alle benötigten Bibliotheken.

4. Führen Sie den folgenden Befehl aus:

```
ng build <Anwendungsname>
```

5. Wechseln Sie in das Verzeichnis mit dem Kompilat (üblicherweise `dist/<Anwendungsname>`) und fügen Sie alle Dateien und Unterordner einer neuen ZIP-Datei mit dem Namen `Htm1_<Anwendungsname>.zip` hinzu.
6. Kopieren Sie die ZIP-Datei in den Unterordner `imxweb\custom` Ihrer Arbeitsumgebung.

TIPP: Sollte der Ordner noch nicht existieren, erstellen Sie ihn.
7. Importieren Sie die ZIP-Datei mithilfe des Programms Software Loader 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*.

8. Kopieren Sie die ZIP-Datei in den Unterordner `bin\imxweb\custom` Ihrer IIS-Installation.
| TIPP: Sollte der Ordner noch nicht existieren, erstellen Sie ihn.
9. Importieren Sie die ZIP-Datei mithilfe des Programms Software Loader in Ihre One Identity Manager-Datenbank und weisen Sie sie der Maschinenrolle **Business API Server** zu.
10. (Optional) Um zu prüfen, ob Ihr Angular-Paket (und damit Ihre HTML-Anwendung) korrekt geladen werden, prüfen Sie ob im Administrationsportal die entsprechenden Pakete mit dem Zusatz **Angepasstes Paket** versehen sind. Weitere Informationen zum Anzeigen der Pakete finden Sie im *One Identity Manager Konfigurationshandbuch für Webanwendungen*.

Um eine HTML-Anwendung ohne API Server zu kompilieren und bereitzustellen

1. Exportieren Sie die API Server-Dateistruktur mithilfe des Programms Software Loader aus Ihrer One Identity Manager-Datenbank in einen lokalen Ordner und wählen Sie dabei die Maschinenrollen **Server, Web, Business API Server** und **SCIM Provider**. Weitere Informationen zum Exportieren von Dateien mit dem Software Loader finden Sie im *One Identity Manager Administrationshandbuch für betriebsunterstützende Aufgaben*.
2. Starten Sie ein Kommandozeilenprogramm.
3. Wechseln Sie in das Verzeichnis des Angular-Workspace.
4. Kompilieren Sie alle Bibliotheken, die von der HTML-Anwendung geladen werden. Falls Sie Nx einsetzen, führen Sie dazu den folgenden Befehl aus:

```
npx build <Anwendungsname>
```

| HINWEIS: Angular-Bibliotheken müssen nicht gesondert bereitgestellt werden. HTML-Anwendungen enthalten alle benötigten Bibliotheken.

5. Führen Sie den folgenden Befehl aus:

```
ng build <Anwendungsname>
```

6. Wechseln Sie in das Verzeichnis mit dem Kompilat (üblicherweise `dist/<Anwendungsname>`) und fügen Sie alle Dateien und Unterordner einer neuen ZIP-Datei mit dem Namen `Htm1_<Anwendungsname>.zip` hinzu.
7. Kopieren Sie die ZIP-Datei in den Unterordner `imxweb\custom` Ihrer Arbeitsumgebung.
| TIPP: Sollte der Ordner noch nicht existieren, erstellen Sie ihn.
8. Importieren Sie die ZIP-Datei mithilfe des Programms Software Loader 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*.

9. Kopieren Sie die ZIP-Datei in den Unterordner `bin\imxweb\custom` des im ersten Schritt erstellten Ordners.

| **TIPP:** Sollte der Ordner noch nicht existieren, erstellen Sie ihn.

10. Importieren Sie die ZIP-Datei mithilfe des Programms Software Loader in Ihre One Identity Manager-Datenbank und weisen Sie sie der Maschinenrolle **Business API Server** zu.
11. (Optional) Um zu prüfen, ob Ihr Angular-Paket (und damit Ihre HTML-Anwendung) korrekt geladen werden, prüfen Sie ob im Administrationsportal die entsprechenden Pakete mit dem Zusatz **Angepasstes Paket** versehen sind. Weitere Informationen zum Anzeigen der Pakete finden Sie im *One Identity Manager Konfigurationshandbuch für Webanwendungen*.

Debugging

Das Ausführen und Debuggen von HTML-Anwendungen ist mit den Standardwerkzeugen der Angular-CLI-Toolchain möglich.

Sie können beispielsweise den Befehl `ng serve qer-app-portal` verwenden, um die Web Portal-HTML-Anwendung zu debuggen.

Um eine HTML-Anwendung zu debuggen

1. Hosten Sie den API Server lokal (siehe [API Server lokal hosten](#) auf Seite 17).
2. Starten Sie ein Kommandozeilenprogramm.
3. Wechseln Sie in das Verzeichnis des Angular-Workspace.
4. Führen Sie den folgenden Befehl aus:

```
npm run start <Name der HTML-Anwendung>
```

Ein Webserver, der standardmäßig unter der URL `http://localhost:4200` erreichbar ist und die HTML-Anwendung hostet, wird gestartet.

5. Starten Sie das Debugging in einer entsprechenden Entwicklungsumgebung (beispielsweise Visual Studio Code).

API Server lokal hosten

Um HTML-Anwendungen zu debuggen und zu entwickeln, benötigen Sie eine API Server-Instanz, mit der sich die HTML-Anwendungen verbinden. Für diese Zwecke können Sie einen API Server lokal hosten.

HINWEIS: Die HTML-Anwendungen verbinden sich mit dem API Server über die URL, die in der Datei `environment.ts` der HTML-Anwendungen definiert ist. Die Standard-URL, unter der ein lokal gehosteter API Server ausgeführt wird, ist `http://localhost:8182`.

Um einen API Server lokal zu hosten

1. Starten Sie ein Kommandozeilenprogramm.
2. Wechseln Sie in das One Identity Manager-Installationsverzeichnis.
3. Führen Sie den folgenden Befehl aus:

```
imxclient.exe run-apiserver -B
```

HINWEIS: Dieses Kommando verwendet nicht die Ergebnisse einer lokalen Kompilierung aus Ihrer Entwicklungsumgebung, sondern den kompilierten Stand der HTML-Anwendungen aus der Auslieferung. Wie Sie eine HTML-Anwendung aus Ihrer lokalen Entwicklungsumgebung aufrufen, erfahren Sie unter [Debugging mit Plugins](#) auf Seite 17.

Debugging mit Plugins

Sie können das Debugging auch mit Plugins durchführen. Das Debugging mit Plugins funktioniert nur, wenn der lokale API Server das Plugin auch finden kann.

Um eine statische Angular-Bibliothek zu debuggen

1. Hosten Sie den API Server lokal (siehe [API Server lokal hosten](#) auf Seite 17).
2. Starten Sie ein Kommandozeilenprogramm.
3. Wechseln Sie in das Verzeichnis des Angular-Workspace.
4. Führen Sie den folgenden Befehl aus:

```
npm run build:watch <Name der Angular-Bibliothek>
```

5. Starten Sie ein weiteres Kommandozeilenprogramm.
6. Wechseln Sie in das Verzeichnis des Angular-Workspace.
7. Führen Sie den folgenden Befehl aus:

```
npm run start <Name der HTML-Anwendung>
```

Ein Webserver, der standardmäßig unter der URL `http://localhost:4200` erreichbar ist und die HTML-Anwendung hostet, wird gestartet.

8. Starten Sie das Debugging in einer entsprechenden Entwicklungsumgebung (beispielsweise Visual Studio Code).

Um eine Angular-Plugin-Bibliothek zu debuggen

1. Hosten Sie den API Server lokal (siehe [API Server lokal hosten](#) auf Seite 17).
2. Starten Sie ein Kommandozeilenprogramm.
3. Wechseln Sie in das Verzeichnis des Angular-Workspace.
4. Führen Sie den folgenden Befehl aus:

```
npm run build:watch:dynamic <Angular-Plugin-Bibliothek>
```

5. Starten Sie ein weiteres Kommandozeilenprogramm.
6. Wechseln Sie in das Verzeichnis des Angular-Workspace.
7. Führen Sie den folgenden Befehl aus:

```
npm run start <Name der HTML-Anwendung>
```

Ein Webserver, der standardmäßig unter der URL `http://localhost:4200` erreichbar ist und die HTML-Anwendung hostet, wird gestartet.

8. Starten Sie das Debugging in einer entsprechenden Entwicklungsumgebung (beispielsweise Visual Studio Code).

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