DOPaK2020

Development

Marvin Rohrbach | Martin Kozianka
Noah Pfeffel | Sebastian Straub
Dr. Veit Jahns | Thomas Richter

D5

Development

Inhalt

- DOCUMENTS Manager 2.0 (Marvin Rohrbach, Martin Kozianka)
- OAuth2.0 (Noah Pfeffel, Sebastian Straub)
- Scripting Libraries (Dr. Veit Jahns)
- VS Code Extension (Thomas Richter)

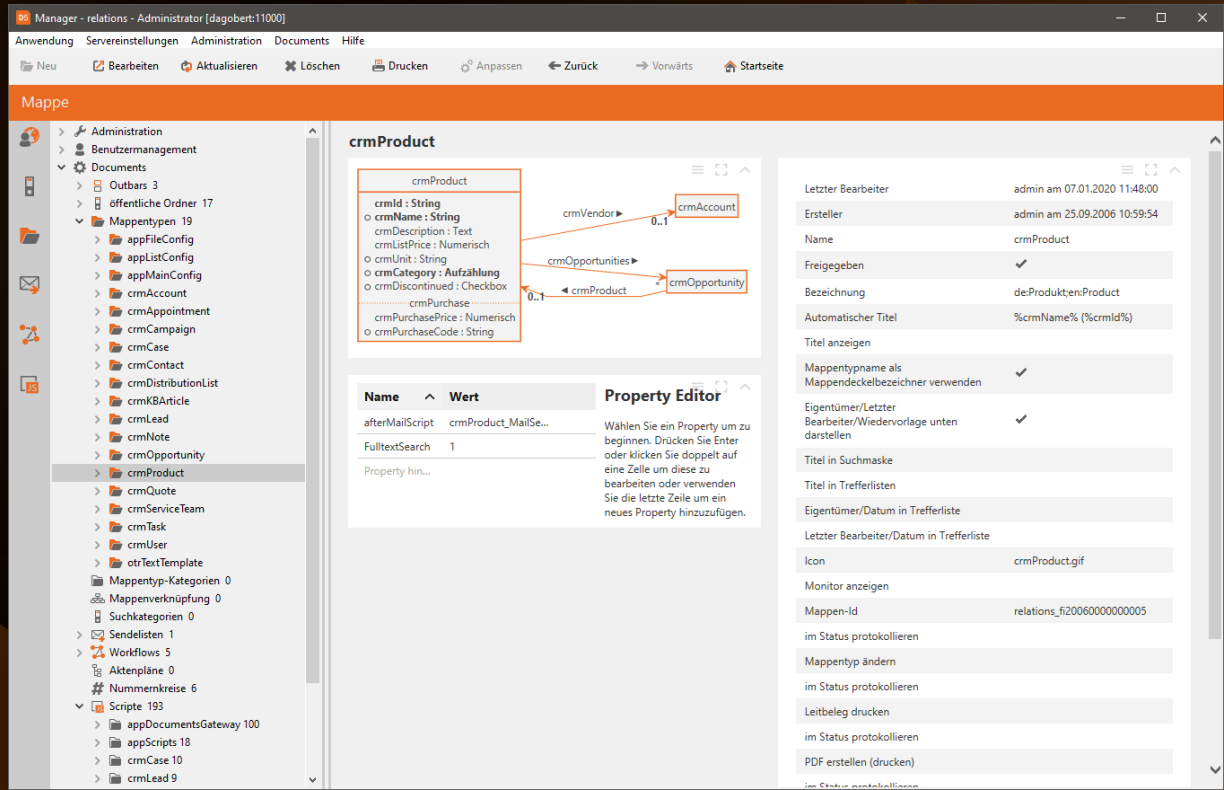
DOCUMENTS Manager 2.0

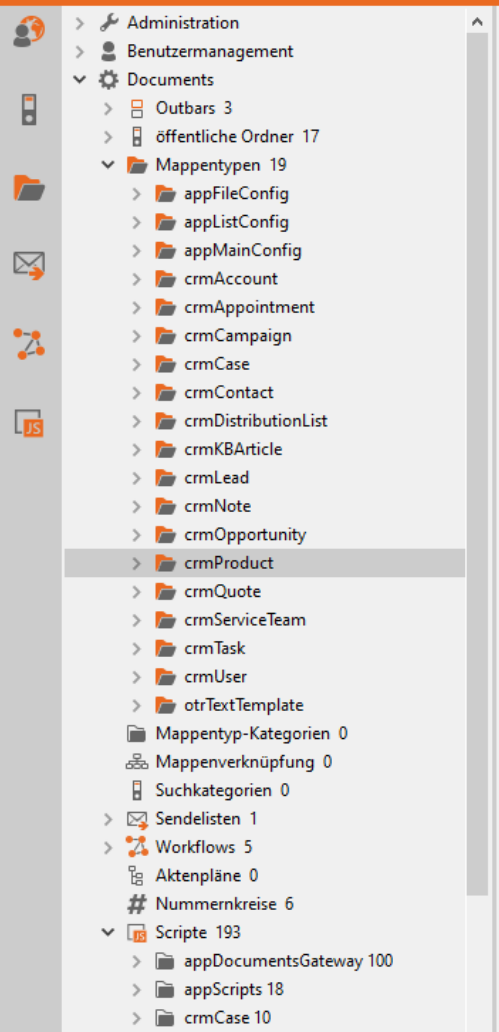
Neues Design

- Moderner
- Touchfreundlicher

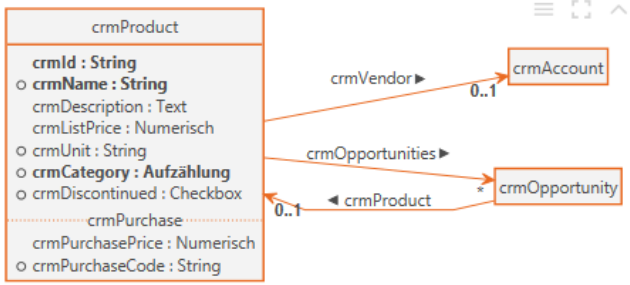
Neue Funktionen

- Diagramme
- Codeeditor
- Propertyeditor





crmProduct



Name	Wert
afterMailScript	crmProduct_MailSe...
FulltextSearch	1
Property hin...	

Property Editor

Wählen Sie ein Property um zu beginnen. Drücken Sie Enter oder klicken Sie doppelt auf eine Zelle um diese zu bearbeiten oder verwenden Sie die letzte Zeile um ein neues Property hinzuzufügen.

Automatisches Layout
Je nach verfügbarem Platz

Letzter Bearbeiter	admin am 07.01.2020 11:48:00
Ersteller	admin am 25.09.2006 10:59:54
Name	crmProduct
Freigegeben	✓
Bezeichnung	de:Produkt;en:Product
Automatischer Titel	%crmName% (%crmId%)
Titel ändern	
Mappentyp ändern	✓
Mappenicon ändern	
Eigentümer/Letzter Bearbeiter/Wiedervorlage unten darstellen	✓
Titel in Suchmaske	
Titel in Trefferlisten	
Eigentümer/Datum in Trefferliste	
Letzter Bearbeiter/Datum in Trefferliste	
Icon	crmProduct.gif
Monitor anzeigen	
Mappen-Id	relations_fi20060000000005
im Status protokollieren	
Mappentyp ändern	
im Status protokollieren	
Leitbeleg drucken	
im Status protokollieren	
PDF erstellen (drucken)	

Menü
Aktiviert erweiterte Funktionen

Vollbild
Zeigt die Kachel im gesamten Bereich an

Minimieren
Blendet die Kachel aus

Property Editor

Kategorien

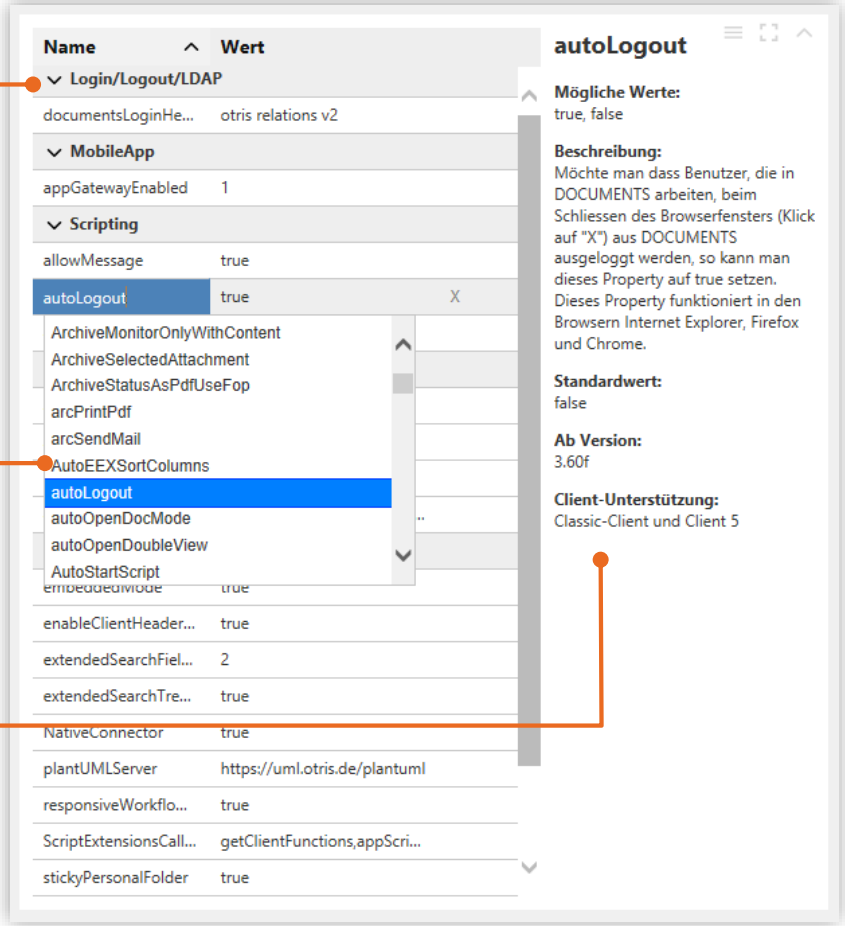
- Properties können nach Kategorien sortiert angezeigt werden
- Konfigurierbar über den Menüknopf oben rechts

Autovervollständigung

- Properties und deren Werte werden automatisch vorgeschlagen

Dokumentation

- Die Dokumentation der selektierten Eigenschaft wird rechts angezeigt



Skript Editor

Suchfunktion

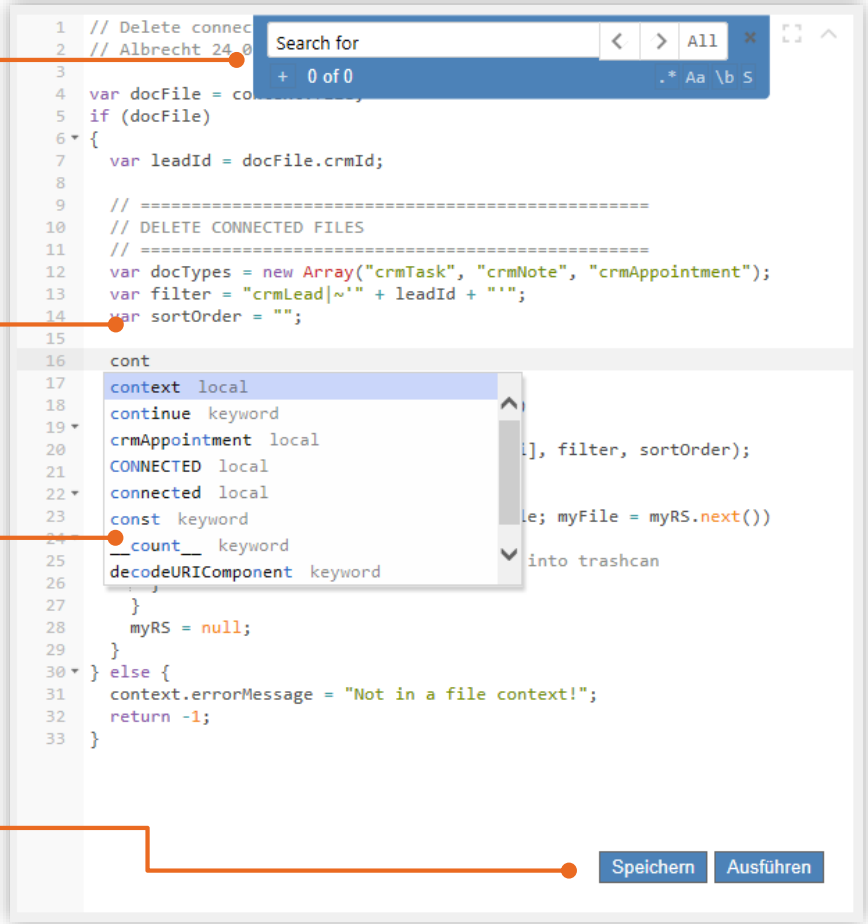
- Skript kann im Manager durchsucht werden (Strg+F)
- Ersetzungen sind auch möglich (Strg+H)

Syntaxhervorhebung

Autovervollständigung

- Bereits verwendete Elemente werden automatisch vervollständigt

Speichern und Ausführen

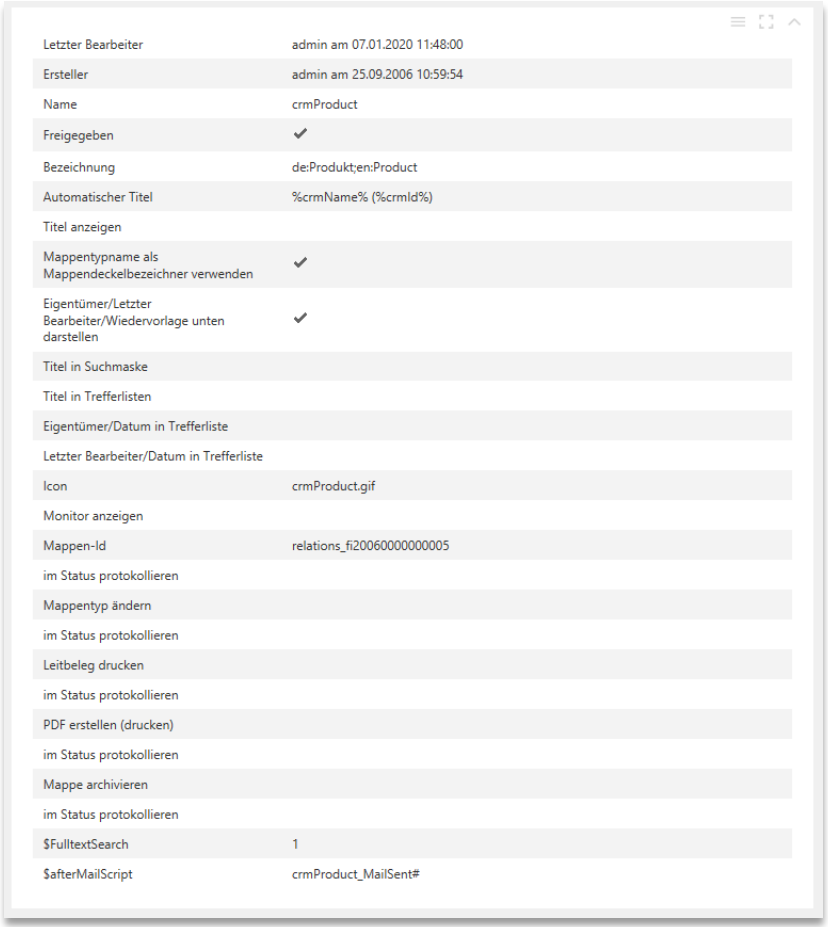


Detail Ansicht

Darstellung der Objektattribute

Zwei Ansichten

- Alle Attribute
- Attribute, welche verändert wurden
  - D.h. aktueller Wert != Standardwert
- Umgeschaltet wird mit dem Menüknopf oben rechts

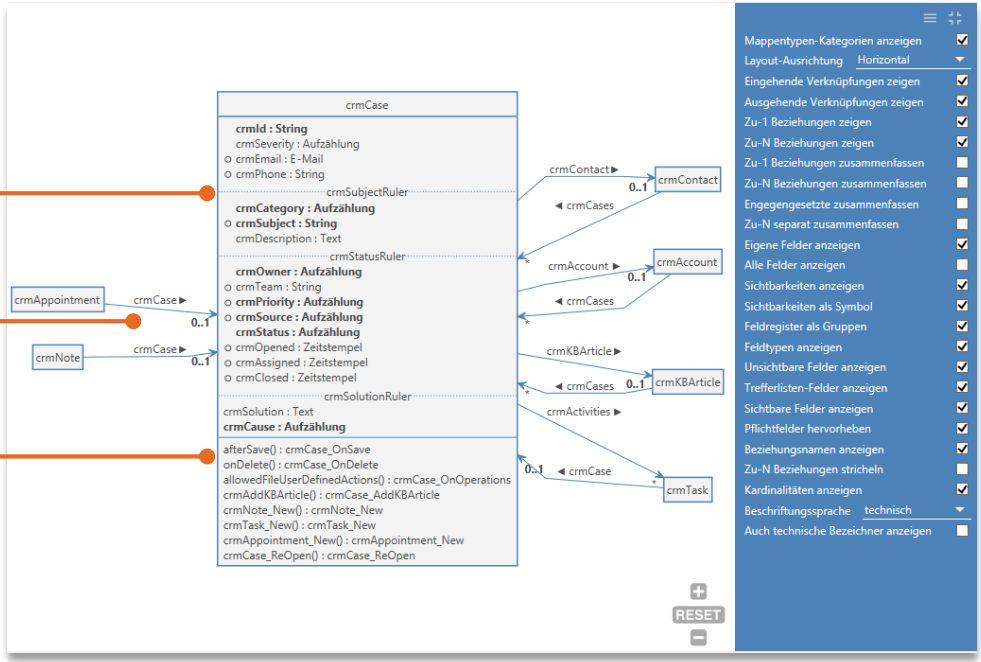


Letzter Bearbeiter	admin am 07.01.2020 11:48:00
Ersteller	admin am 25.09.2006 10:59:54
Name	crmProduct
Freigegeben	✔
Bezeichnung	de:Produkt;en:Product
Automatischer Titel	%crmName% (%crmId%)
Titel anzeigen	
Mappentypname als Mappendeckelbezeichner verwenden	✔
Eigentümer/Letzter Bearbeiter/Wiedervorlage unten darstellen	✔
Titel in Suchmaske	
Titel in Trefferlisten	
Eigentümer/Datum in Trefferliste	
Letzter Bearbeiter/Datum in Trefferliste	
Icon	crmProduct.gif
Monitor anzeigen	
Mappen-Id	relations_fi20060000000005
im Status protokollieren	
Mappentyp ändern	
im Status protokollieren	
Leitbeleg drucken	
im Status protokollieren	
PDF erstellen (drucken)	
im Status protokollieren	
Mappe archivieren	
im Status protokollieren	
$FulltextSearch	1
$afterMailScript	crmProduct_MailSent#

Mappentyp Diagramm

Grafische Aufbereitung des Mappentypen

- Felder
- Beziehungen
- Aktionen



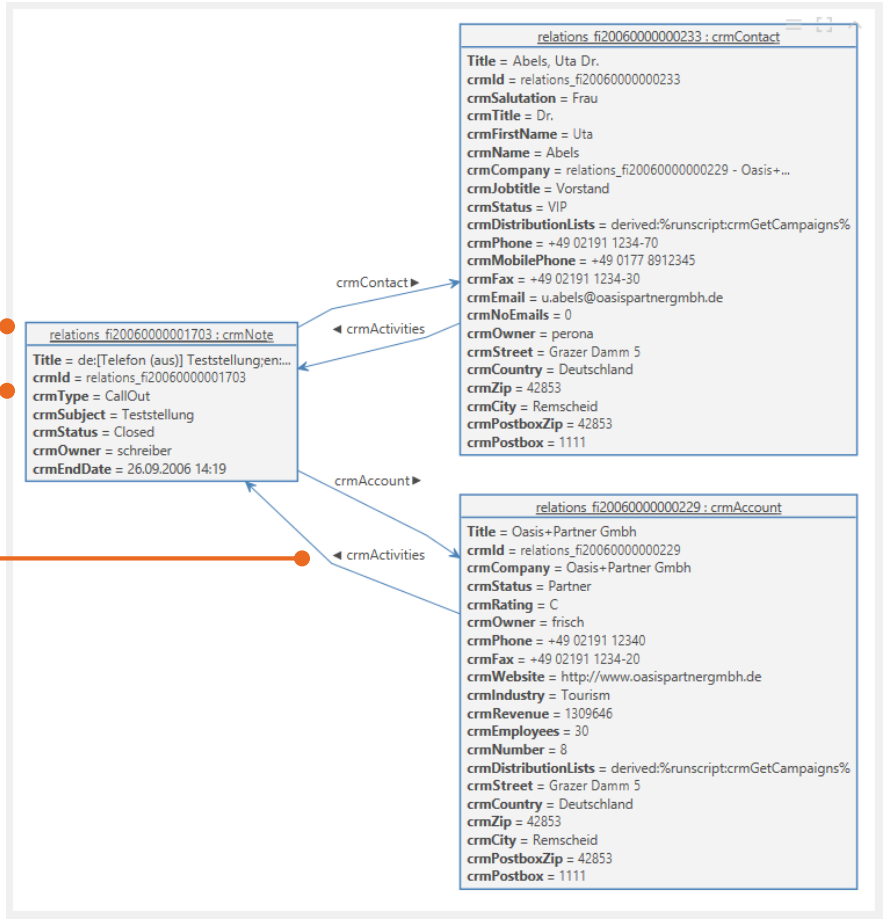
Erweiterte Konfigurationen

- Erreichbar über den Menüknopf oben rechts

Akten Diagramm

Grafische Aufbereitung eines Vorgangs

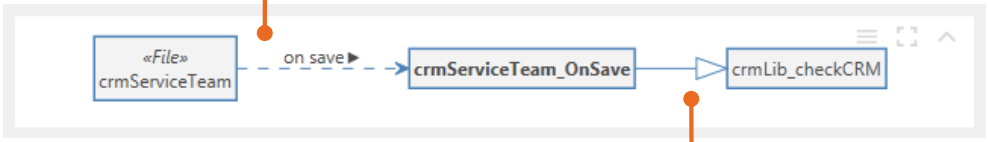
- Mappenid und Typ
- Felder und Feldwerte
- Beziehungen



Skript Diagramm

Einbindungspunkte des Skripts

- Hinweis: Nicht alle Einbindungspunkte werden untersucht, daher keine Garantie für Vollständigkeit



Importierte Skripte

- Nur #import wird unterstützt!

Exkurs: PlantUML

Um die Diagrammfunktionen zu Nutzen, muss ein PlantUML Server konfiguriert werden!

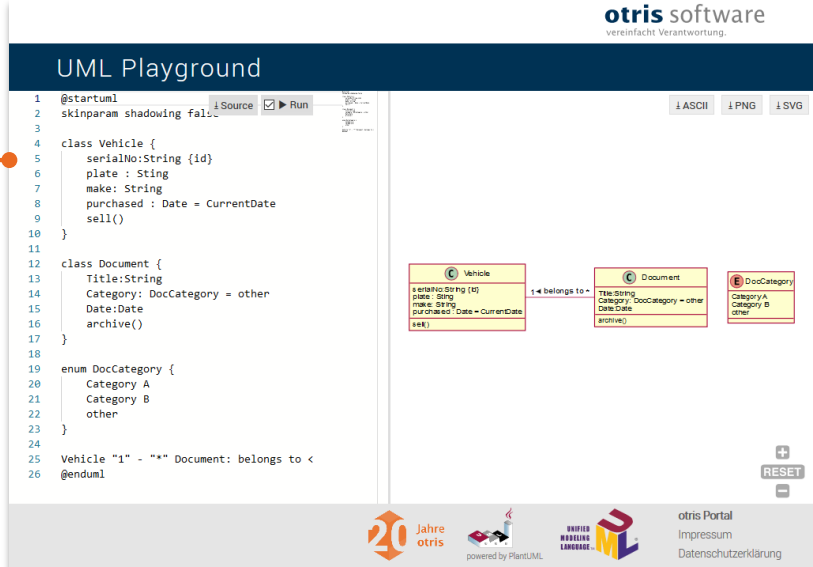
- Documents Einstellungen → Eigenschaften → plantUMLServer

Die otris stellt einen PlantUML Server bereit

- https://uml.otris.de/plantuml
- Kann auch direkt aufgerufen werden

Eigenes Hosting möglich

- Mehr Informationen: https://plantuml.com/



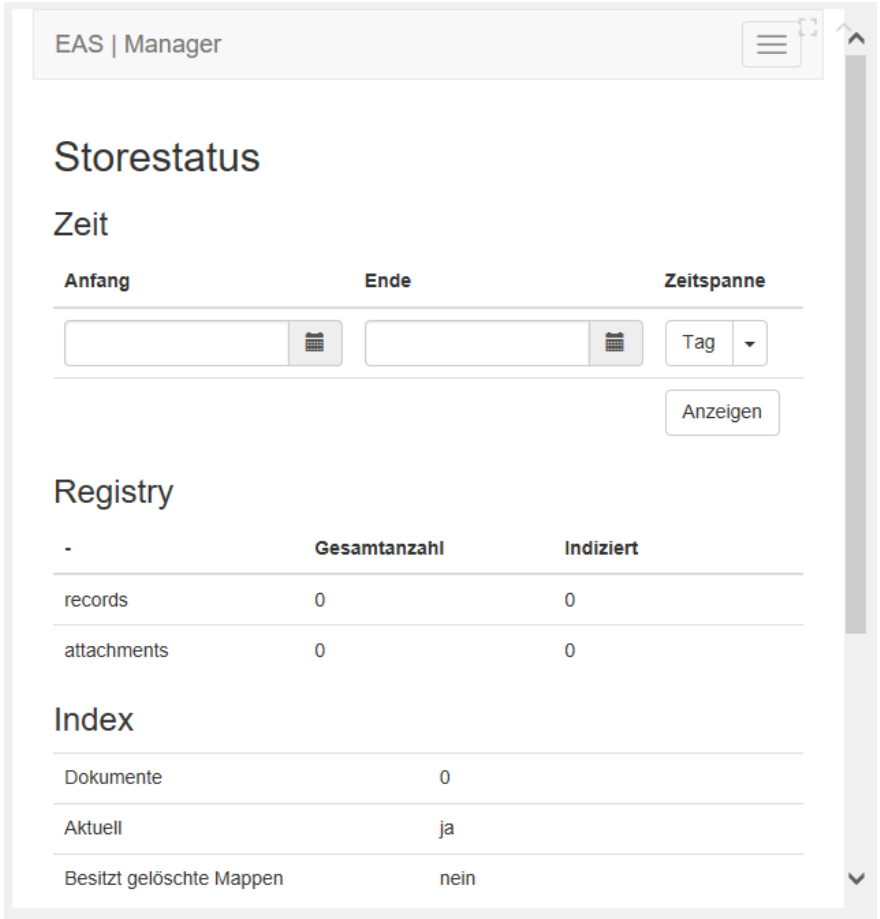
EAS-Manager

Darstellung des EAS-Managers

innerhalb des DOCUMENTS Managers

Folgende Voraussetzungen

- EAS Administratorkonto und Passwort müssen am Server konfiguriert sein
- Falls Manager extern gestartet wird: Portfreigabe des EAS-Servers



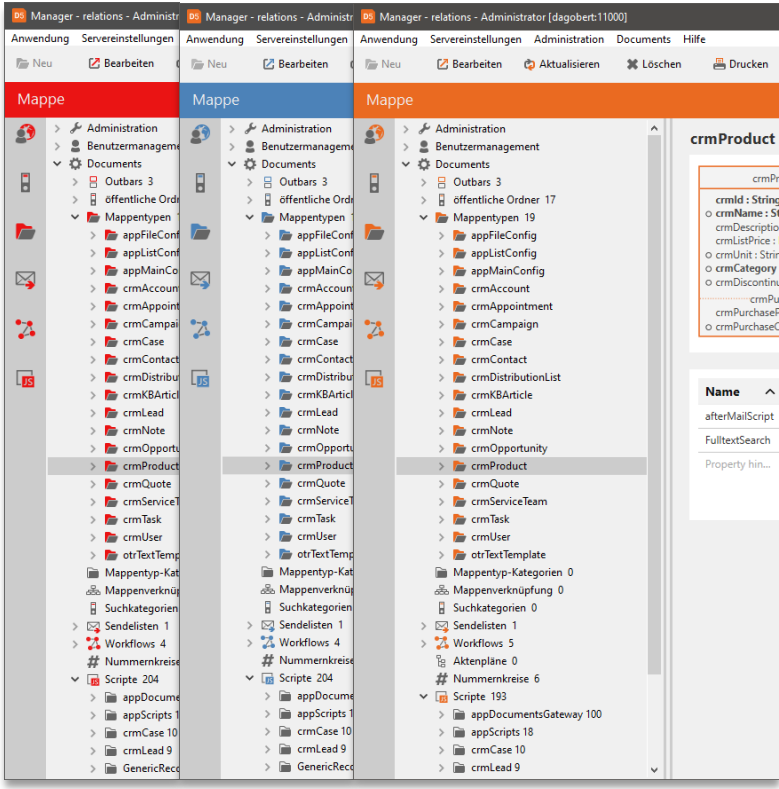
Manager Farbschema anpassen

Anpassung der Farbe des Managers

- Bsp: Rot als Warnhinweis für Produktivsystem

Serverweite Einstellungen in der documents.ini

- Highlightfarbe → $DMSkinColor1 234,20,20
- Hintergrundfarbe → $DMSkinColor2 80,80,80
- Einstellung in [Rot],[Grün],[Blau]



OAuth2.0

Inhalt

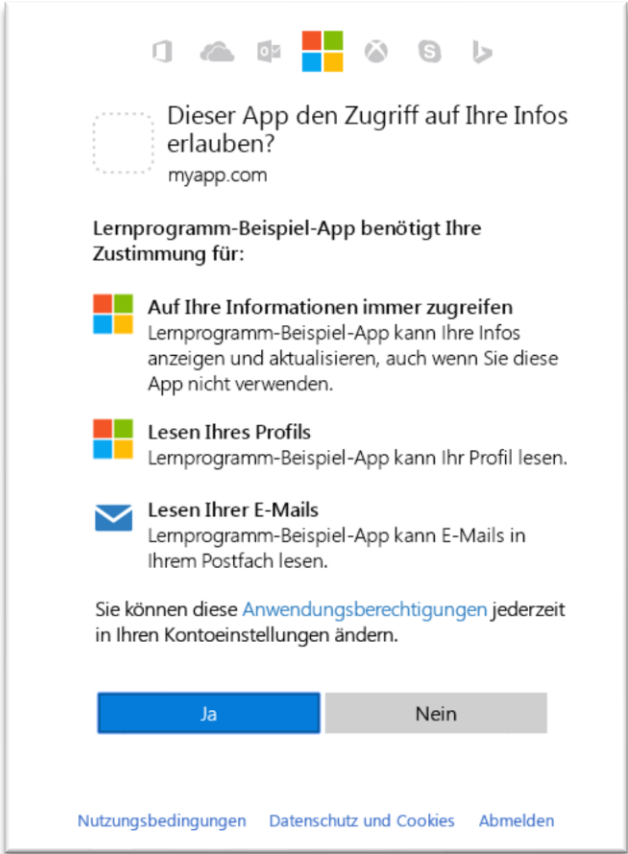
- Grundlagen
- OneDrive Anbindung
- Signatur mit DocuSign

Was ist OAuth2.0?

- Kurz für “Open Authorization” in Version 2.0
- Standardprotokoll für sichere API-Autorisierung
- Dient als Schnittstelle zum sicheren übermitteln von Daten zwischen Anwendungen

Stichwort Autorisierung

- Kein Weitergeben der Login-Daten
- Nur beschränkter Zugriff
- Zugriff kann von beiden Seiten entzogen werden



Wer unterstützt OAuth2.0?

Office 365

G Suite

DocuSign

salesforce

LinkedIn

GitHub

Trello

slack

Und viele mehr...

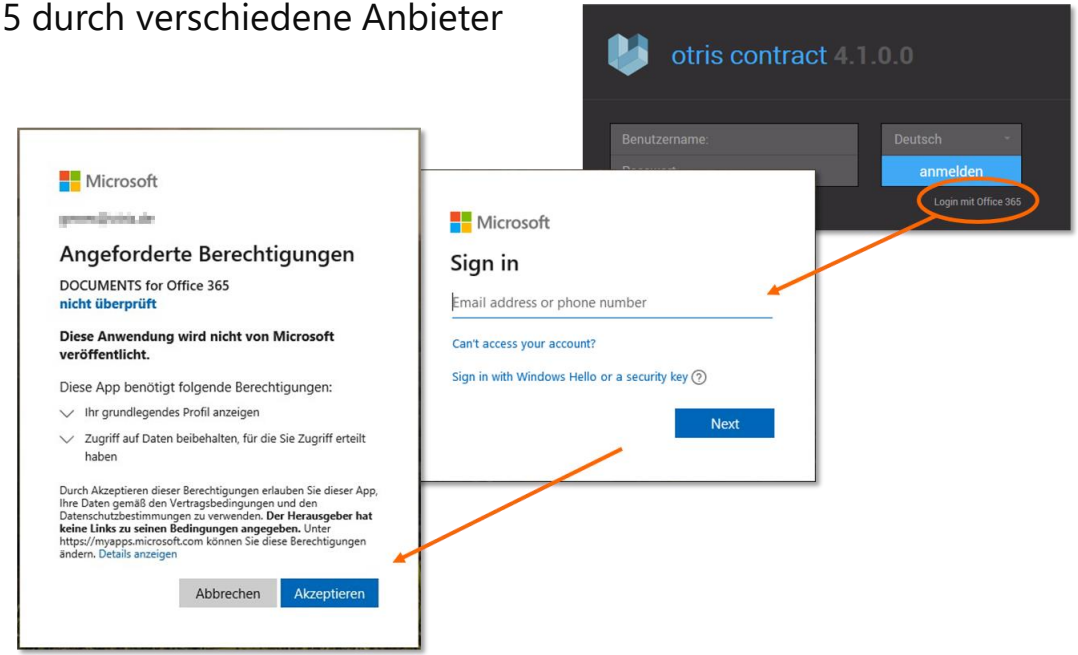
Login

OpenID Connect als Aufsatz für OAuth2.0

Ermöglicht den Login in Documents5 durch verschiedene Anbieter

Features je nach Anbieter:

- SSO
- Massenimport der Benutzer
- Individueller Benutzerimport
- 2-Faktor Authentifizierung



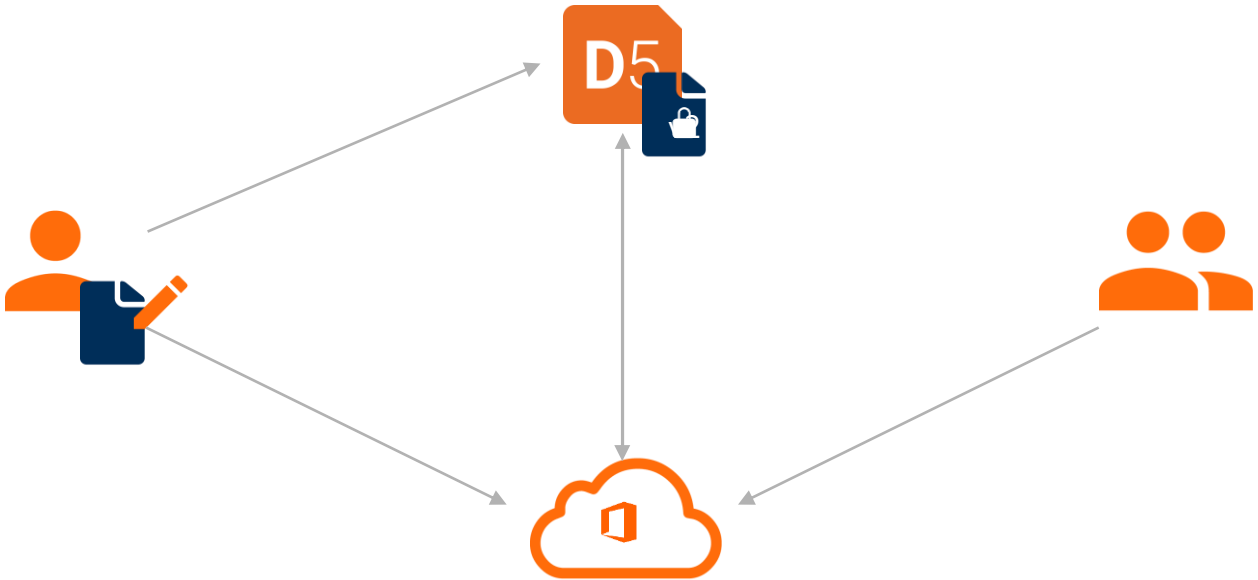
Login mit Benutzerimport

otris software

OneDrive

Kollaboratives Arbeiten

Kollaboratives Arbeiten in DOCUMENTS

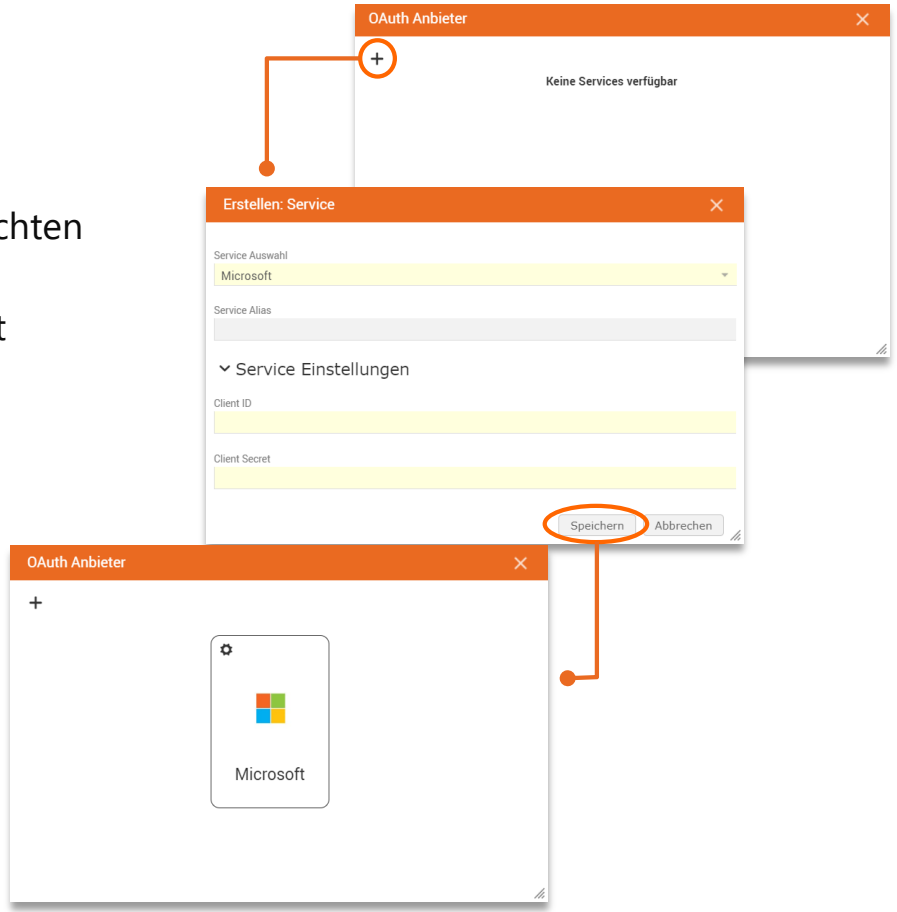


Einrichtung

- Administratoren können Verbindungen einrichten
- Microsoft wird als erster Anbieter unterstützt

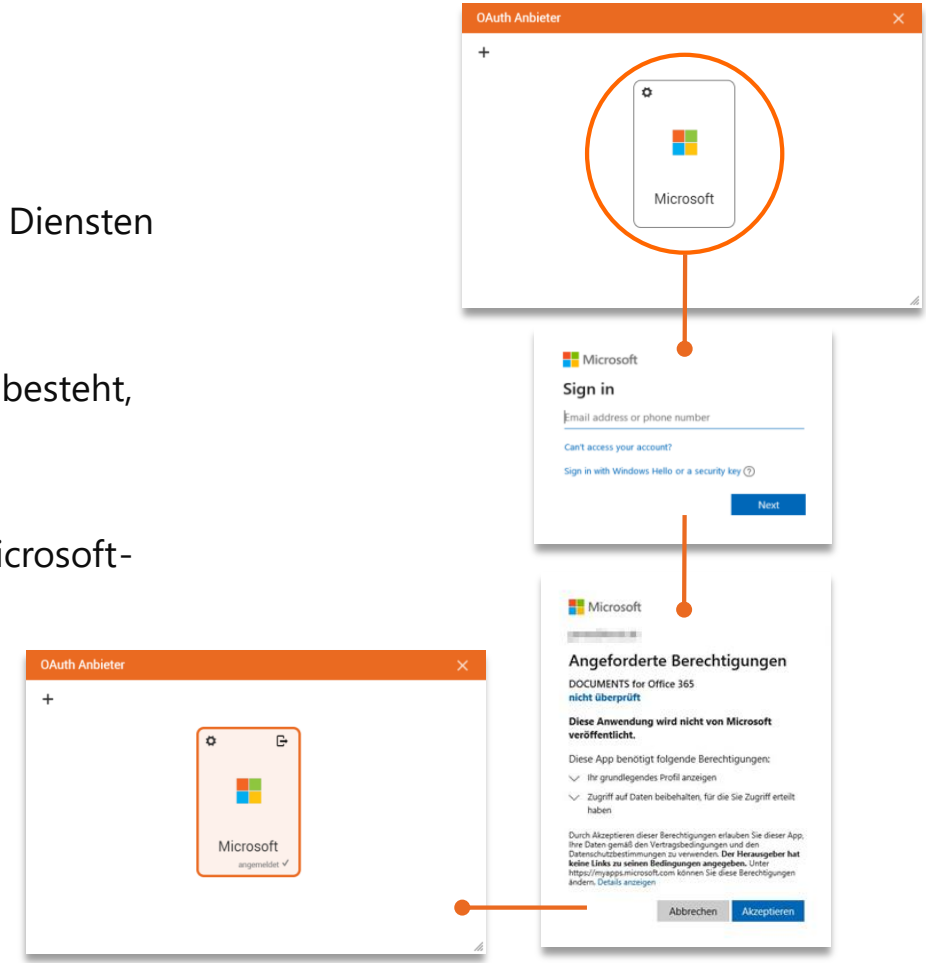
Aktuell eingebaute Microsoft Features:

- Benutzer für Dokumente freigeben
- Dokumente hochladen
- Dokumente herunterladen
- Benutzer importieren



Anmeldung

- Benutzer können sich bei eingerichteten Diensten anmelden
- Wenn SSO möglich ist und eine Session besteht, fallen die Microsoft-Schritte aus
- Der angemeldete Benutzer kann jetzt Microsoft-Funktionen in DOCUMENTS benutzen

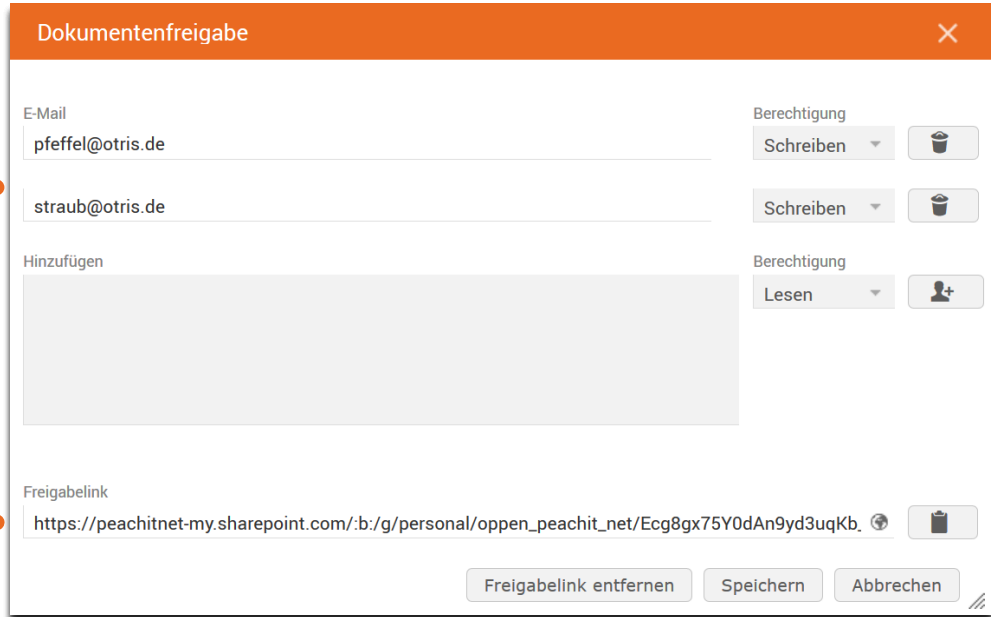


Dokumente Hochladen

- Voraussetzung: Benutzer ist bei einem Cloud-Service angemeldet
- Dokument auswählen und Hochladen-Button klicken

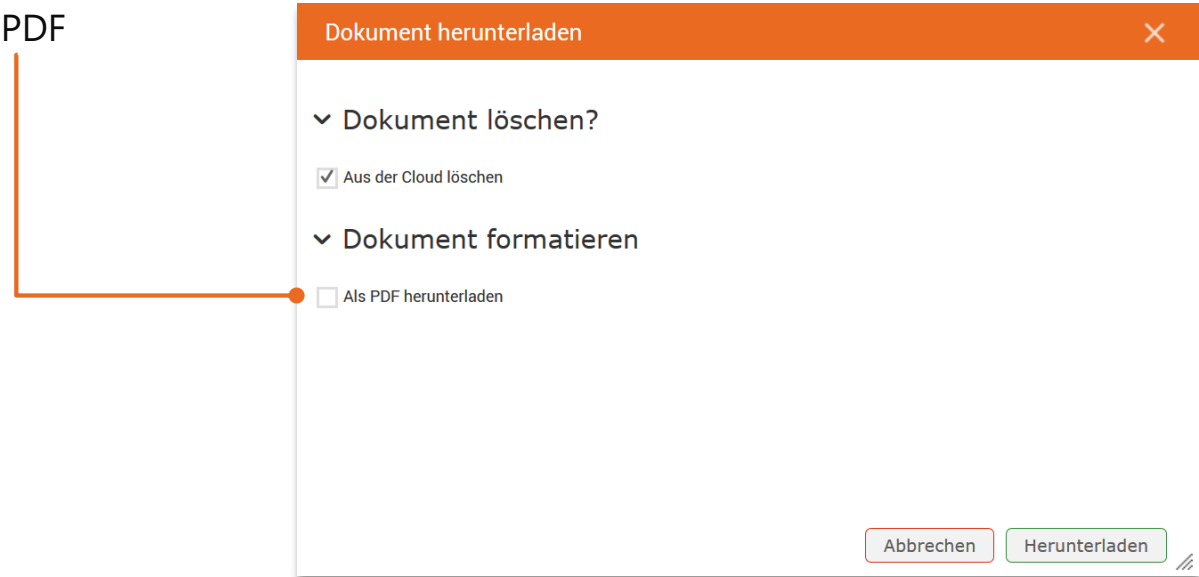
Dokumente Teilen

- Voraussetzungen: Dokument ist in der Cloud & Benutzer ist berechtigt
- Personen individuell berechtigen
- Freigabelink mit Leseberechtigung



Dokumente Herunterladen

- Voraussetzungen: Dokument ist in der Cloud & Benutzer ist berechtigt
- Bei Änderungen wird das alte Dokument versioniert
- Optional: Herunterladen als PDF



otris software

DocuSign

Verträge elektronisch signieren

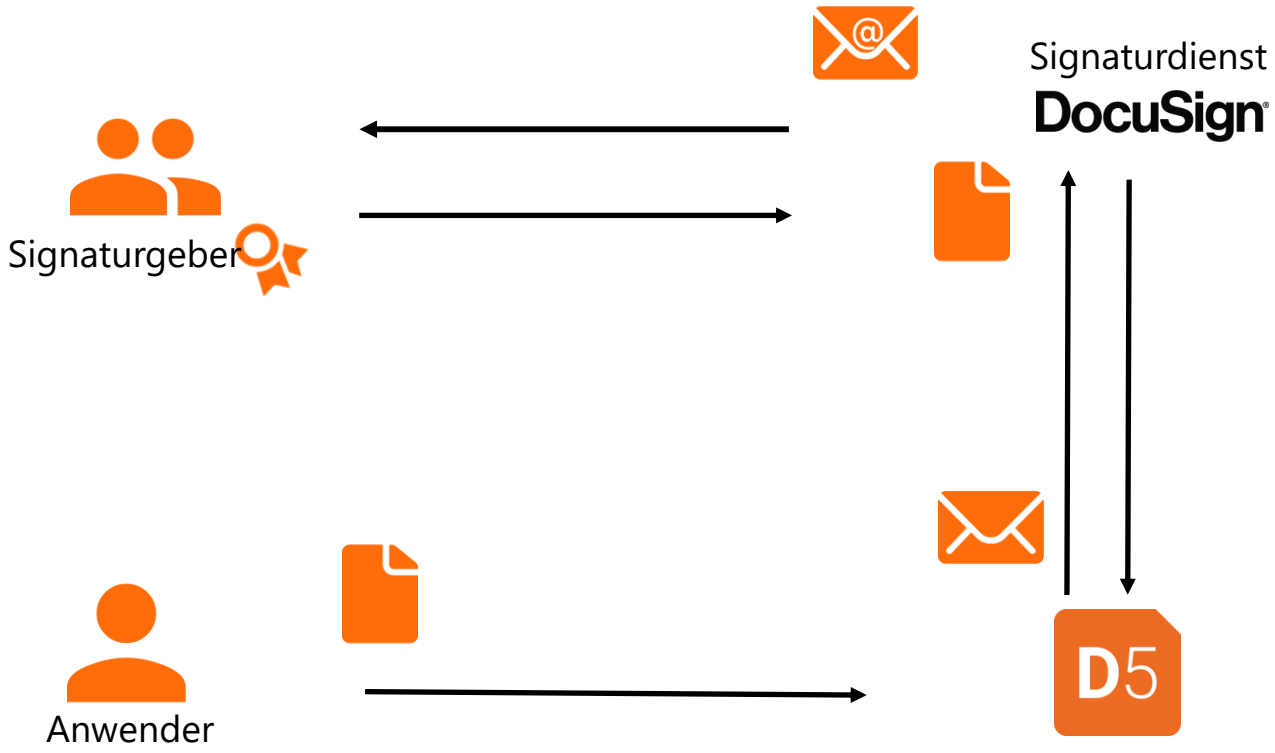
Die elektronische Signatur

Unter einer elektronischen Signatur versteht man

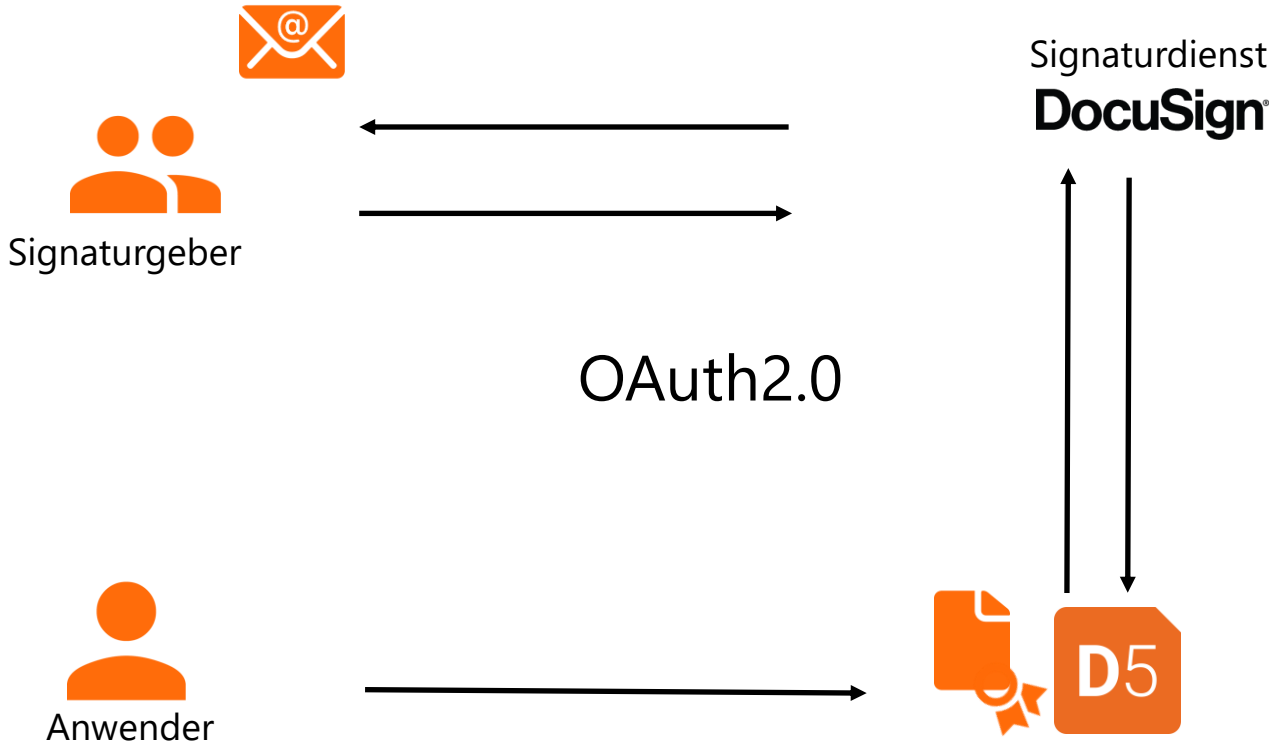
- Informationen in elektronischer Form, die
- logisch mit anderen Informationen (bspw. Dokumenten) verknüpft sind und
- zur Identifizierung des Signaturerstellers dienen.

Der Unterzeichner fügt einem Dokument weitere Daten hinzu, die er als Unterschrift verwendet.

Ablauf einer Unterschrift



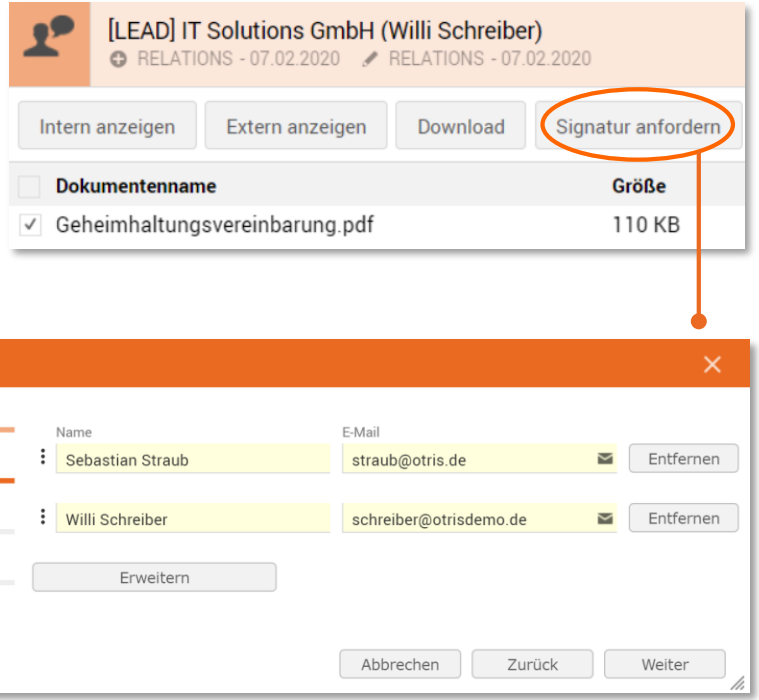
Ablauf einer Unterschrift



Unterschrift anfordern

Einheitlicher Prozess für alle Versendungen

- Einfaches Anfordern von Unterschriften
- Ohne Medienbruch direkt aus dem Dokumentenregister
- Geführter Dialog mit Datenvorauswahl
- Identischer Eingang unterschriebener Dokumente



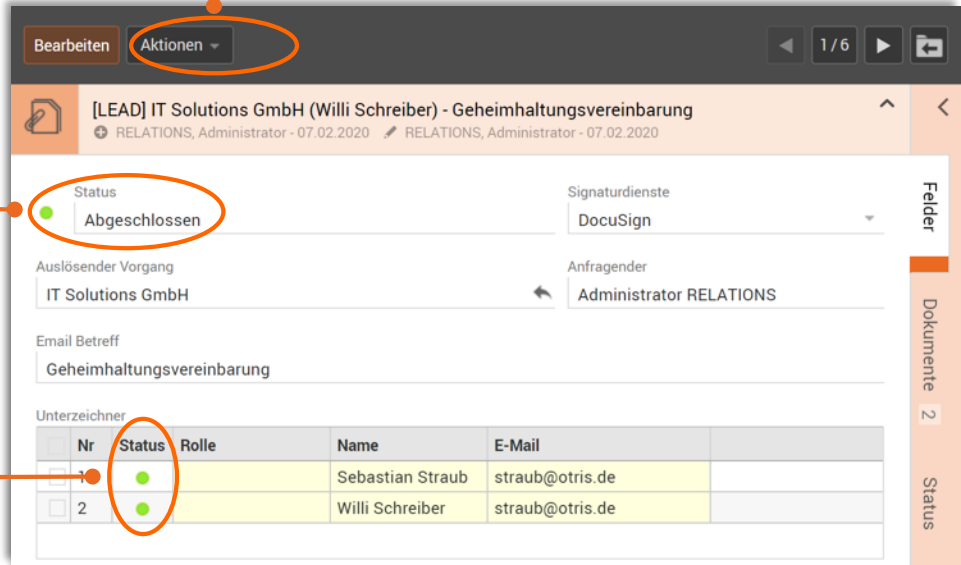
Monitoring und Überwachung

Erinnerungen verschicken

- Im Prozess ausstehende Signaturgeber benachrichtigen

Prozessüberwachung im Ampelsystem

- Gesamtstatus des Signaturdokuments
- Status der einzelnen Signaturgeber



OAuth2.0

Zusammenfassung

- ✓ Ein standardisiertes Protokoll zur Autorisierung
- ✓ Wird von vielen Anbieter unterstützt
- ✓ Einrichtung und Anmeldung kann über DOCUMENTS erfolgen
- ✓ Anmeldung bei Documents über OpenID Connect

Scripting Libraries

Inhalt

- otrLogger: Erweitertes Logging
- otrAssert: Erweiterte Fehlerbehandlung

otrLogger: Erweitertes Logging

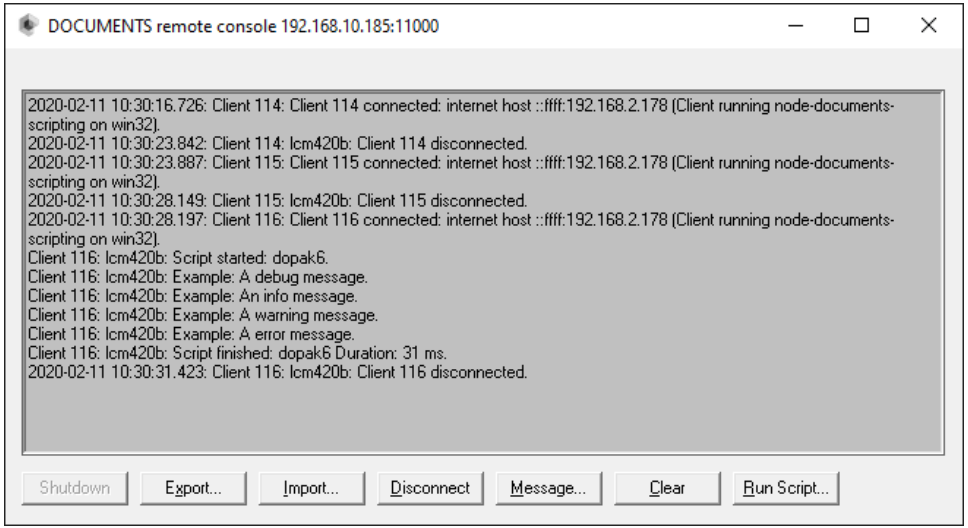
Log-Ausgabe per util.out()

- Unstrukturiert
- Keine Filtermöglichkeit
- Erschwert Fehlersuche
- Unnötige Log-Ausgaben im laufenden Betrieb

otrLogger

- Ausgabe pro Log-Level
- Filtern nach Log-Level
- Filtern nach Log-Kontext

```
util.out("Example: A debug message");
util.out("Example: An info message");
util.out("Example: A warning message");
util.out("Example: A error message");
```



Ausgabe von Log-Meldungen

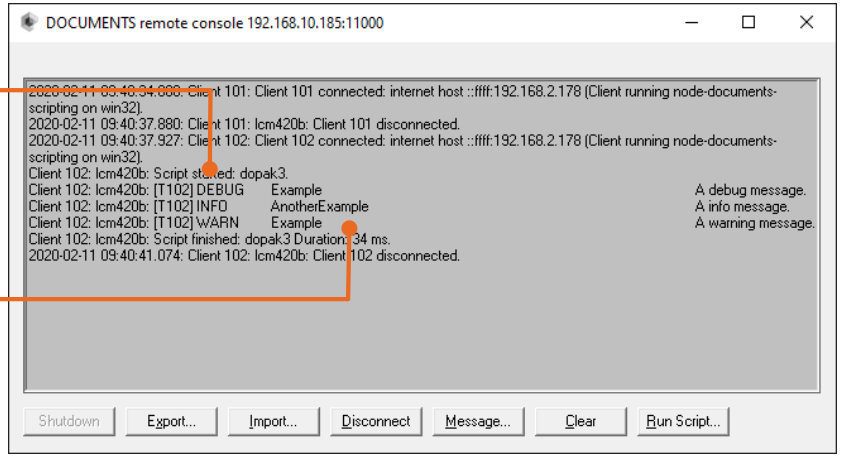
```
context.enableModules();
var otrLogger = require("otrLogger");

otrLogger.logDebug("Example", "A debug message");
otrLogger.logInfo("AnotherExample", "An info message");
otrLogger.logWarn("Example", "A warning message");
```

Level einer Log-Meldung

- logDebug(), logWarn() ,logInfo(), logError(), logFatal()

Kontext der Log-Meldung

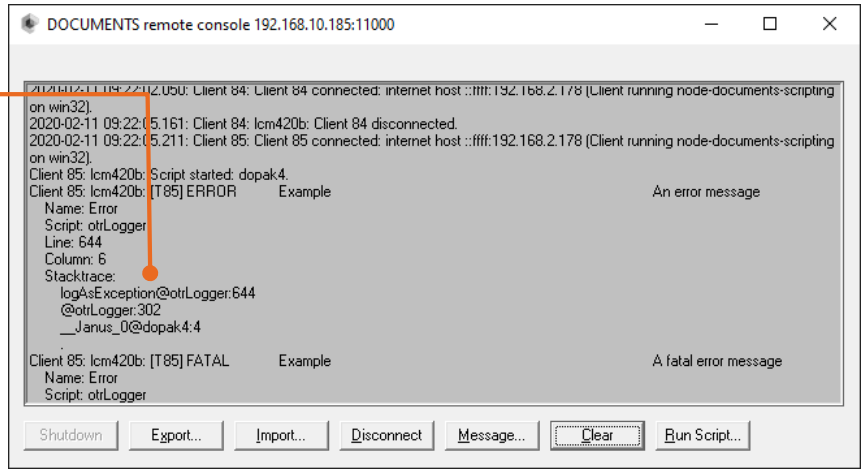


Ausgabe von Fehlermeldungen

```
context.enableModules();
var otrLogger = require("otrLogger");

otrLogger.logError("Example", "An error message");
otrLogger.logFatal("Example", "A fatal error message");
```

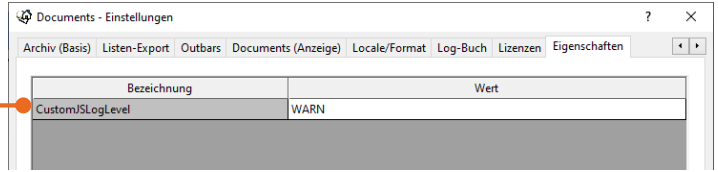
- Automatische Ergänzung des Stacktrace
- Exception als Log-Meldung
- Fatale Fehlermeldungen werden gecacht

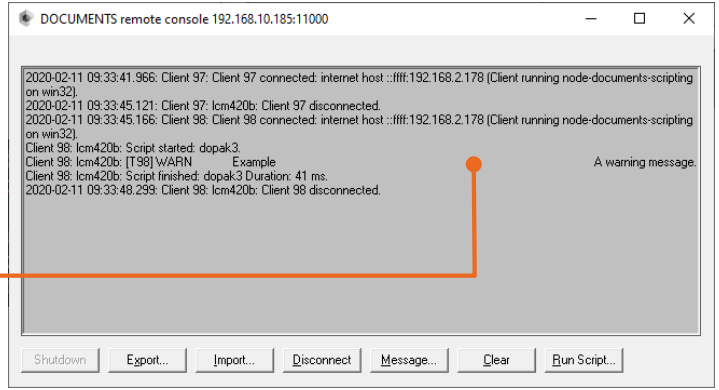


Filtern von Log-Meldungen

```
context.enableModules();
var otrLogger = require("otrLogger");

otrLogger.logDebug("Example", "A debug message");
otrLogger.logInfo("AnotherExample", "An info message");
otrLogger.logWarn("Example", "A warning message");
```





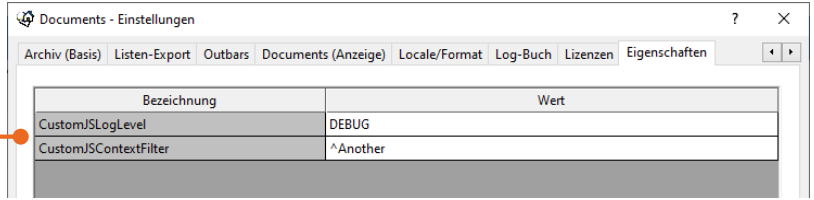
Filtern per Log-Level

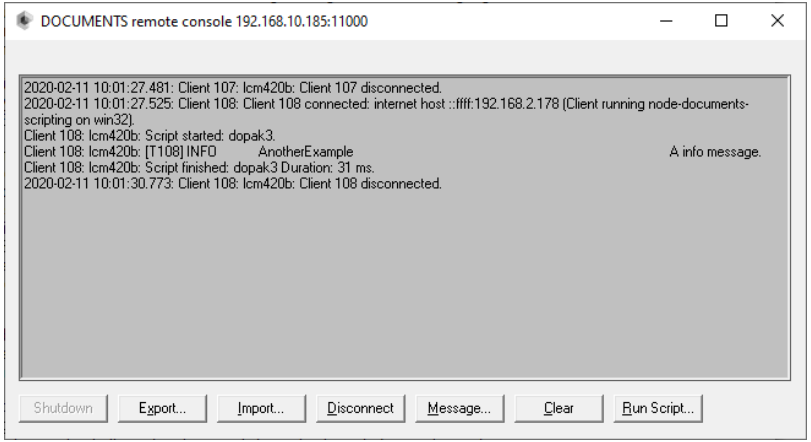
- Per $CustomJSLogLevel
- Mindest-Level
  DEBUG > INFO > WARN >
  ERROR > FATAL > NONE

Filtern von Log-Meldungen

```
context.enableModules();
var otrLogger = require("otrLogger");

otrLogger.logDebug("Example", "A debug message");
otrLogger.logInfo("AnotherExample", "An info message");
otrLogger.logWarn("Example", "A warning message");
```





Filtern per Log-Kontext

- Per $CustomJSContextFilter
- Regulärer Ausdruck auf den Kontext einer Log-Meldung

Stacktrace in Fehlermeldungen

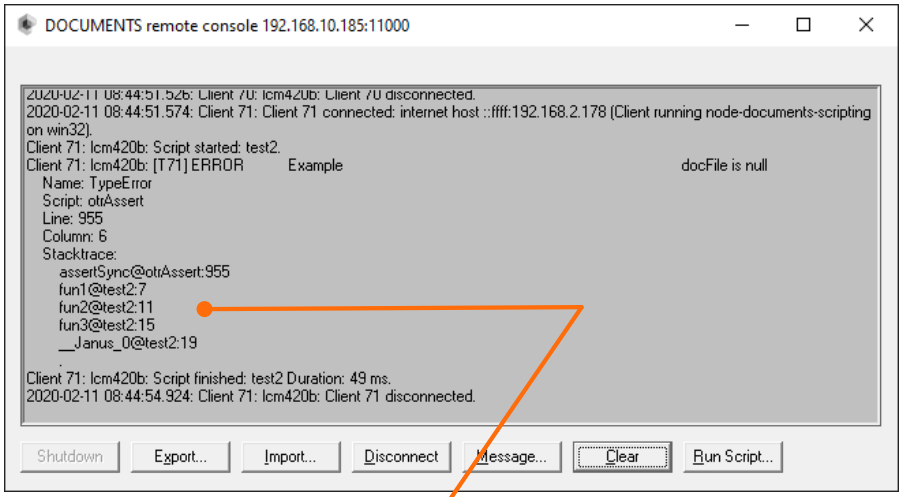
```
context.enableModules();
var otrLogger = require("otrLogger");

function fun1(file) {
   var otrAssert = require("otrAssert");
   // [...] some code [...]
   otrAssert.assertSync(file, "fun1");
}

function fun2(file) {
   fun1(file);
}

function fun3(file) {
   fun2(file);
}

try {
   fun3(context.file);
} catch (e) {
   otrLogger.logError("Example", e);
   return e.message;
}
```



Stacktrace
Für alle Fehlermeldung und Exceptions als Log-Meldung

otrAssert: Erweiterte Fehlerbehandlung

Fehlerbehandlung im PortalScripting

- Auswertung der Fehlerflags
- Auswertung von getLastError()

Nachteile

- Komplexe Kontrollpfade
- Code wird unübersichtlich
- Code wird wartungsanfällig

```
var file = context.file;

if (!file.startEdit()) {
   return file.getLastError();
}

file.field = "New value";

// [...] more code [...]

if (!file.commit()) {
   var commitError = file.getLastError();

   if (!file.abort()) {
      return commitError + "\n" +
            file.getLastError();
   } else {
      return commitError;
   }
}
```

PortalScripting-API „mit“ Exceptions

Wrapper um Funktionen der API

- Automatische Auswertung der Fehlerflags
- Werfen von Exception

Vorteile

- Vereinfachung der Fehlerbehandlung
- Fehlerbehandlung mit try-catch-finally
- Ausgabe des Stacktrace in Kombination mit otrLogger

```
context.enableModules();
var otrAssert = require("otrAssert");
var otrLogger = require("otrLogger");
var file = context.file;


try {
   otrAssert.assertStartEdit(file, "Example");

   file.field = "New value";

   // [...] more code [...]

   otrAssert.assertCommit(file, "Example");
} catch (e) {
   if (file.getOriginal()) {
      otrAssert.assertAbort(file, "Example", e);
   }
   otrLogger.logError("Example", e);
}
```

PortalScripting-API „mit“ Exceptions

Vorteile (Fortsetzung)

- Bereinigung mit try-finally
- Für Erfolgs- und Fehlerfall
- Nutzerwechsel, geöffnete Dateien, Datenbankverbindungen

```
// [...]
var oldUser = context.currentUser;
// no evaluation of the return flag for
// the sake of simplicity
context.changeScriptUser(technicalUser);

try {

   file.field = "New value";

   // [...] more code [...]

   otrAssert.assertSync(file, "Example");
} finally {
   context.changeScriptUser(oldUser);
}
```

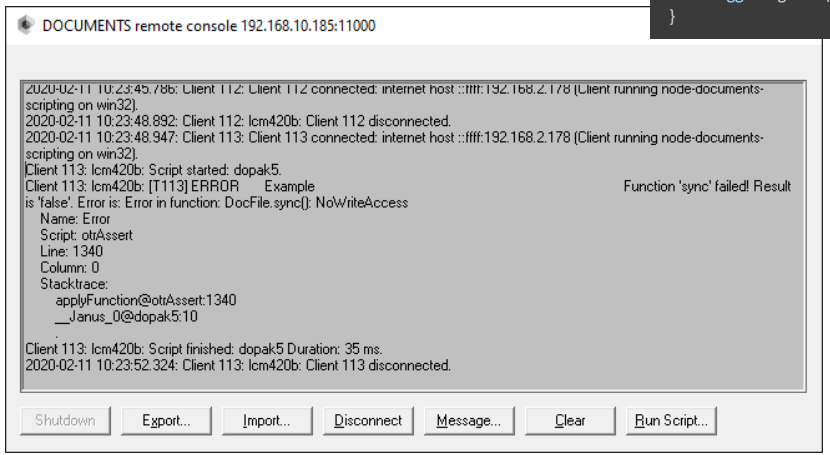
PortalScripting-API „mit“ Exceptions

Wrapper durch JS-Proxies

- Für beliebige Funktionen eines Objekts der PortalScripting-API
- Default-Listen für context und DocFile

```
context.enableModules();
var otrAssert = require("otrAssert");
var otrLogger = require("otrLogger");
var throwingFile = otrAssert.getThrowingObject(context.file, [
   "sync"
]);
// Alternative:
// var throwingFile = otrAssert.getThrowingObject(context.file);

try {
   throwingFile.field1 = "New value";
   throwingFile.sync();
} catch (e) {
   otrLogger.logError("Example", e);
}
```



otrLogger und otrAssert

- ✓ Erleichtern Fehlersuche und Fehlerbehandlung
- ✓ Fester Bestandteil von DOCUMENTS
- ✓ Typings und Dokumentation

VS Code

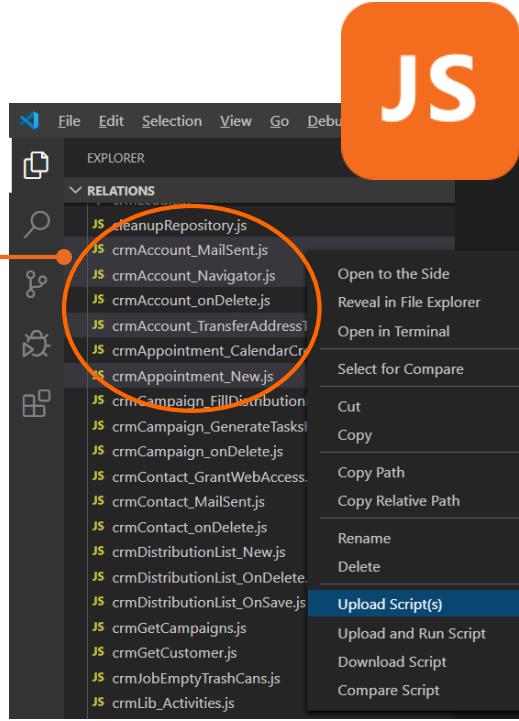
Inhalt

- Multiselect Upload
- XML-Export / XML-Import
- Intellisense
- Intellisense und require()

Neues im VS Code

Script-Upload: Multi-Selection

- Upload Script: mehrere Scripte im Explorer auswählen
- Upload Scripts from Folder: mehrere Ordner im Explorer auswählen



Scriptlibs

- Bearbeitung von Scripten in ScriptLibs -> kein autom. clear cache
- F1 → Clear PortalScript Cache

Wartungsoperationen

- Perform Maintenance Operation…

XML Export / Import… (ff)

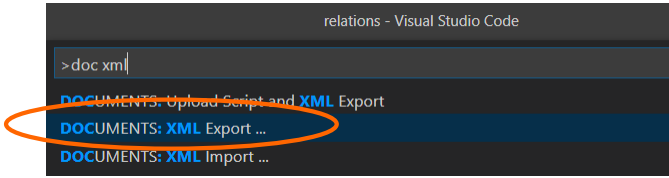
XML Export FileTypes

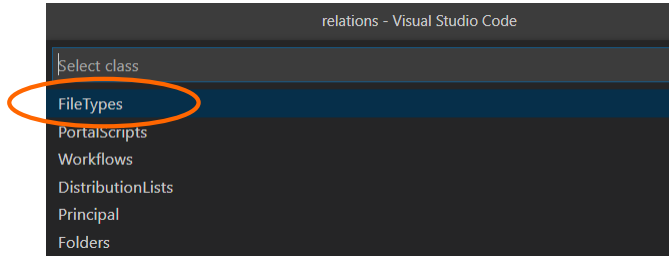
Wizard (Beispiel: Alle FileTypes)

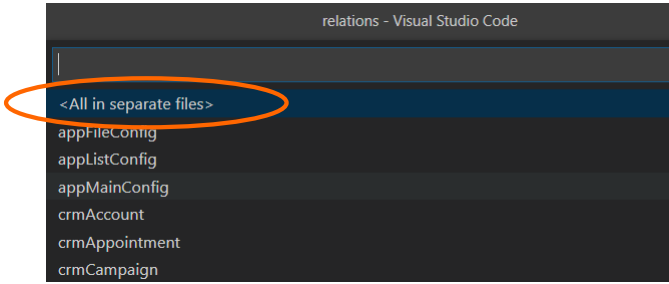
- F1 → XML Export (type: doc xml)
- Select class: FileTypes
- Auswahlliste: <All in separate files>
- Output Pfad setzen: Default lassen

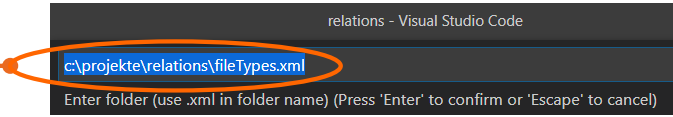
XML Output

- Ordner mit Endung .xml → Kontextmenü… (ff)



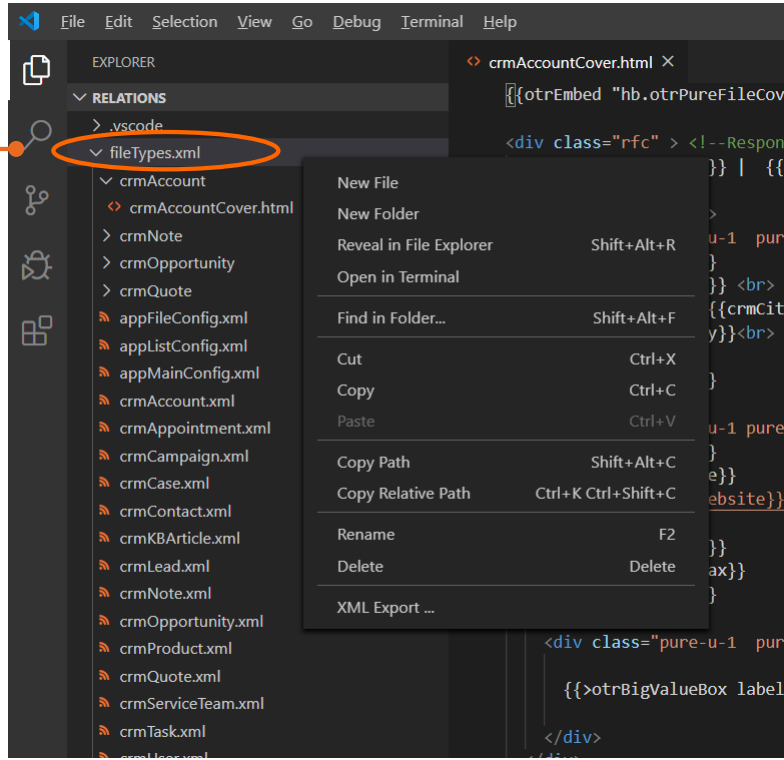






XML Export / Import

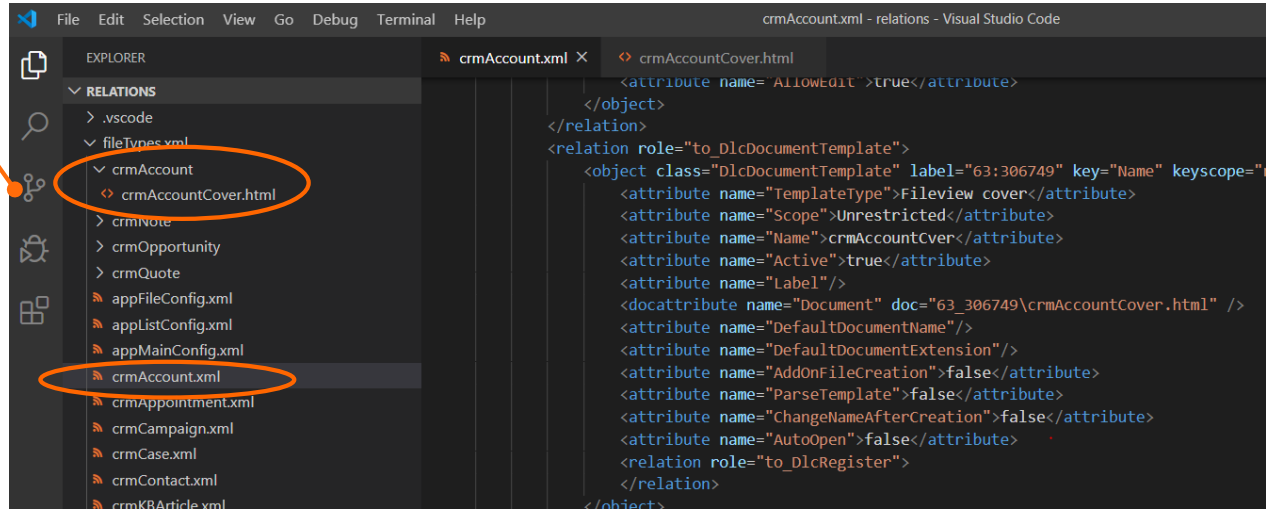
- Kontextmenü:
  - Ordner mit Endung .xml → XML Export
  - Datei mit Endung .xml → XML Import
- Motivation
  - Dokumentvorlagen im VS Code bearbeiten
  - Mappentyp und Dokumente in Git versionieren



Struktur der Dokumentvorlagen

- Ordner-Name = FileType-Name (z.B. crmAccount)
  - (Anders als XML Export im Manager)
- Dokument-Name (crmAccountCover.html) wird beim XML-Import nicht geändert
  - (Anders als XML Import im Manager)

Empfehlung:
Versionierung in Git
(einfach über VS Code)

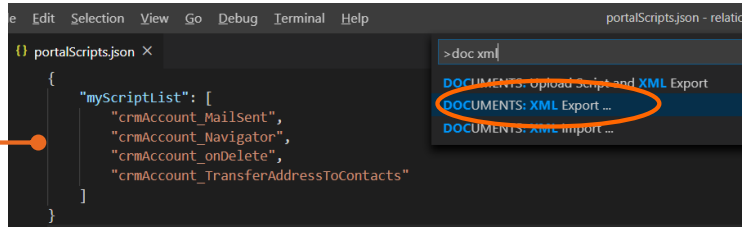


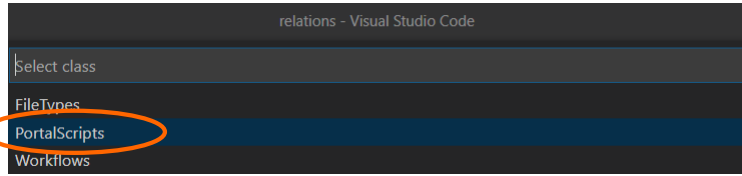
XML Export PortalScripte

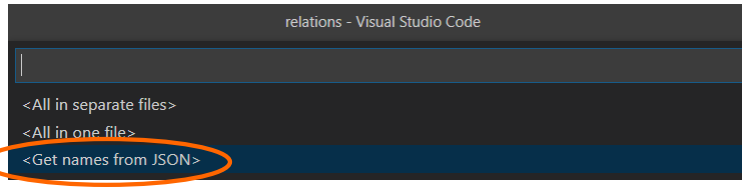
Wizard (Beispiel: JSON Script Liste)

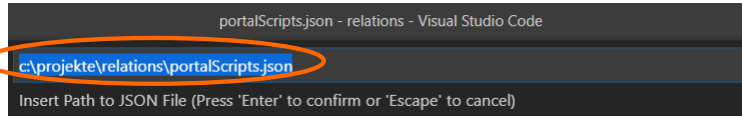
- JSON Datei im Editor: Liste mit Script-Namen
- F1 → XML Export
- Select Class: PortalScripts
- Auswahlliste: <Get names from JSON>
- JSON Datei: Default lassen
- Output Pfad setzen: Default lassen

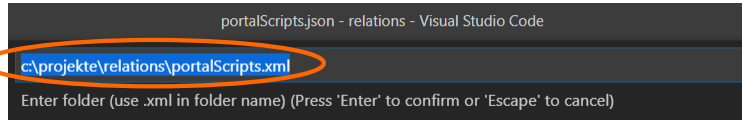
- XML Output:
  - Alle Scripte aus der JSON Liste in einer XML Datei





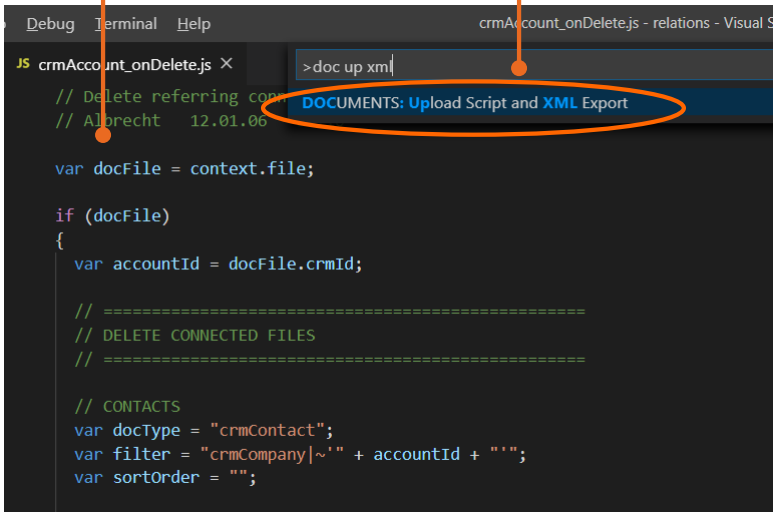


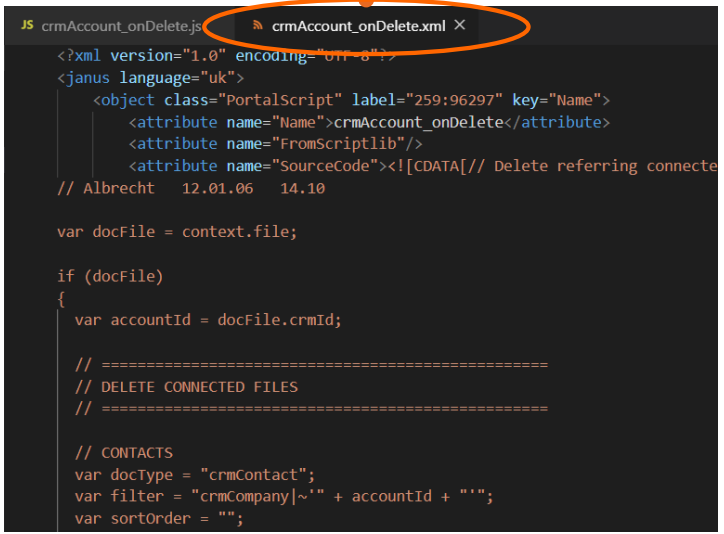




Utilityfunktion für PortalScripte

- Script im Editor geöffnet → F1 → Upload and Export XML

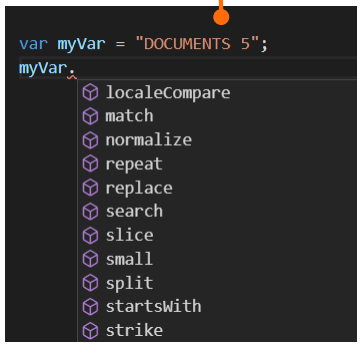




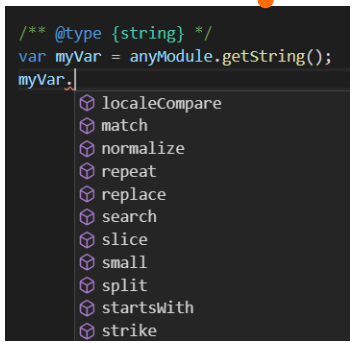
IntelliSense und JSDoc (Wdh.)

- Typ einer Variable/Funktion in JSDoc Kommentar angeben
  - → IntelliSense an der Variable/Parameter für den angegebenen Typ
- Tipp: über einer Funktion `/** Enter` eingeben → alle `@param` automatisch

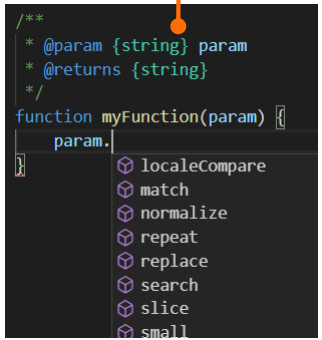
Typ durch Definition klar



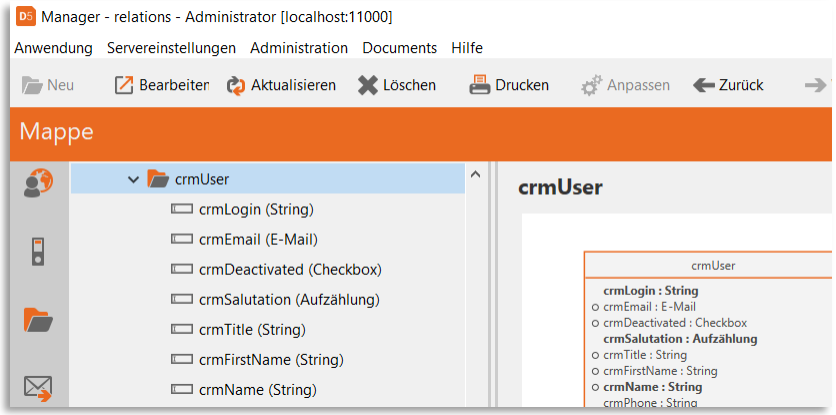
Falls Return-Type der Funktion nicht klar
→ Typ mit JSDoc angeben



Typen für Parameter mit JSDoc angeben



IntelliSense für Mappentypen (Wdh.)



`context.createFile` → Typ `crmUser`

```
var myDocFile1 = context.createFile("crmUser");
myDocFile1.
```
- crmDeactivated
- crmEmail
- crmFax
- crmFirstName
- crmLogin
- crmMobilePhone
- crmName

`context.file` → Typ `DocFile`

```
var myDocFile2 = context.file;
myDocFile2.
```
- abort
- addDocumentFromFileSystem
- addPDF
- archive
- archiveAndDelete
- asJSON
- cancelWorkflow

`context.file` mit JSDoc → Typ `crmUser`

```
/** @type {crmUser} */
var myDocFile2 = context.file;
myDocFile2.
```
- crmDeactivated
- crmEmail
- crmFax
- crmFirstName
- crmLogin
- crmMobilePhone

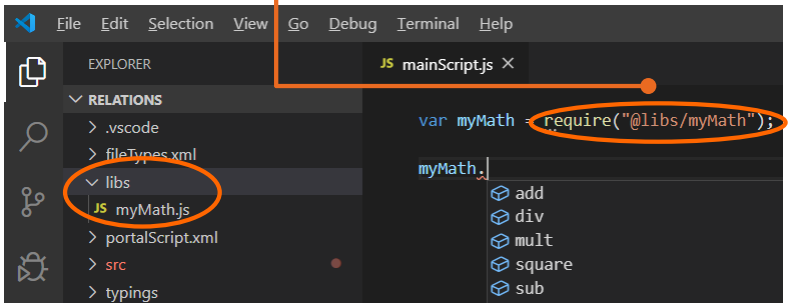
`crmUser.crmName` → Typ `String`

```
/** @type {crmUser} */
var myDocFile2 = context.file;
var myName = myDocFile2.crmName;
myName.
```
- charAt
- charCodeAt
- codePointAt
- concat
- endsWith

Tipps für VS Code

IntelliSense und require (Beispiel):

- myMath.js aus scriptlibs wird im PortalScript verwendet
- myMath.js liegt im Projekt in ./libs/
- IntelliSense für myMath im PortalScript:
  - In jsconfig.json → compilerOptions: Alias für Pfad zu libs setzen
  - Im PortalScript: Alias in require verwenden
- DOCUMENTS entfernt alle Präfixe aus require



Tipps für VS Code

GitLens: VS Code Extension für Git

- Umfangreiche Informationen in eigener Side Bar
- Informationen der Git Commits im Editor anzeigen
  - Committer, Zeitstempel und Message
  - Wird für die markierte Zeile angezeigt