ELO Sync

Beispiele

Inhaltsverzeichnis

Anmeldung an einer Desktop-Anwendung

Der nachfolgende C#-Code zeigt beispielhaft, wie der Zugriffstoken für ELO Sync bei einer .NET Core Desktopanwendung ermittelt werden kann:

```
static async Task<string> SignInUserAndGetTokenUsingMSAL(string[] scopes)
{
   string authority = $"<instance>/<tenant-id>"; // z.B. https://login.microsoftonline.c

   // Initialisierung der MSAL library
   IPublicClientApplication application = PublicClientApplicationBuilder.Create("<own-cl
                                          .WithAuthority(authority)
                                          .WithDefaultRedirectUri()
                                          .Build();


   AuthenticationResult result;
   try
   {
      // Wenn bereits eine Anmeldung durchgeführt wurde, dann wird diese wiederverwendet
      var accounts = await application.GetAccountsAsync();
      result = await application.AcquireTokenSilent(scopes, accounts.FirstOrDefault())
         .ExecuteAsync();
   }
   catch (MsalUiRequiredException ex)
   {
      // Wenn noch keine Anmeldung durchgeführt wurde, wird ein Browserfenster geöffnet,
      result = await application.AcquireTokenInteractive(scopes)
         .WithClaims(ex.Claims)
         .ExecuteAsync();
   }


   return result.AccessToken;
}

string[] scopes = new string[] { "<elosync-client-id>/.default" };

string bearerToken = await SignInUserAndGetTokenUsingMSAL(scopes);
```

Danach gibt die Anmeldung ein Zugriffstoken für die Anwendung zurück, das bei jeder Anfrage an ELO Sync verwendet werden muss, indem es im HTTP-Header `Authorization` als `Bearer` gesendet wird.

Für weitere Informationen siehe Microsoft identity platform code samples for authentication and authorization

Erstellen eines Syncjobs

Hier werden Beispiele dargestellt, wie unterschiedliche Syncjobtypen erstellt werden können. Wichtig ist hierbei, dass mindestens der Name des Syncjobs (name), die Synchronisierungsrichtung (syncDirection) sowie das Synchronisationsziel (syncTargetSystem) angegeben werden müssen.

Für den Namen kann ein Wert frei ausgesucht werden, für die Synchronisierungsrichtung ist einer der folgenden Werte erlaubt:

syncDirection:

- "ToElo"
- "ToThirdSystem"
- "TwoWay"

Als Synchronisationsziel werden aktuell die folgenden Werte unterstützt:

syncTargetSystem:

- "Sharepoint"
- "OneDrive"

Veröffentlichungsauftrag

Nachfolgendes Beispiel zeigt das Anlegen eines Veröffentlichungsauftrags, bei dem Dokumente und Ordner von ELO nach Sharepoint übertragen werden. In der Beispiel-Request sind alle für die Ausführung benötigten Pflichtfelder gesetzt.

Weitere optionale Felder können noch angegeben werden.

Eine Übersicht über den Aufbau eines Syncjob-Objektes mit allen Feldern findet sich hier: SyncJobEntity

Falls Werte fehlen, die zur Ausführung benötigt werden, wie z.B. die Synchronisierungs Quell- und Zielpfade, können diese auch nachträglich per Update hinzugefügt bzw. gefüllt werden.

Dies wird in dem Beispiel Aktualisieren eines Syncjobs näher erklärt.

> **Information**
>
> Bei Veröffentlichungsaufträgen werden keine Maskenangaben benötigt, da die Drittsysteme Metadaten unterschiedlich handhaben. Sollen die Metadaten trotzdem übernommen werden, kann dies über die Metadatenzuordnung spezifiziert werden.

```
POST /odata/syncjobs?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "name":"Publish ELO -> Sharepoint",
```

```
    "syncDirection": "ToElo",
    "syncTargetSystem": "Sharepoint",
    "eloSystemSyncTarget":{
        "folderName":"<elo-folder-name>",
        "folderId":"<elo-folder-id-int>"
    },
    "sharepointSyncTarget":{
        "listId":"<sharepoint-list-id>",
        "siteId":"<sharepoint-site-id>",
        "name":"<sharepoint-target-display-name>"
    }
}
```

Archivierungsauftrag

In nachfolgendem Beispiel wird ein Archivierungsauftrag erstellt, welcher den Inhalt eines SharePoint Online-Ordners in einen ELO-Ordner archiviert. In der Beispiel-Request sind alle für die Ausführung benötigten Pflichtfelder gesetzt.

Weitere optionale Felder können noch angegeben werden.

Eine Übersicht über den Aufbau eines Syncjob-Objektes mit allen Feldern findet sich hier: SyncJobEntity

```
POST /odata/syncjobs?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "name":"New SyncJob",
    "syncDirection": "ToElo",
    "syncTargetSystem": "Sharepoint",
    "masks":[
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"DocMask"
        },
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"FolderMask"
        }
    ],
    "eloSystemSyncTarget":{
        "folderName":"<elo-folder-name>",
        "folderId":"<elo-folder-id-int>"
    },
    "sharepointSyncTarget":{
```

```
        "listId":"<sharepoint-list-id>",
        "siteId":"<sharepoint-site-id>",
        "name":"<sharepoint-target-display-name>"
    }
}
```

Synchronisierungsauftrag

In nachfolgendem Beispiel wird gezeigt, wie ein Synchronisierungsaufrag mit allen für die Ausführung benötigten Pflichtfeldern direkt angelegt werden kann.

Weitere optionale Felder können noch angegeben werden.

Eine Übersicht über den Aufbau eines Syncjob-Objektes mit allen Feldern findet sich hier: SyncJobEntity

```
POST /odata/syncjobs?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "name":"Sync ELO <-> Sharepoint",
    "syncDirection": "TwoWay",
    "syncTargetSystem": "Sharepoint",
    "masks":[
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"DocMask"
        },
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"FolderMask"
        }
    ],
    "eloSystemSyncTarget":{
        "folderName":"<elo-folder-name>",
        "folderId":"<elo-folder-id-int>"
    },
    "sharepointSyncTarget":{
        "listId":"<sharepoint-list-id>",
        "siteId":"<sharepoint-site-id>",
        "name":"<sharepoint-target-display-name>"
    }
}
```

Auftrag mit allen Eigenschaften

Je nach Auftragstyp können unterschiedliche Eigenschaften angegeben werden. Eigenschaften, die für den jeweiligen Auftragstyp nicht existieren, werden ignoriert.

Nachfolgendes Beispiel zeigt, welche Eigenschaften bei einem Syncjob generell angegeben werden können:

```
POST /odata/syncjobs?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "name":"New SyncJob",
    "syncDirection": "ToElo",
    "syncTargetSystem": "Sharepoint",
    "isActive":true,
    "masks":[
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"DocMask"
        },
        {
            "eloMaskId":"<mask-id-int>",
            "name":"<mask-display-name>",
            "type":"FolderMask"
        }
    ],
    "eloSystemSyncTarget":{
        "folderName":"<elo-folder-name>",
        "folderId":"<elo-folder-id-int>"
    },
    "settings":[
        {
            "category":"Approvals",
            "name":"MaxSyncDocumentSize",
            "value":"5242880", // Example: 5MB in Byte
            "hierarchyContext":"SyncJob"
        },
        {
            "category":"Approvals",
            "name":"MaxSyncableDocuments",
            "value":"50", // Example: 50 Documents
            "hierarchyContext":"SyncJob"
        },
        {
            "name":"ShMetadataMappings",
            "category":"MetadataMappings",
```

```
            "value":"{\"eloField\":\"Ordner\\\\ELOINDEX\",\"thirdSystemField\":\"3\\\\Sh
            "hierarchyContext":"SyncJob"
        },
        {
            "name":"ShOnlineMetadataExportType",
            "value":"AsJson",
            "hierarchyContext":"SyncJob"
        },
        {
            "name": "MaxFolderReferencesDepthPublish",
            "value": "50",
            "hierarchyContext": "SyncJob"
        },
        {
            "name": "ShOnlineArchivedEntriesReplacementMode",
            "value": "ReplaceByUrlFile",
            "hierarchyContext": "SyncJob"
        },
        {
            "name": "SyncConflictResolutionStrategy",
            "value": "1",
            "hierarchyContext": "SyncJob"
        }
    ],
    "sharepointSyncTarget":{
        "listId":"<sharepoint-list-id>",
        "siteId":"<sharepoint-site-id>",
        "name":"<sharepoint-target-display-name>"
    },
    "timeTrigger":{
        "timePeriod":"Weekly",
        "timeZoneId":"Europe/Berlin",
        "cronJobFormatValue":"* 00 22 * * 1",
        "isActive":true
    }
}
```

Syncjob aktualisieren

Um einen Syncjob zu aktualisieren, ist es ausreichend, wenn nur die Eigenschaften angegeben werden, die sich geändert haben. Die nicht angegebenen Eigenschaften bleiben dann einfach unverändert erhalten.

> **Beachten Sie**
>
> Das Zielsystem (`syncTargetSystem`) sowie die Synchronisierungsrichtung (`syncDirection`) lassen sich nach der Erstellung eines Auftrags nicht mehr ändern!
>
> Wenn das Zielsystem oder die Auftragsart/Synchronisierungsrichtung geändert werden soll, muss ein neuer Auftrag angelegt werden und der bereits existierende Auftrag gelöscht werden.

Bei der Anfrage zur Änderung eines Syncjobs muss dabei die ID des Jobs sowohl im Pfad als auch im Body unter `syncJobId` angegeben werden.

Nachfolgend wird beispielhaft der Name eines Syncjobs geändert:

```
PUT /odata/syncjobs(<syncjobid>)?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "name":"New Syncjobname",
    "syncJobId":<syncjobid>
}
```

Syncjob löschen

Ein Syncjob kann gelöscht werden, indem der folgende Endpunkt mit der Syncjob-ID aufgerufen wird:

```
DELETE  /odata/syncjobs(<syncjobid>)?repository=repo1
Header: Authorization: <bearer_token>
```

Syncjob ausführen

Ein Syncjob kann manuell ausgeführt werden, indem der folgende Endpunkt mit der Syncjob-ID aufgerufen wird:

```
POST  /odata/syncjobs/run(<syncjobid>)?repository=repo1
Header: Authorization: <bearer_token>
```

Die Request startet dabei den Syncjob nur, es wird nicht gewartet, bis die Ausführung abgeschlossen ist.

Um auf den Abschluss eines Jobs zu warten kann SignalR verwendet werden.

Laufenden Syncjob abbrechen

Ein aktuell laufender Syncjob kann über den cancel-Endpunkt abgebrochen werden:

```
POST  /odata/syncjobs/cancel(<syncjobid>)?repository=repo1
Header: Authorization: <bearer_token>
```

Syncjob aktivieren/deaktivieren

Ein Syncjob kann aktiviert oder deaktiviert werden, indem einer der entsprechenden Endpunkte aufgerufen wird. Dadurch wird bestimmt, ob der Syncjob durch den konfigurierten Zeit-Trigger automatisch ausgeführt werden soll.

Syncjob aktivieren

```
POST /odata/syncjobs/activate(<syncjobid>)?repository=repo1
Header: Authorization: <bearer_token>
```

Syncjob deaktivieren

```
POST /odata/syncjobs/deactivate(<syncjobid>)?repository=repo1
Header: Authorization: <bearer_token>
```

Alternative: Syncjob bearbeiten

Alternativ kann der Zustand des Syncjobs auch per Aktualisierung des Syncjobs gesetzt werden. Dazu muss die Eigenschaft `"isActive"` gesetzt werden, erlaubt sind hier die Werte `true` oder `false`.

Sync-Konflikt lösen

Um einen Konflikt zu lösen, wird die ID benötigt sowie die Lösung für diesen Konflikt.

Die Lösung besteht aus der ID des Elements, dessen Änderungen übertragen werden sollen.

Zurzeit ist es nicht möglich eine Kombination der Änderungen aus beiden Elementen zu übertragen, dies muss manuell gemacht werden und dann der Syncjob erneut ausgeführt werden.

Sync-Konflikte abfragen

Um herauszufinden, ob es Konflikte gibt und wenn ja, welche Elemente Konflikte verursachen, können die Konflikte über folgenden Endpunkt abgefragt werden:

```
GET  /odata/syncconflicts?repository=repo1
Header: Authorization: <bearer_token>
```

Dieser liefert die aufgetretenen Sync-Konflikte zurück.

> **Information**
>
> Über die Eigenschaft `"firstSystem"` und `"secondSystem"` lässt sich ermitteln, zu welchem System die Elemente gehören. Die `"firstId"` und die `"secondId"` sind die jeweiligen IDs der Konflikt-Items.

Lösen von Konflikten

Nachdem ermittelt wurde, welches Item zu welchem System gehört, können die Konflikte gelöst werden. Dazu muss als `"Resolution"` die entsprechende itemId (*firstId* oder *secondId* des Konflikts) angegeben werden.

Nachfolgend wird ein Beispiel für die Lösung zweier Konflikte gezeigt, bei denen einmal eine ELO-Version und einmal eine Sharepoint-Online-Version genommen wird.

Die IDs müssen entsprechend mit den IDs des SyncItems übereinstimmen, diese lässt sich wie oben beschrieben aus dem eigentlichen Konflikt ermitteln.

```
POST  /odata/syncconflicts/resolve?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

[
    {
        "SyncConflictId": 1,
        "Resolution": "<elo-itemId>"
    },
    {
        "SyncConflictId": 4,
```

```
        "Resolution": "<sp-itemId>"
    }
]
```

Freigabe bearbeiten

Um eine Freigabe zu lösen, muss die entsprechende ID und der neue Freigabestatus angegeben werden. Zusätzlich muss der Status der Freigabe angegeben werden. Die gültigen Werte lassen sich von ApprovalStatus entnehmen.

Aktualisieren einer Freigabe

Eine einzelne Freigabe kann unter Angabe der Approval-ID bearbeitet werden, indem der Status gesetzt wird:

```
PUT  /odata/approvals(<approval-id>)?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

{
    "approvalId": "<approval-id>",
    "status": "ApprovedOnce"
}
```

Aktualisieren von mehreren Freigaben auf einmal

Es können auch mehrere Freigaben auf einmal bearbeitet werden. Dazu kann der folgende Endpunkt verwendet werden:

```
POST  /odata/approvals/batchupdate?repository=repo1
Content-Type: application/json
Header: Authorization: <bearer_token>

[
    {
        "id": "<approval-id-1>",
        "changes": {
            "approvalId": "<approval-id-1>",
            "status": "ApprovedOnce"
        }
    },
    {
        "id": "<approval-id-2>",
        "changes": {
            "approvalId": "<approval-id-2>",
            "status": "Approved"
        }
    },
    {
        "id": "<approval-id-3>",
        "changes": {
```

```
            "approvalId": "<approval-id-3>",
            "status": "Declined"
        }
    }
]
```

Synchronisationsprotokoll auslesen

In dem Synchronisationsprotokoll wird festgehalten, was in einem Jobdurchlauf genau synchronisiert wurde.

Protokoll für den letzten Durchlauf eines Syncjobs auslesen

Um das Synchronisationsprotokoll des letzten Durchlaufs eines Syncjobs zu ermitteln, kann der folgende Endpunkt verwendet werden. In diesem Beispiel werden zusätzlich noch die Synchronisationsdetails in die Ergebnisse mit aufgenommen, um einen genaueren Überblick über die Synchronisation zu erhalten.

```
GET  /odata/synchronizationactivities/latest(syncJobId=<syncjobid>)?repository=repo1&$ex
Header: Authorization: <bearer_token>
```

Verwendung von SignalR

ELO Sync verwendet SignalR, um Events wie z.B: das Ausführen und Bearbeiten von Syncjobs, das Bearbeiten von Freigaben oder auch das Lösen von Konflikten an Clients mitzuteilen. Durch die Implementierung von SignalR kann eine Client-Anwendung einfach auf die von ELO Sync ausgelösten Events reagieren.

Unter folgendem Link findet sich eine Einführung in SignalR:

Was ist SignalR?

SignalR-Events in ELO Sync

ELO Sync stellt die folgenden SignalR-Events bereit:

SyncJobSyncStateChanged:

- Ereignis, das über die Änderung des Syncjob Synchronisierungszustands informiert. In diesem Fall z.B., ob dieser gerade läuft, wartet, erfolgreich durchgelaufen ist oder ob er mit Konflikten oder Fehlern durchgelaufen oder abgebrochen ist.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.

SyncJobUpdated:

- Ereignis, das über geänderte SyncJobs informiert.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.

SyncJobAdded:

- Ereignis, das über hinzugefügte SyncJobs informiert.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.

SyncJobRemoved:

- Ereignis, das über entfernte SyncJobs informiert.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.

SyncJobModeChanged:

- Ereignis, das über die Änderung des Syncjob Zustands informiert. In diesem Fall, ob dieser aktiv oder inaktiv ist.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.

ApprovalStatusChanged:

- Ereignis, das über die Änderung des Freigabezustands informiert.
- Es wird ein Approval-Objekt mitgeliefert, welches die Eigenschaften aus dem ApprovalEntity besitzt.

ConflictResolved:

- Ereignis, das über die Lösung eines Konflikts informiert.
- Es wird ein SyncConflictEntity-Objekt des entsprechenden gelösten Konflikts mitgeliefert,

SyncJobFinished:

- Ereignis, das über das Beenden eines SyncJobs informiert.
- Es wird ein SyncJob-Objekt mitgeliefert, in dem die Eigenschaften SyncJobId, die OwnerId, die EloOwnerId, der RepositoryKey, möglicherweise eine Fehlermeldung Error, der SyncState, IsActive, sowie das SyncTargetSystem vorhanden sind.