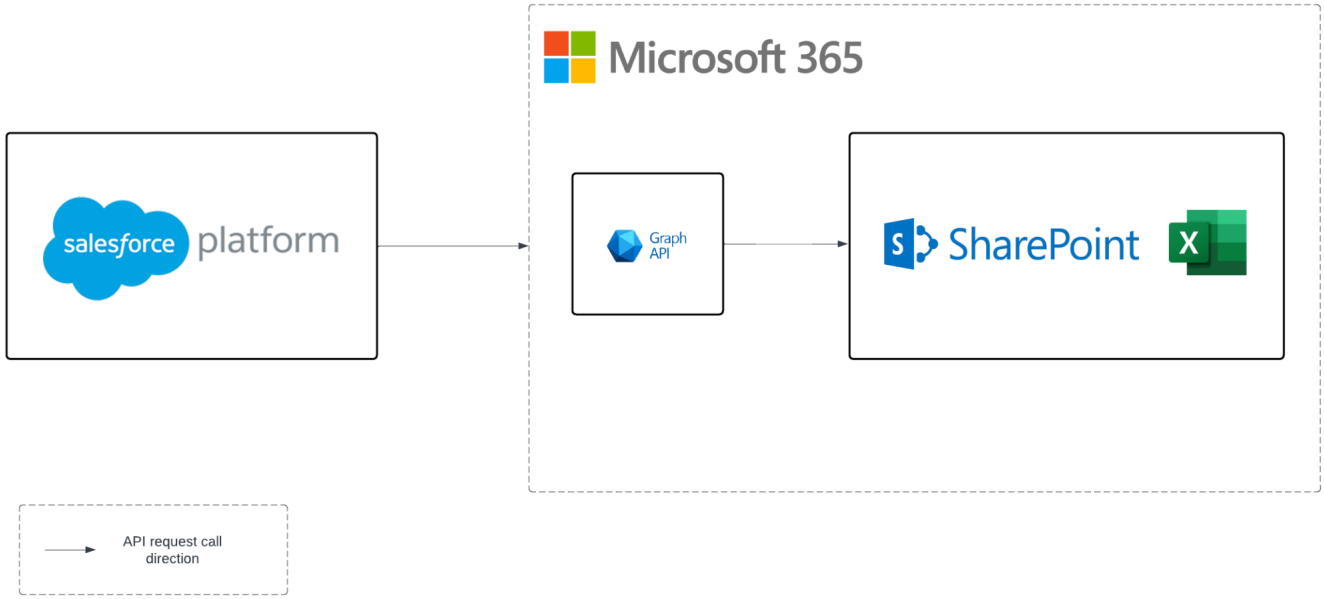
High-level Architecture Diagram



Item	Description
Salesforce Platform	Salesforce Platform is the app development platform that extends your CRM's reach and functionality. You do not have to be a developer to build apps using the Salesforce Platform.
Graph API	Microsoft Graph is a RESTful web API that enables you to access Microsoft Cloud service resources.
Sharepoint	Organizations use Microsoft SharePoint to create websites. You can use it as a secure place to store, organize, share, and access information from any device.

Authentication method

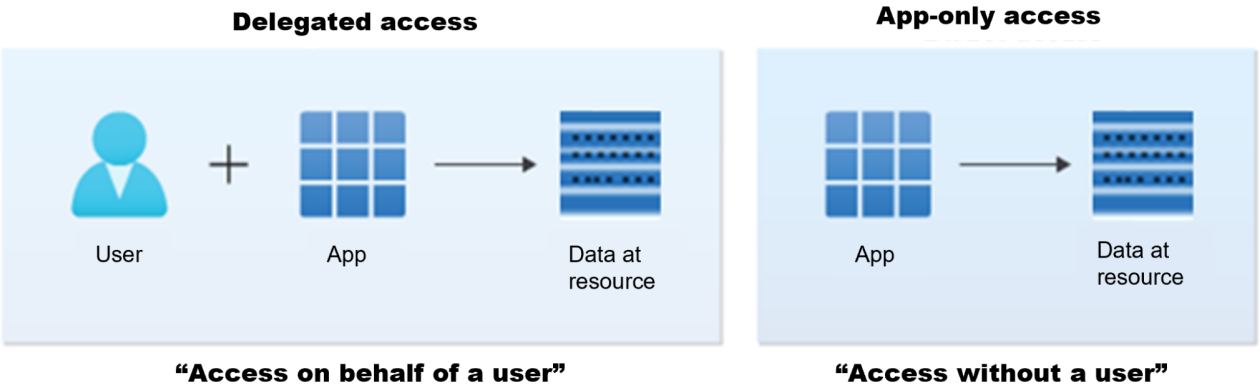
To get an access token, the application must be registered with the Microsoft Identity platform and be granted Microsoft Graph permissions. Registration integrates the application with the Microsoft platform and establishes the information that it uses to get tokens, including:

- **Application ID** - a unique ID assigned by the Microsoft identity platform;

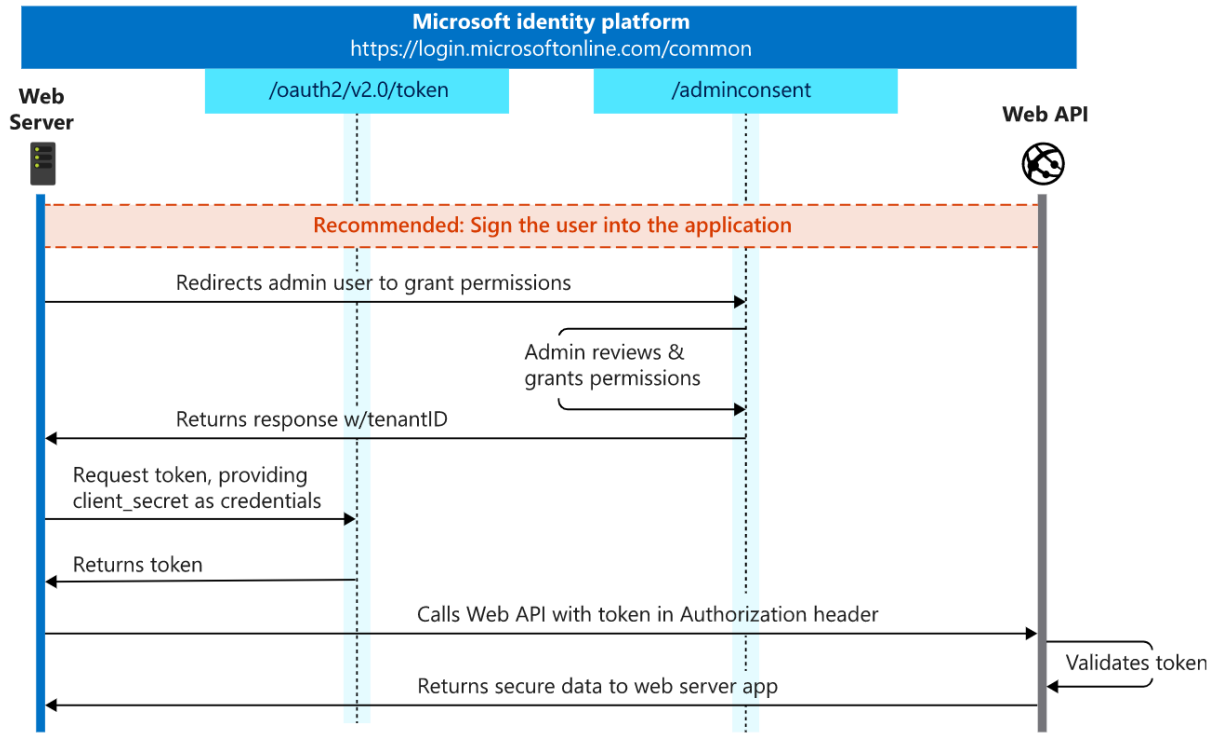
- **Redirect URL** - one or more endpoints at which the application will receive responses from the Microsoft identity platform;
- **Client Secret** - a password or a public/private key that the application uses to authenticate with the Microsoft identity platform.

There 2 methods that the application can use to authenticate with the Microsoft identity platform:

- **Delegated access** - the application acting on behalf of a signed-in user
- **App-only access** - the application acting with its own identity



In our case, the “**SharePoint Data Sync Manager**” solution uses an **app-only** authentication method which gets access tokens from Azure AD by using the OAuth 2.0 client credentials grant flow:



To summarize, the following are main steps to configure a service and get a token from the Microsoft identity platform endpoint using an app-only authentication method:

1. Register an application
2. Configure permissions for Microsoft Graph on the application
3. Get administrator consent
4. Gen an access token
5. Use the access token to call Microsoft Graph

In the OAuth 2.0 client credentials grant flow, the **application ID** and **client secret** values are used to request an access token directly from the Microsoft identity platform **/token** endpoint. Pre-configured permissions are specified by passing https://graph.microsoft.com/.default as the value for the **scope** parameter in the token request.

The following is an example of a token request:

```
POST https://login.microsoftonline.com/{tenant}/oauth2/v2.0/token HTTP/1.1
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id=535fb089-9ff3-47b6-9bfb-4f1264799865
&scope=https%3A%2F%2Fgraph.microsoft.com%2F.default
&client_secret=qWgdYAmab0YSkuL1qKv5bPX
&grant_type=client_credentials
```

A successful token response looks like this:

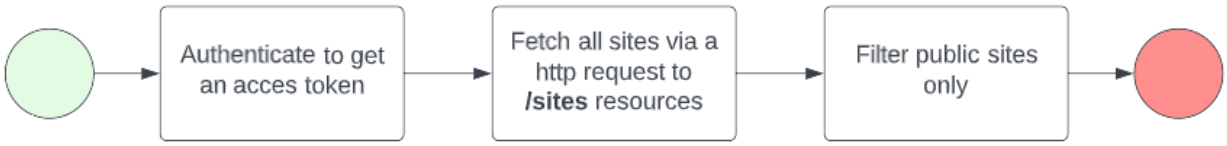
```
{
  "token_type": "Bearer",
  "expires_in": 3599,
  "ext_expires_in":3599,
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik1uQ19WWmNBVGZNNXBP..."
}
```

After obtaining the access token, it is then used to call Microsoft Graph by including the token in the **Authorization** header of a request:

```
GET https://graph.microsoft.com/v1.0/sites
Authorization: Bearer eyJ0eXAiO ... 0X2tnSQLEANnSPHY0gKcgw
Host: graph.microsoft.com
```

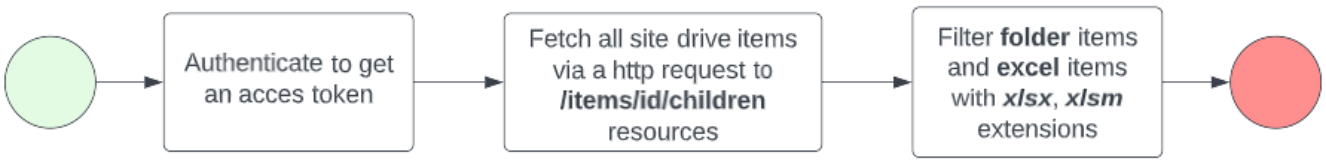
Data Flow

- **Fetch public SharePoint sites**

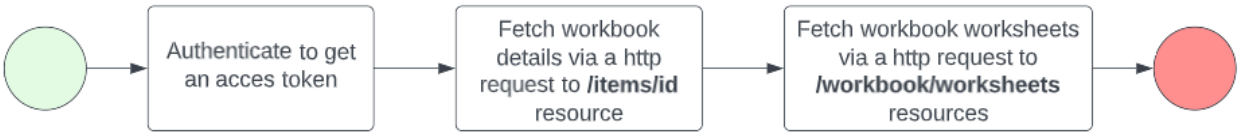


- **Fetch SharePoint site drives**

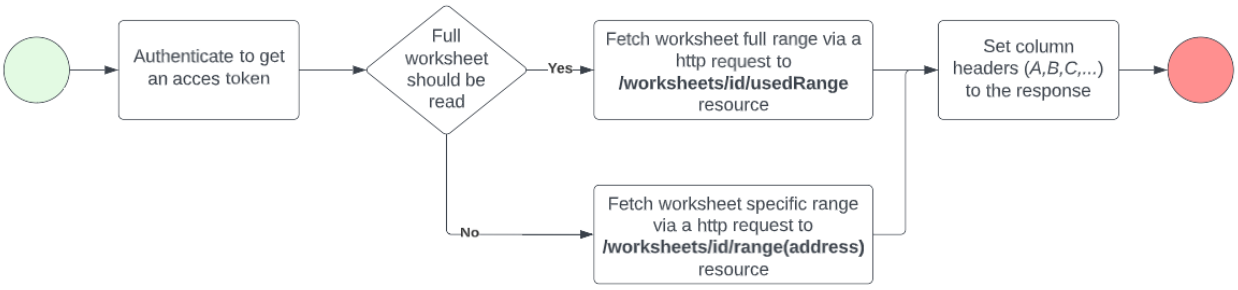
- **Fetch SharePoint site drive items**



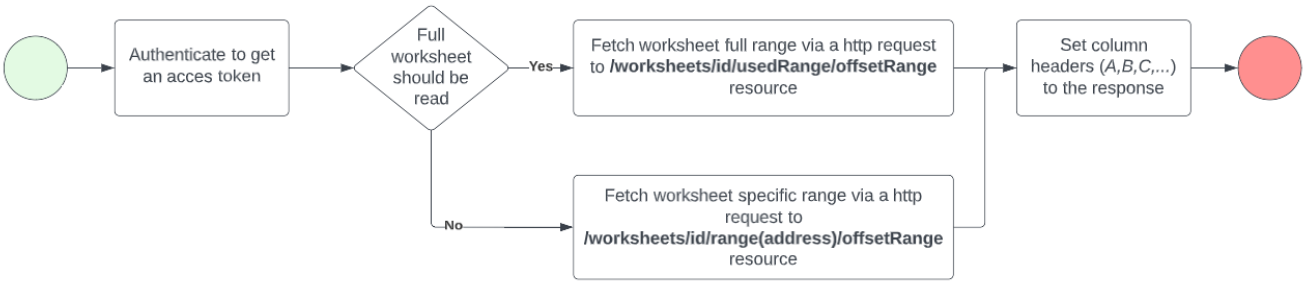
- **Fetch SharePoint excel workbook**



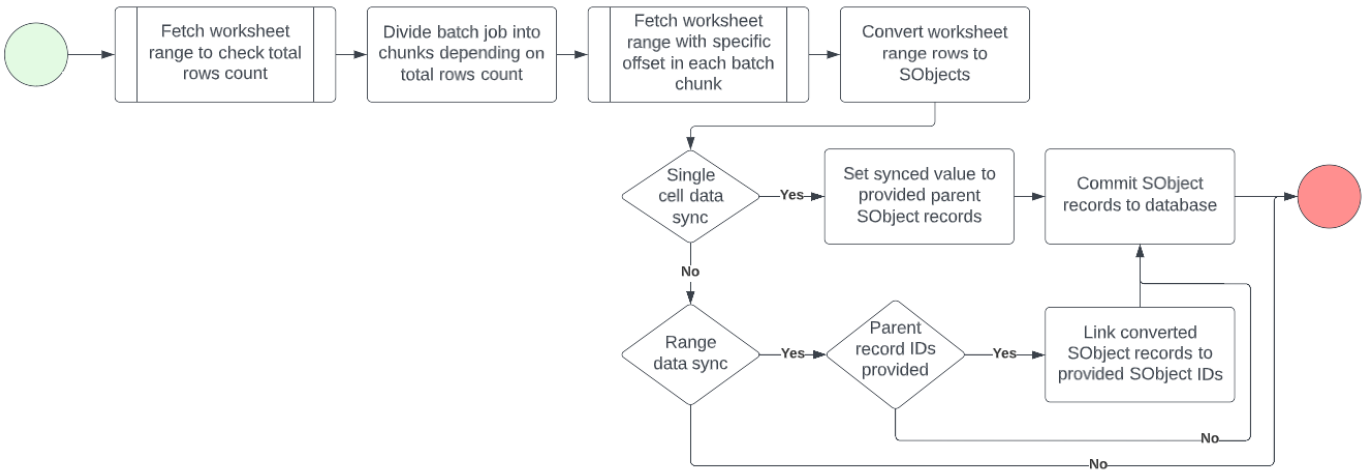
- **Fetch SharePoint excel worksheet range**



- **Fetch SharePoint excel worksheet range with offset**



- **Synchronize SharePoint excel data to Salesforce**



Solution Usage Instruction

The following setup steps are required after the installation of the "**SharePoint Data Sync Manager**" package:

1. Give application permission set
2. Configure named credentials to connect to SharePoint
3. Create corresponding SharePoint sync configurations
4. Configure application settings (optional)

Application Permissions Sets

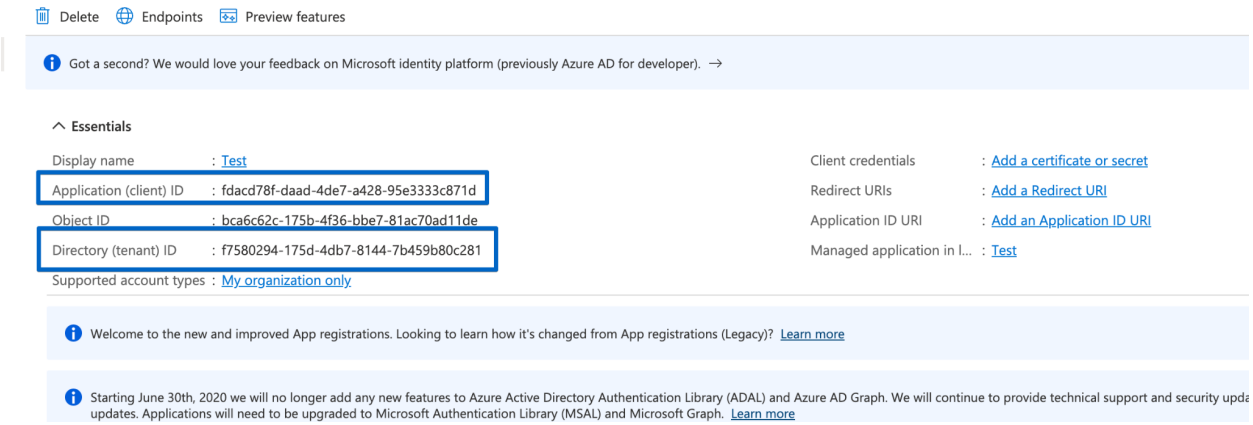
There are 2 permission sets in the package:

Permission Set Name	Description
SharePoint Data Sync Admin	This permission set is intended for application administrators. Users with such permission set: - Have access to "**SharePoint Data Sync Manager**" lightning application; - Can Read/Create/Edit/Delete all the SharePoint sync configurations; - Can View/Modify all the SharePoint sync logs; - Have access to application setup; - Can synchronize data from SharePoint to Salesforce.
SharePoint Data Sync User	This permission set is intended for application users that don't need administrator permissions. Users with such permission set: - Have access to "**SharePoint Data Sync Manager**" lightning application; - Can read all the SharePoint sync configurations; - Can synchronize data from SharePoint to Salesforce.

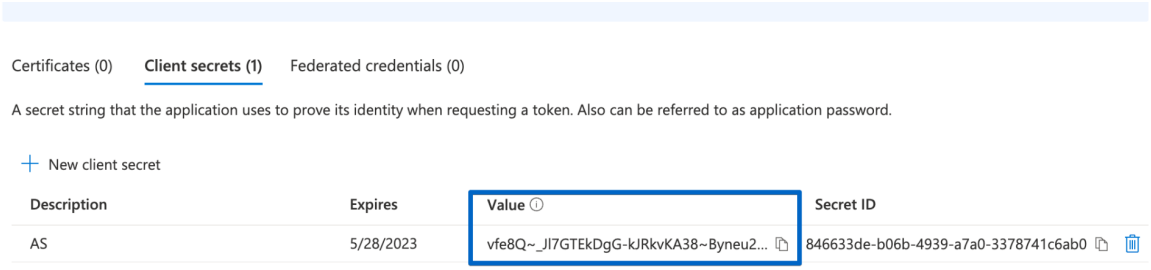
Named Credentials Configuration

The process of connection to SharePoint starts with configuration of Microsoft Azure:

1. Login to Microsoft Azure;
2. Open “**Azure Active Directory**”;
3. Select “**App registrations**”;
4. Click “**+ New registration**” and create an application;
5. Copy and save “**Application (client) ID**” and “**Directory (tenant) ID**”;



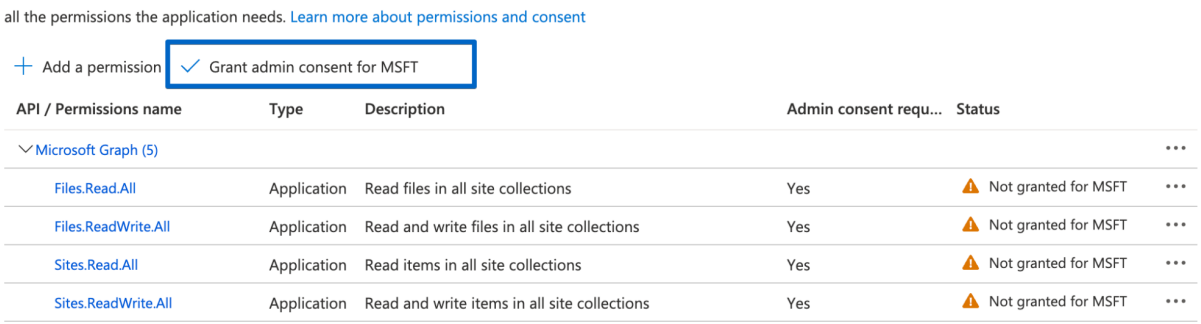
6. Open “**Certificates & secrets**” and create a client secret;
7. Copy and save “**Value**” column of the client secret;



8. Open “**API permissions**” and click “**+ Add a permission**”;
9. Select “**Microsoft Graph**” > “**Application permissions**”;
10. Add the following permissions:

    - Files.Read.All
    - Files.ReadWrite.All
    - Sites.Read.All
    - Sites.ReadWrite.All

11. Click the “**Grant admin consent**” button.



When the configuration of Microsoft Azure is done, the next step is to configure an authentication provider on the Salesforce side:

1. Open “**Setup**” > “**Auth. Providers**” and click “**New**” button;
2. Select “**AzureClientCredentialsAuthProvider**” option for “**Provider Type**”;
3. Enter the corresponding “**Name**” and “**URL Suffix**” values;
4. Enter the “**Callback URL**” value by the following pattern: “**/services/authcallback/YOUR_URL_SUFFIX**”;
5. Enter the “**Client ID**”, “**Client Secret**” and “**Tenant ID**” values you copied earlier;
6. Make sure the “**Scopes**” value is “**https://graph.microsoft.com/.default**”;
7. Enter the “**Custom Error URL**” value by the following pattern: “**YOUR_INSTANCE_URL/apex/sp_data_sync__ErrorAuth**”;

   Example: “**https://test-dev-ed.lightning.force.com/apex/sp_data_sync__ErrorAuth**”

8. Select the corresponding user for “**Execute As**”.

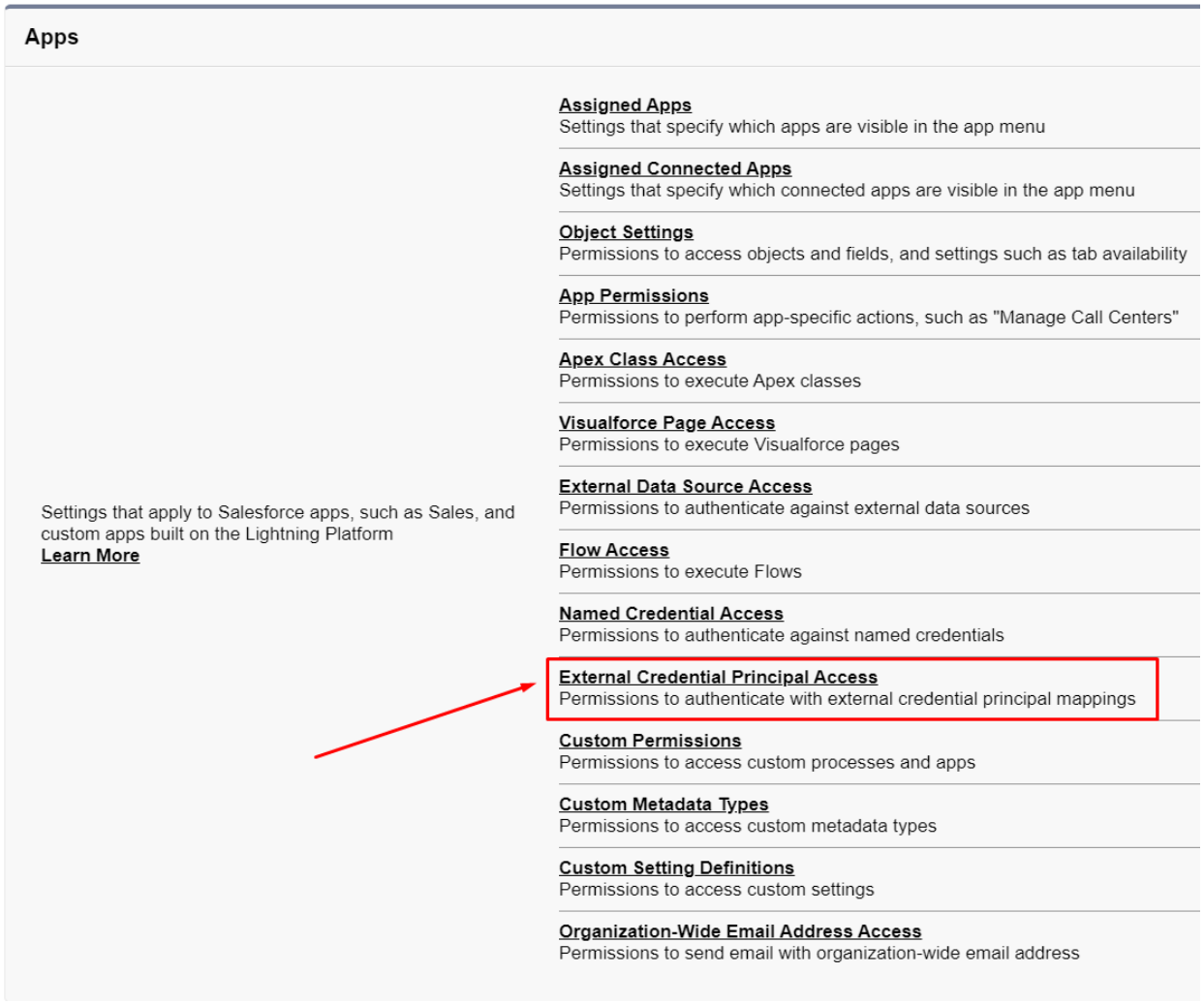
After the configuration of the authentication provider, the next step is to configure named credentials:

1. Open “**Setup**” > “**Named Credentials**”;
2. Select “**External Credentials**” tab and click “**New**” button;
3. Enter the corresponding “**Label**” and “**Name**” values;
4. Select “**OAuth 2.0**” option for “**Authentication Protocol**”;
5. Select “**Browser Flow**” option for “**Authentication Flow Type**”;
6. Select the authentication provider you created earlier for “**Authentication Provider**”;
7. Click “**Save**” button;
8. Scroll down to “**Principals**” section and click “**New**” button;
9. Enter the corresponding name for the principal (e.g. *Admin* or *Marketing Group*);
10. Select “**Named Principal**” option for “**Identity Type**”;
11. Click “**Save**” button;
12. Click “**Authenticate**” button for the created principal;

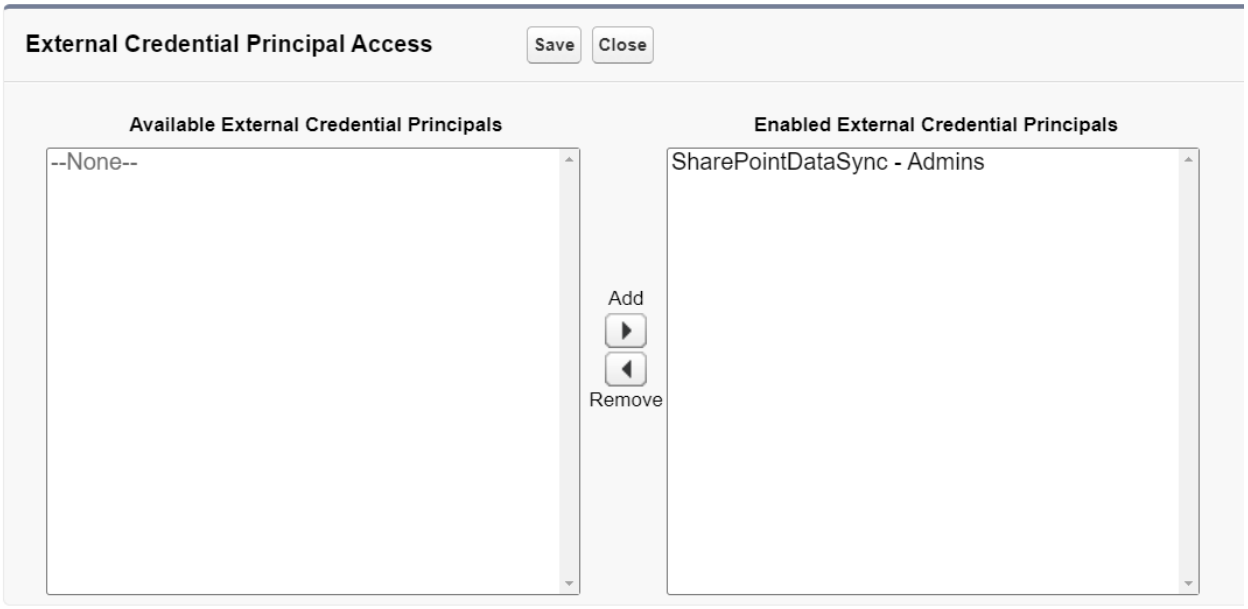
13. In case, the authentication is failed, check the authentication provider for correct setup and repeat step 12;
14. In case, the authentication is successful, the "**Authentication Status**" value should be changed to "**Configured**";
15. Open "**Setup**" > "**Named Credentials**";
16. Select "**Named Credentials**" tab and click "**New**" button;
17. Enter the corresponding "**Label**" and "**Name**" values;
18. Enter the "**https://graph.microsoft.com/v1.0**" value for "**URL**";
19. Select the external credential you created earlier for "**External Credential**";
20. Make sure the "**Generate Authorization Header**" checkbox is selected;
21. Enter the "**sp_data_sync**" namespace value for "**Allowed Namespaces**";
22. Click "**Save**" button.

The final step is to map the created named principal to a permission set/profile. It is necessary to authorize users to make callouts. For example, let's create a new permission set which gives access to the external credential principal:

1. Open "**Setup**" > "**Permission Sets**";
2. Click "**New**" button;
3. Enter the corresponding "**Label**" and "**API Name**" values and click "**Save**" button;
4. Click "**External Principal Access**";



5. Click “**Edit**” button;
6. Add the corresponding principal from “**Available External Credential Principal**” to “**Enabled External Credential Principal**” section;

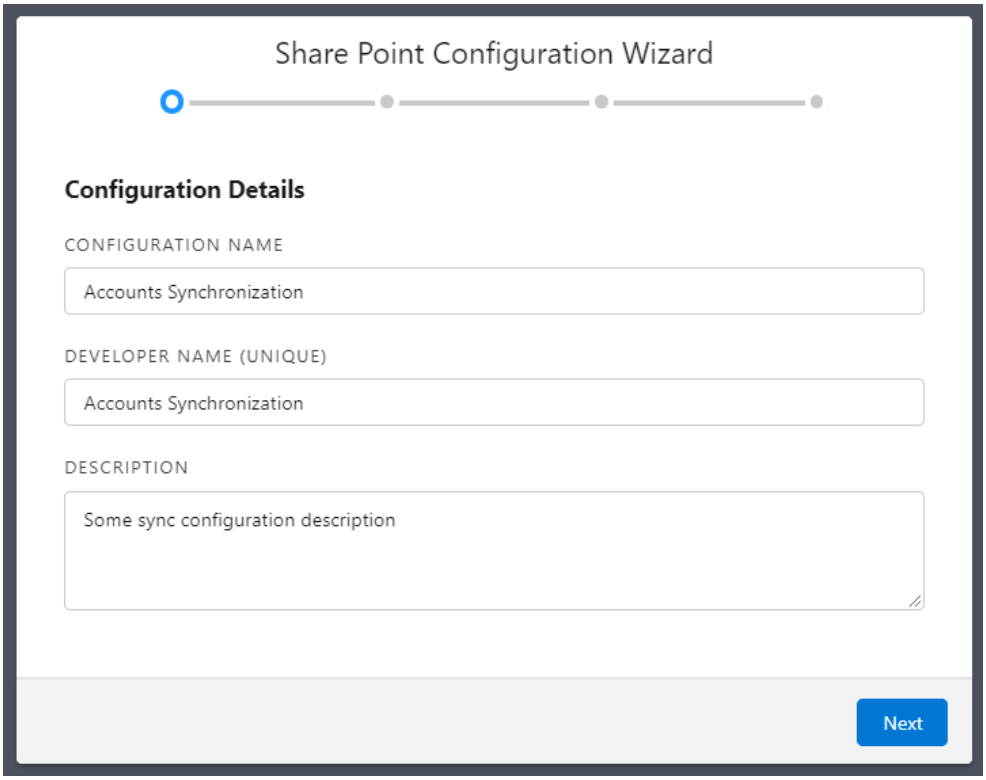


7. Click “**Save**” button;
8. Assign the permission set to the appropriate users.

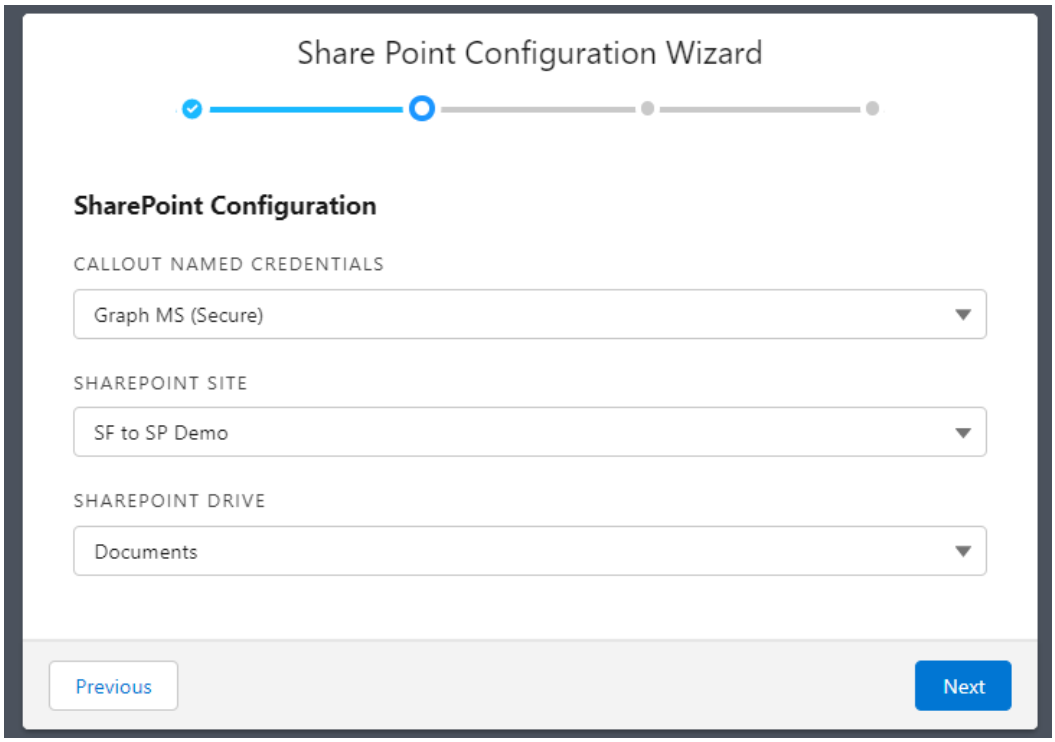
SharePoint Sync Configurations Setup

When the connection to SharePoint is properly configured and the necessary package permission set is granted to the user, the next step is to setup SharePoint sync configurations which will then be used for data synchronization.

To start creation of a SharePoint sync configuration, open a “**SharePoint Sync Configurations**” tab and click a “**New**” button. The first step is to enter the corresponding “**Name**”, “**Developer Name**” and “**Description**” values. Make sure that the “**Developer Name**” value is unique.



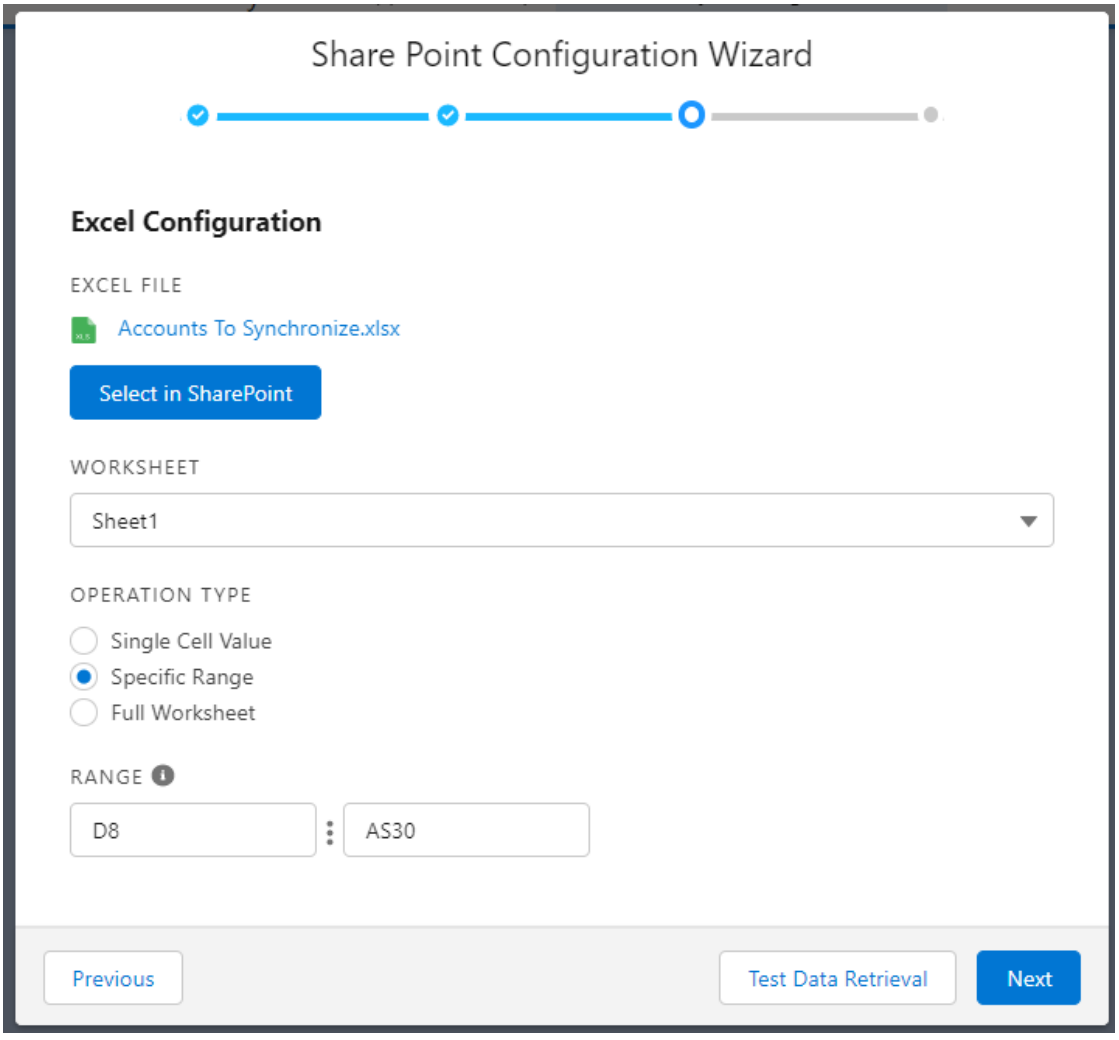
The next step is to specify the corresponding SharePoint configuration settings. Select the named credentials you created earlier for "**Callout Named Credentials**", choose the correct SharePoint site and its drive.



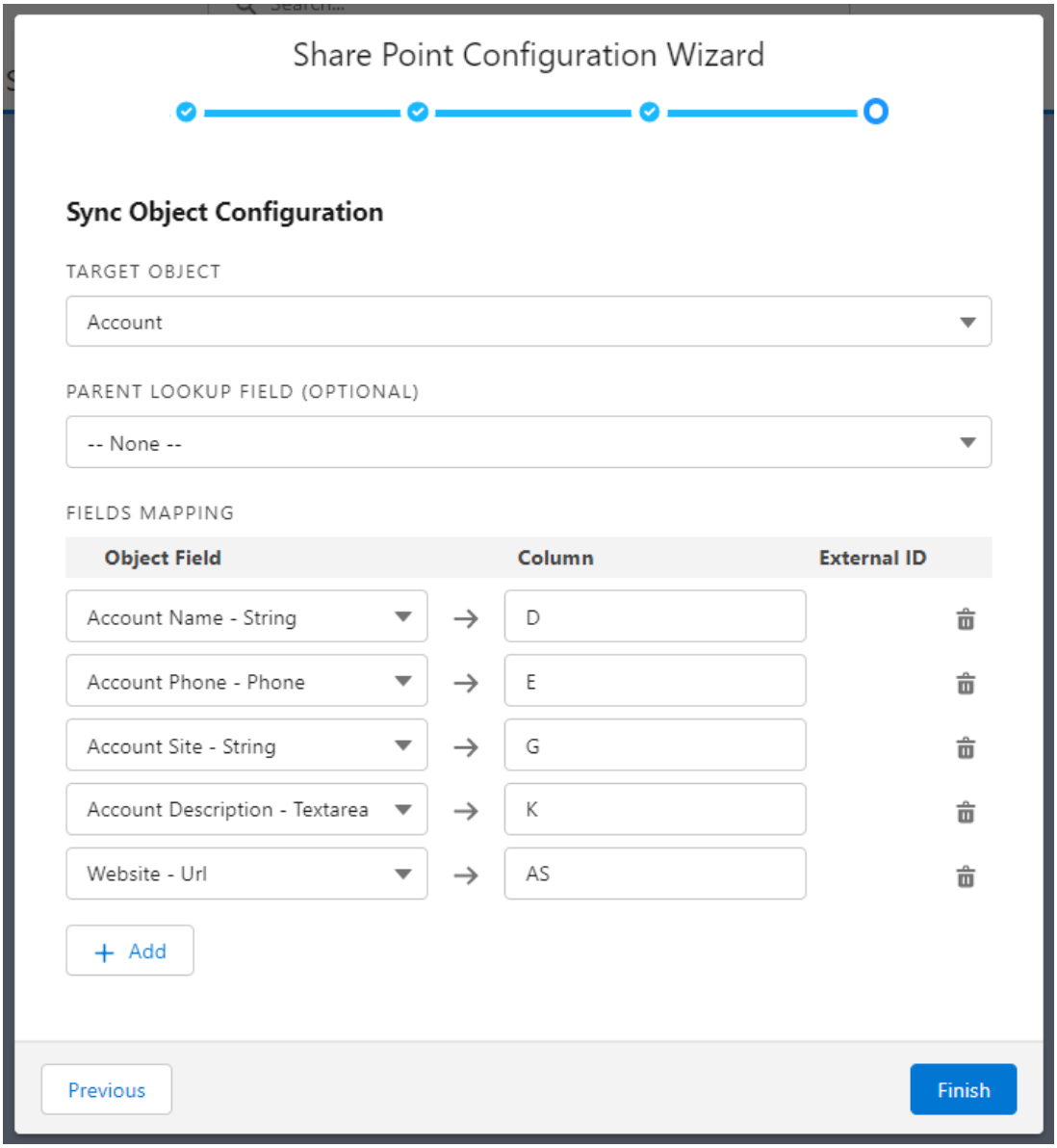
The next step is to select a SharePoint excel file which contains the data for synchronization, and specify its corresponding worksheet. Also, there must be specified the operation type for reading of data. The following operation types are supported:

- **Single Cell Value** - reading of data in a specific cell in the worksheet;
- **Specific Range** - reading of data in a specific range in the worksheet;
- **Full Worksheet** - reading of all the data in the worksheet;

To ensure the excel configuration is correct, click the "**Test Data Retrieval**" button. It will open a modal window with the read data from the worksheet.



The last step of the SharePoint sync configuration is to specify the synchronization object mapping. There must be selected a target Salesforce object and its fields mapping to worksheet columns.



When each step is properly configured, click the “**Finish**” button to create the SharePoint sync configuration.

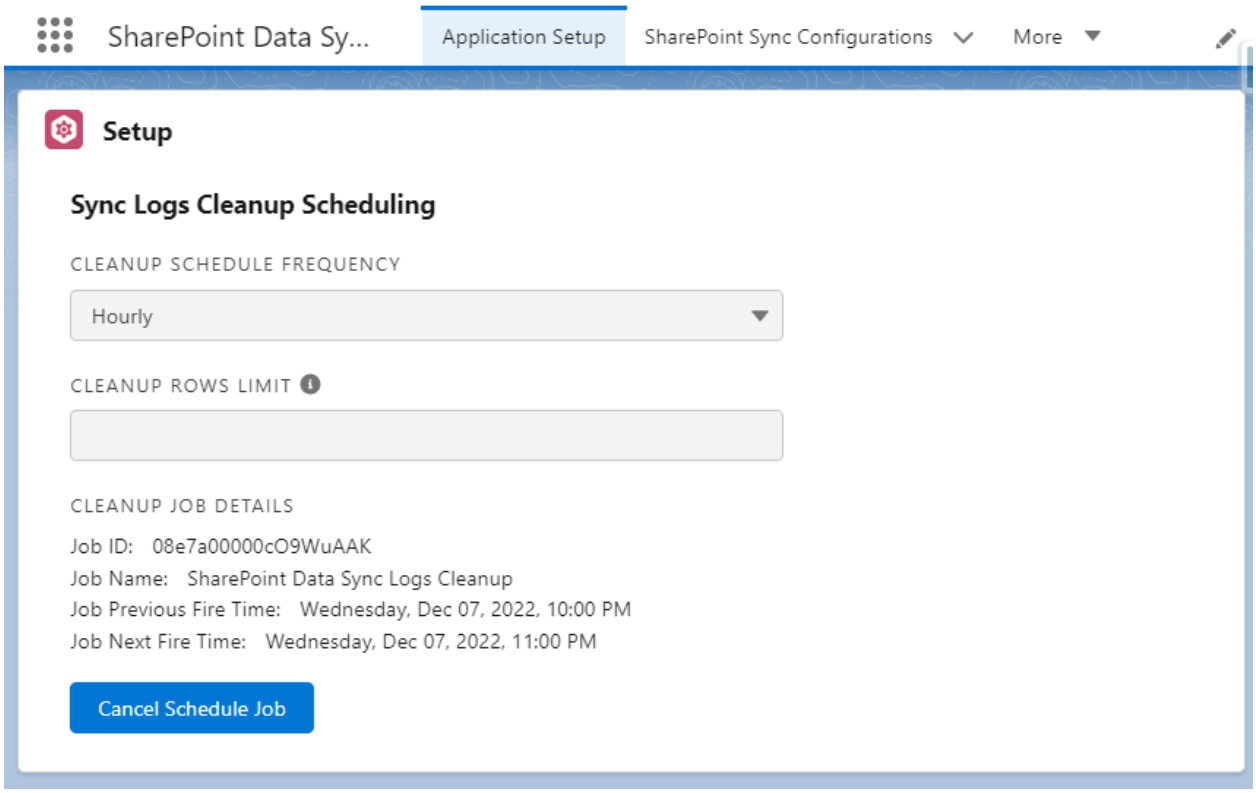
In order to be able to use the created sync configuration for data synchronization, it must be activated. This can be done with the “**Activate Configuration**” button. When the configuration is activated, it cannot be edited. In case the sync configuration must be adjusted, click the “**Deactivate Configuration**” button first and then “**Edit**” button.

Application Settings Configuration

There is an “**Application Setup**” tab which is visible for users with a granted “**SharePoint Data Sync Admin**” permission set. There the user can adjust various application settings.

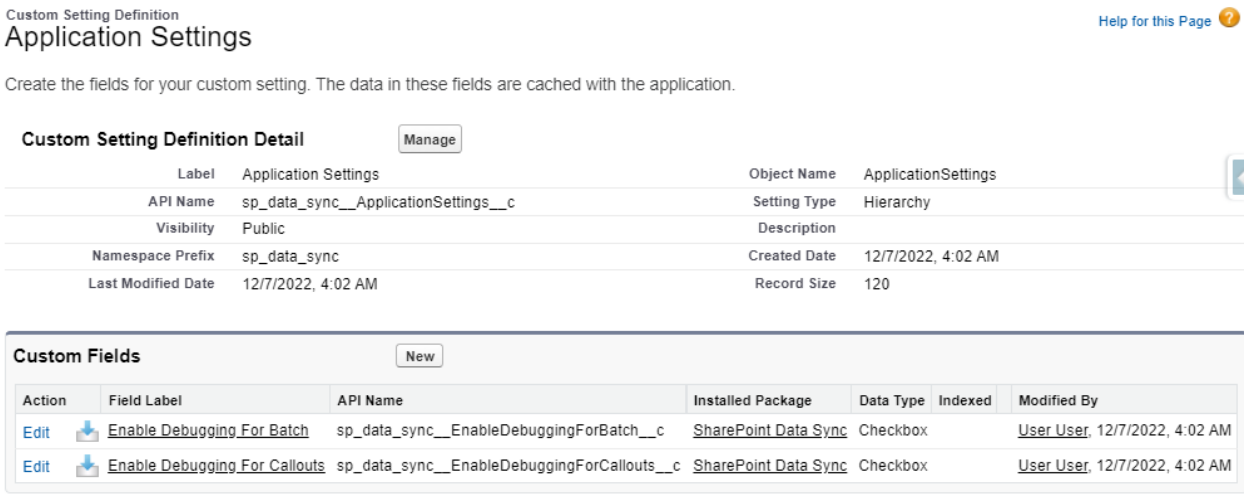
Currently, the following items can be configured in the “**Application Setup**” tab:

Setting Name	Description
Sync Logs Cleanup Scheduling	Setting which allows to schedule a job to regularly clean of application sync logs. The cleanup job can be scheduled to run hourly/daily/weekly/monthly.



Also, there is a public package “**Application Settings**” custom setting. Here the admin user can configure the debugging behavior of the package. The allowed settings are:

- **Enable Debugging for Batch** - if enabled, all the INFO debugs in the package batches are automatically logged;
- **Enabled Debugging for Callouts** - if enabled, all the http callouts to SharePoint system are automatically logged.



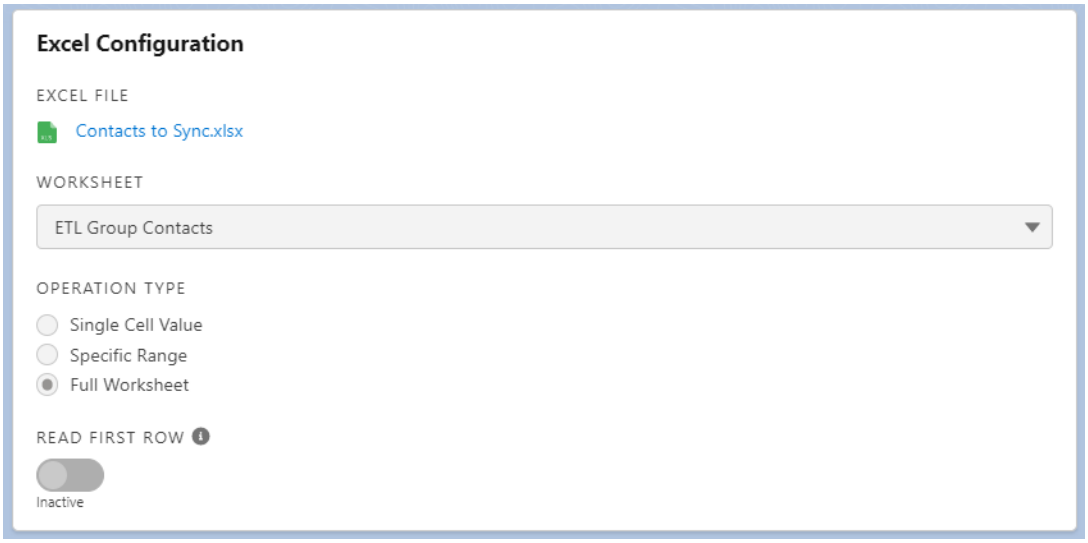
Congratulations! Each package configuration step is successfully completed. Now you are able to synchronize the data from SharePoint. There is a global interface in the package which can be used for data synchronization. Here is its description:

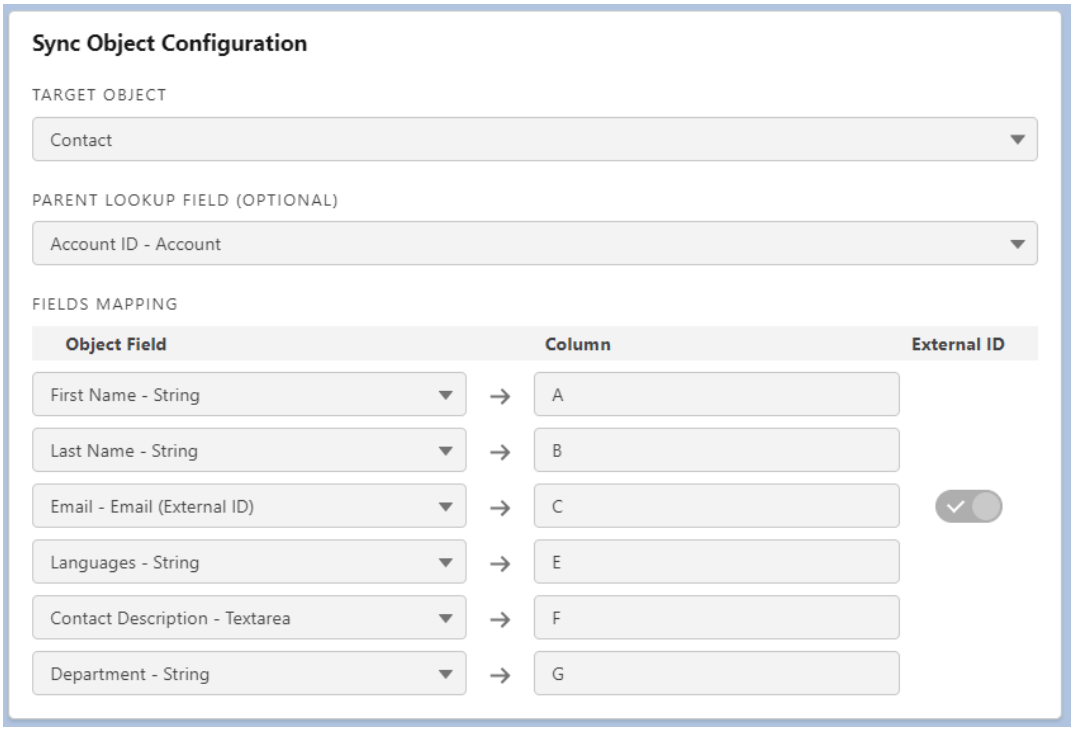
```
global with sharing class SharePointDataSynchronizator {

    /*
       This method is intended for synchronization of excel data to Salesforce.

       Method inputs are:
         - configurationDeveloperName - name of SharePoint sync configuration;
         - batchSize - records count to process in each batch chunk;
         - dmlOperationType - type of DML operation for synced records (INSERT or UPSERT).

       Method returns a batch ID.
    */
    global static Id syncExcelDataToSalesforce(
        String configurationDeveloperName,
        BatchSize batchSize,
        DmlOperationType dmlOperationType
    ) {
        ...
    }


    /*
       This method is intended for synchronization of excel data to Salesforce and linking
the synced records to specified SObject records.
```

```
       Method inputs are:
         - configurationDeveloperName - name of SharePoint sync configuration;
         - Set<Id> parentSObjectRecordIds  -  IDs of SObject records which must be linked to
synced excel records;
         - batchSize - records count to process in each batch chunk;
         - dmlOperationType - type of DML operation for synced records (INSERT or UPSERT).

       Method returns a batch ID.
    */
    global static Id syncExcelDataToSalesforce(
        String configurationDeveloperName,
        Set<Id> parentSObjectRecordIds,
        BatchSize batchSize,
        DmlOperationType dmlOperationType
    ) {
        ...
    }


    /*
       This method is intended for reading data from an excel worksheet.

       Method inputs are:
         - configurationDeveloperName - name of SharePoint sync configuration.

       Method returns read data in JSON.
    */
    global static String getExcelWorksheetData(String configurationDeveloperName) {
        ...
    }


    global enum BatchSize {
        SIZE_1,
        SIZE_10,
        SIZE_50,
        SIZE_100,
        SIZE_200
    }

    global enum DmlOperationType {
        INSERT_TYPE,
        UPSERT_TYPE
    }

}
```

The following are examples of various scenarios for data synchronization:

- **Synchronize and create all the contacts from an excel worksheet and link them to an account Salesforce record**



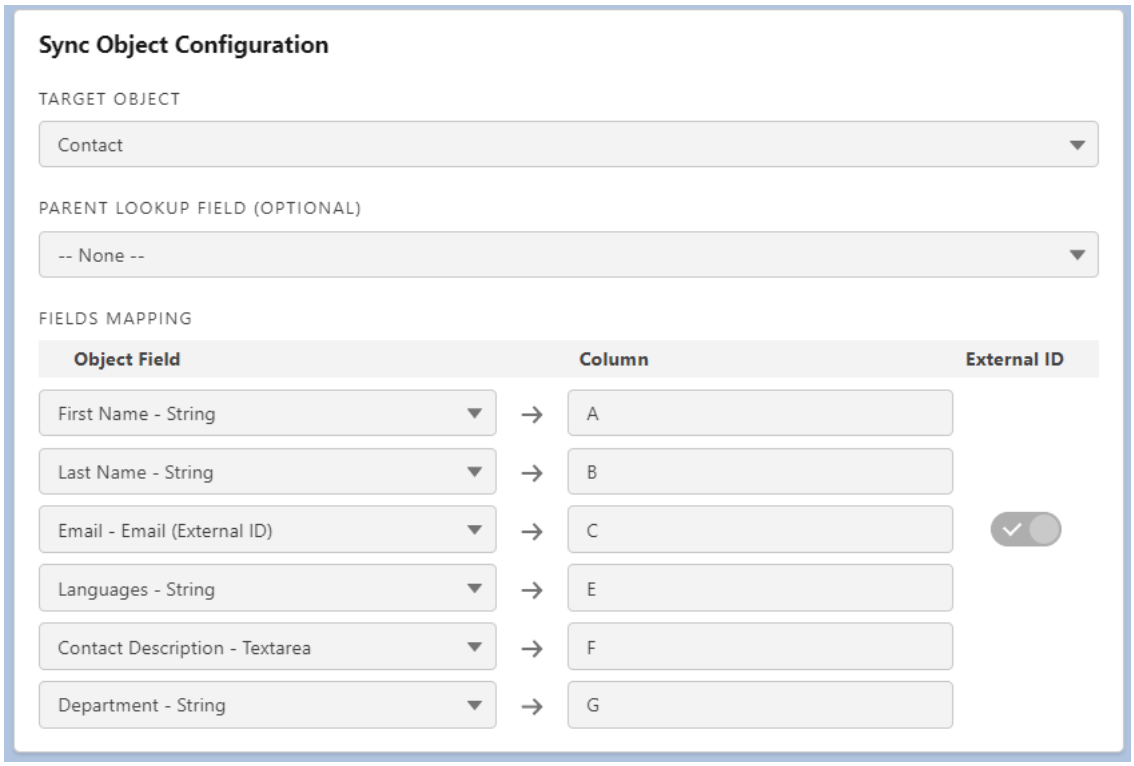


Code fragment:

```
String configurationDeveloperName = 'Contacts Synchronization';
Set<Id> accountIds = new Set<Id> { '0017a000027qDIRAA2' };
```

```
Id batchId = sp_data_sync.SharePointDataSynchronizator.syncExcelDataToSalesforce(
    configurationDeveloperName,
    accountIds,
    sp_data_sync.SharePointDataSynchronizator.BatchSize.SIZE_50,
    sp_data_sync.SharePointDataSynchronizator.DmlOperationType.INSERT_TYPE
);
```

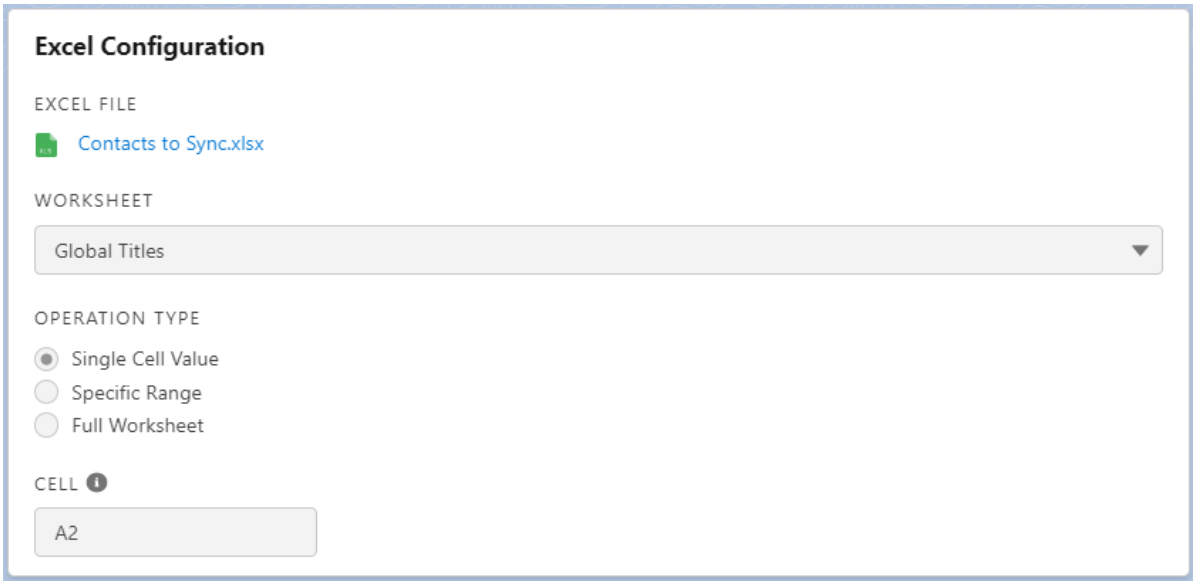
- **Synchronize and upsert specific contacts from an excel worksheet**

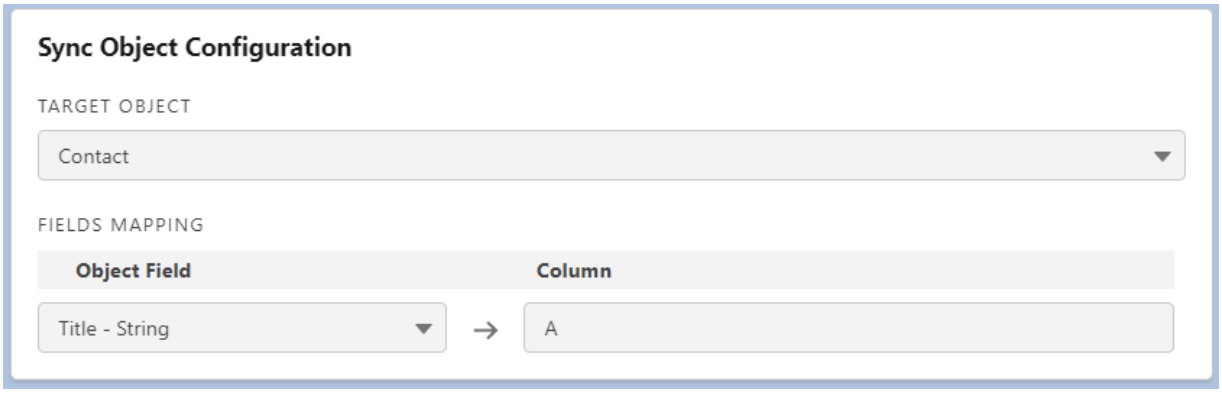


Code fragment:

```
String configurationDeveloperName = 'Contacts Synchronization';

Id batchId = sp_data_sync.SharePointDataSynchronizator.syncExcelDataToSalesforce(
    configurationDeveloperName,
    sp_data_sync.SharePointDataSynchronizator.BatchSize.SIZE_100,
    sp_data_sync.SharePointDataSynchronizator.DmlOperationType.UPSERT_TYPE
);
```

- **Synchronize and set specific contact title from an excel worksheet to specified contact Salesforce records**



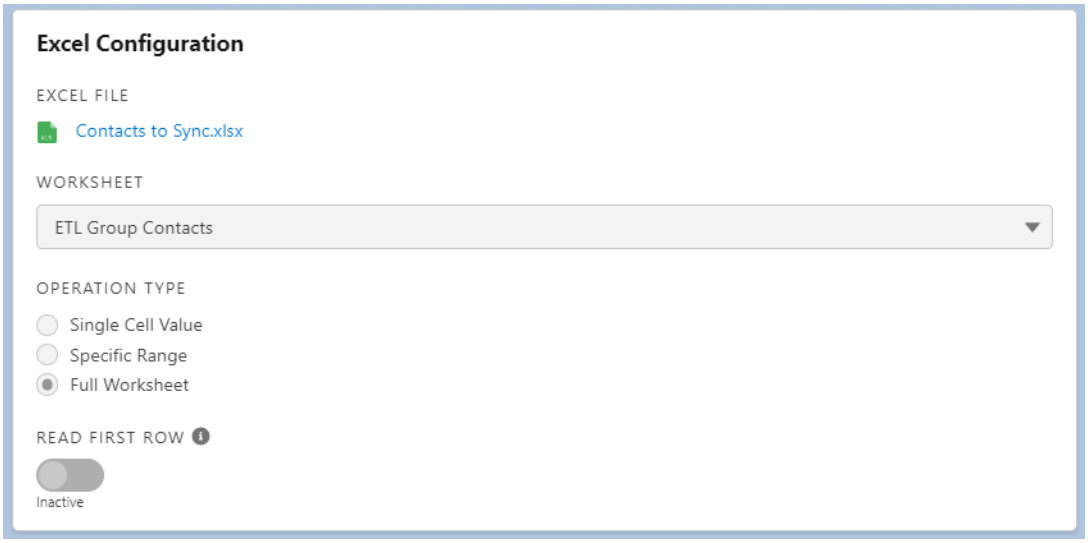


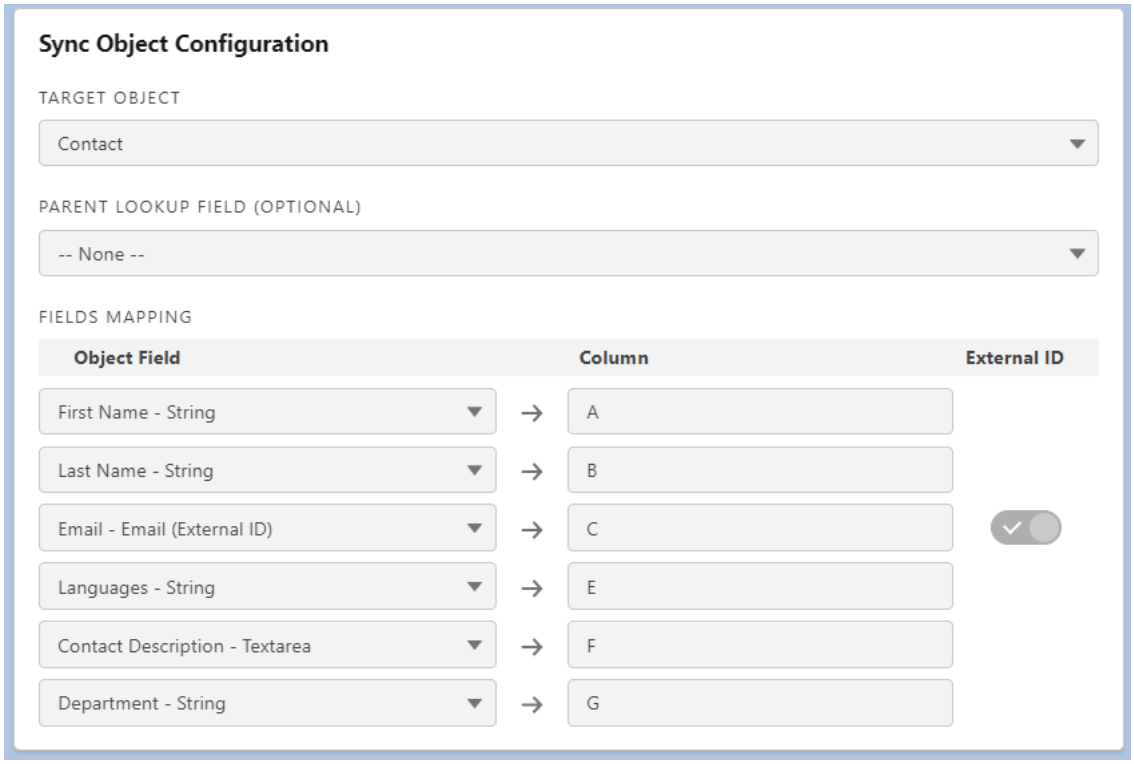
Code fragment:

```
String configurationDeveloperName = 'Contacts Title Synchronization';
Set<Id> contactIds = new Set<Id> {
```

```
    '0037a00001r0DZEAA2',
    '0037a00001r0DZDAA2',
    '0037a00001r0DZ9AAM',
    '0037a00001r0DZCAA2'
};

Id batchId = sp_data_sync.SharePointDataSynchronizator.syncExcelDataToSalesforce(
    configurationDeveloperName,
       contactIds,
       sp_data_sync.SharePointDataSynchronizator.BatchSize.SIZE_1,
       sp_data_sync.SharePointDataSynchronizator.DmlOperationType.UPSERT_TYPE
);
```

- **Read all the contacts from an excel worksheet**





Code fragment:

```
String configurationName = 'Contacts Synchronization';

String jsonResponse =
sp_data_sync.SharePointDataSynchronizator.getExcelWorksheetData(configurationName);

Map<String, Object> response = (Map<String, Object>) JSON.deserializeUntyped(jsonResponse);

System.debug('Address: ' + response.get('address'));
System.debug('Column Count: ' + response.get('columnCount'));
System.debug('Column Index: ' + response.get('columnIndex'));
System.debug('Row Count: ' + response.get('rowCount'));
System.debug('Row Index: ' + response.get('rowIndex'));
System.debug('Headers: ' + response.get('headers'));

for (Object row : (List<Object>) response.get('rows')) {
    System.debug('Row: ' + row);
}
```

JSON structures:

- **Worksheet Range:**

```
{
  "address": String,
  "columnCount": Integer,
  "columnIndex": Integer,
  "rowCount": Integer,
  "rowIndex": Integer,
  "headers": Object,
  "rows": array<Object>
}
```

- **Worksheet Range Row:**

```
{
  "values": array<Object>,
  "emptyRow": Boolean
}
```

- **Worksheet Range Row Value:**

```
{
  "valueType": Object,
  "value": Object
}


Supported value types:
    -   UNKNOWN_TYPE
    -   EMPTY_TYPE
    -   STRING_TYPE
    -   INTEGER_TYPE
    -   DOUBLE_TYPE
    -   BOOLEAN_TYPE
    -   ERROR_TYPE
```