SELISE Signature

Developer Manual & API-Reference

Version: 2.5

Date: 6th October 2025

Inhalt

Introduction

SELISE has a suite of microservices that enable client applications to work seamlessly. As a part of the Blocks ecosystem, the SELISE signature adheres to the guidelines and software specifications to enable their clients to consume the services through APIs (Application Programming Interface). Signature being a part and consumer of the blocks ecosystem it inherently uses some core microservices i.e., storage, identity, UAM (user-access-management), notification and many more.

In this documentation, we would provide the API steps to successfully initiate and sign documents using SELISE signature.

SELISE Signature	Swagger UI
Storage Service	Swagger UI
API Documentation	API Documentation (selisesignature.com)
Postman Collection	Visit this link

For Staging server, the base URLs and the versions are**:**

Storage Service (Status URL)	https://app.selisestage.com/api/**storageservice/v23**
Identity Service (Status URL)	https://app.selisestage.com/api/**identity/v25**
Signature Service (Status URL)	https://app.selisestage.com/api/**selisign**/v**65**

For Production the base URLs and the versions are**:**

Storage Service (Status URL)	https://selise.app/api/**storageservice/v100**
Identity Service (Status URL)	https://selise.app/api/**identity/v100**
Signature Service (Status URL)	https://selise.app/api/**selisign/s1**

Please note the above URLs depict the base address for the SELISE microservices. **You need to use the complete URL formatted correctly** to reach the endpoints. The complete URL pattern is provided below along with an example.

Replace

https://app.selisestage.com/api/{**serviceName**}/{**version**} -> https://selise.app/api/{**serviceName**}/{**version**}

i.e.:

https://selise.app/api/storageservice/v100/StorageService/Management/Ping

https://selise.app/api/identity/v100/identity/token

Versions

	Staging	Production
Signature	v65	s1
Identity	v25	v100
Storage	v23	v100

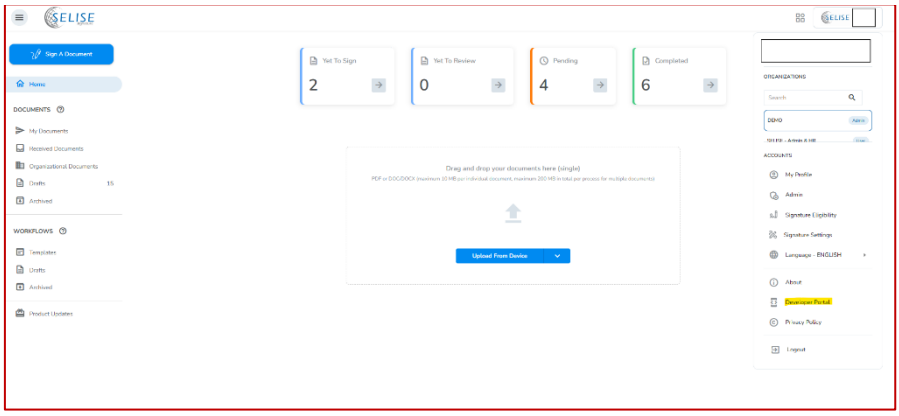
API has the following URL format

Host	Path	Version	Resource Path
https://selise.app	api/selisign	s1	SeliSign/ExternalApp/PrepareContract
https://selise.app	api/storageservice	v100	StorageService/StorageQuery/GetPreSignedUrlForUpload

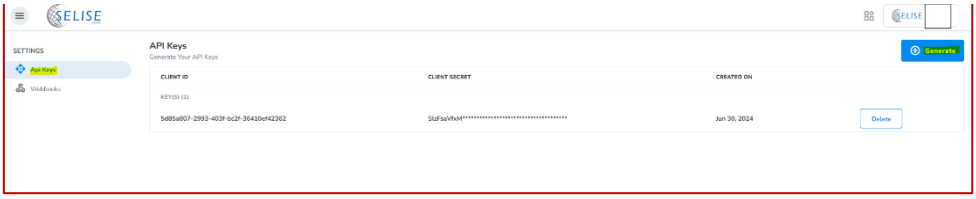
Generate Credentials

You can get your own client credentials from SELISE Signature portal. Go to the Developer Portal and generate your own credentials. Note that the secret is visible for the first 60 minutes only, so we advise you to store the secret somewhere safe.

1. Login to Signature **2.** Got to Developer Portal **3.** Choose API Keys **4. Generate**



Developer portal in SELISE Signature



Generate Client credentials

Token

Before using this API, one must sign up for the SELISE app portal. Navigate into the **Developer Portal** and create a Client Credential. Use the created credential to generate a token from the SELISE identity service. Use this token to make HTTP calls.

Javascript

```
fetch('https://app.selisestage.com/api/identity/v25/identity/token', {
  method: 'POST',
  headers: {
   'Origin': 'https://app.selisestage.com'
  },
  body: new URLSearchParams({
   'grant_type': 'client_credentials',
   'client_id': '***',
   'client_secret': 'Slz***'
  })
});
```

C#

```
var client = new HttpClient();
var request = new HttpRequestMessage(HttpMethod.Post,
"https://app.selisestage.com/api/identity/v25/identity/token");
request.Headers.Add("Origin", "https://app.selisestage.com");
var collection = new List<KeyValuePair<string, string>>();
collection.Add(new("grant_type", "client_credentials"));
collection.Add(new("client_id", "***"));
collection.Add(new("client_secret", "Slz***"));
var content = new FormUrlEncodedContent(collection);
request.Content = content;
var response = await client.SendAsync(request);
response.EnsureSuccessStatusCode();
Console.WriteLine(await response.Content.ReadAsStringAsync());
```

Java

```
OkHttpClient client = new OkHttpClient().newBuilder()
  .build();
MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
RequestBody body = RequestBody.create(mediaType, "grant_type=client_credentials&client_id= *** &client_secret=Slz***");
Request request = new Request.Builder()
  .url("https://app.selisestage.com/api/identity/v25/identity/token")
  .method("POST", body)
  .addHeader("Origin", "https://app.selisestage.com")
  .addHeader("Content-Type", "application/x-www-form-urlencoded")
  .build();
Response response = client.newCall(request).execute();
```

Sample Response:


```
{
  "scope": "offline_access",
  "token_type": "Bearer",
  "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZW5hbnRfaWQiOiI4NzRBNkEyQi1GMEY1LTQzRUItQTQ1OC00N0MyQzFBMjEyMzEiLCJzdWIiOiIyNGRlNTM5NS02ZWRhLTQ1OTUtODg4Mi04OTkyNmZkN2QwNTEiLCJzaXRlX2lkIjoiMzMyRjcxNjEtNjc0QS00MDk2LUE3OUItM0E2Rjg0Q0U5NTdGIiwib3JpZ2luIjoiYXBwLnNlbGlzZXN0YWdlLmNvbSIsInNlc3Npb25faWQiOiJzZWxpc2VibG9ja3MtMDMwY2M3ZWEtNzhmMi00NjI3LTk5ZTYtZGE0ZDZlOTkiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjI0ZGU1Mzk1LTZlZGEtNDU5NS04ODgyLTg5OTI2ZmQ3ZDA1MSIsImRpc3BsYXlfbmFtZSI6IkNsaWVudCBDcmVkZW50YWlscyIsInNpdGVfbmFtZSI6IkVjYXAgVGVhbSIsInVzZXJfbmFtZSI6IjI0ZGU1Mzk1LTZlZGEtNDU5NS04ODgyLTg5OTI2ZmQ3ZDA1MSIsImVtYWlsIjoiaW5mb0BzZWxpc2UuY2giLCJwaG9uZV9udW1iZXIiOiJuby1waG9uZSIsImxhbmd1YWdlIjoiZW4tVVMiLCJ1c2VyX2xvZ2dlZGluIjoiVHJ1ZSIsIm5hbWUiOiIyNGRlNTM5NS02ZWRhLTQ1OTUtODg4Mi04OTkyNmZkN2QwNTEiLCJ1c2VyX2F1dG9fZXhwaXJlIjoiRmFsc2UiLCJ1c2VyX2V4cGlyZV9vbiI6IjAxLzAxLzAwMDEgMDA6MDA6MDAiLCJvcmdpZCI6IjM0NzQ1NGIyLTFkNWItNGQwNC04ZmQ2LTYwMjdiODcyYmE5NCIsInJvbGUiOlsiYXBwdXNlciIsImNvbXBhbnlhZG1pbiIsImFub255bW91cyIsImNvbXBhbnl1c2VyIiwiZXNpZ2V4dGVybmFsIiwidXNlcl8zNDc0NTRiMi0xZDViLTRkMDQtOGZkNi02MDI3Yjg3MmJhOTQiXSwiaGRyIjoiVHJ1ZSIsIm5iZiI6MTcyNjU4NzMyMSwiZXhwIjoxNzI2NTg3NzQxLCJpc3MiOiJDTj1FbnRlcnByaXNlIENsb3VkIEFwcGxpY2F0aW9uIFBsYXRmb3JtIiwiYXVkIjoiKiJ9.ac_FFfSgTdytOiMnLPmMuEF-6lLhkc9tB5i9urgwWTXhGo-CZy6jHKtfQqE4KMuX9UGZfpvIxG2R677B-1Ft-Iqg782FlgLHbRYqUUglpcMxivt-hjwZf-5zACS1pX3ZL3P8s1In3zD2nnTq8TCQeucsHWJGQEBRXImUutz_tjA3-rkzEl4e-BUgLOPuLozRfQM8MfZSu4hUwALKLyLl7z0xudm0dR2DCxkPor625SQQ1eVOwoaXuyf247xR2YnDLKdAAJkQfKBf53_FepOv6isG0BLs1P0y8jCMxgLNBzSQpwmkZHseB4Ju_oXObzk3rzzDb2saZqPNAjVcJZ878w",
  "expires_in": 420,
  "refresh_token": "370***",
  "ip_address": "59.153.102.108",
  "may_access": "selisesignature.com,free.selisestage.com,"
}
```


Notice: After configuring client credentials go to the Developer Portal and configure Webhooks. SELISE signature forwards all the document preparation/processing steps to the configured webhooks. You can configure your own webhooks into our system and trigger/customise actions on your side.

Files

Upload File

This API is a part of SELISE storage service and is used to upload and download files. To continue using the SELISE signature, you must have PDFs and signature images uploaded into the system. You can use this API to upload PDFs and Signature images in the SELISE cloud storage. The signature application can access these files from the signature application process.

Uploading is a two-step process.

Step 1: HTTP – Post request to the GetPreSignedUrlForUpload API

Javascript

```
fetch('https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetPreSignedUrlForUpload', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'bearer {{TOKEN}}'
  },
  body: JSON.stringify({
    'ItemId': 'Use a guid',
    'MetaData': '{}',
    'Name': 'file_name.format',
    'ParentDirectoryId': '',
    'Tags': '["File"]',
    'AccessModifier': 'Private'
  })
});
```

C#

```
var client = new HttpClient();
var request = new HttpRequestMessage(HttpMethod.Post,
"https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetPreSignedUrlForUpload");
request.Headers.Add("Authorization", "Bearer {{TOKEN}}");
var content = new StringContent("{ \n    \"ItemId\": \"Use a guid\", \n    \"MetaData\": \"{}\", \n    \"Name\": \"file_name.format\", \n
\"ParentDirectoryId\": \"\", \n    \"Tags\": \"['File']\", \n    \"AccessModifier\": \"Private\" \n  }", null, "application/json");
request.Content = content;
var response = await client.SendAsync(request);
response.EnsureSuccessStatusCode();
Console.WriteLine(await response.Content.ReadAsStringAsync());
```

Java

```
OkHttpClient client = new OkHttpClient().newBuilder()
  .build();
MediaType mediaType = MediaType.parse("application/json");
RequestBody body = RequestBody.create(mediaType, "{ \n    \"ItemId\": \"Use a guid\", \n    \"MetaData\": \"{}\", \n    \"Name\":
\"file_name.format\", \n    \"ParentDirectoryId\": \"\", \n    \"Tags\": \"['File']\", \n    \"AccessModifier\": \"Private\" \n  }");
Request request = new
Request.Builder() .url("https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetPreSignedUrlForUpload")
  .method("POST", body)
  .addHeader("Content-Type", "application/json")
  .addHeader("Authorization", "Bearer {{TOKEN}}")
  .build();
Response response = client.newCall(request).execute();
```

Sample Response


```
{
    "UploadUrl": "https://arcfalcon.blob.core.windows.net/stage/874A6A2B-F0F5-43EB-A458-47C2C1A21231/7ebeb40a-4eb0-45c8-b213-85c80ce9b0b4/5956b4d3-7964-453c-b2f9-d1ffefd8f560/file.pdf?sv=2018-03-28&sr=b&sig=gGJI5KXnPYY0M%2B3R%2BN8MXBT0UA%2FHOrQEQpSorfsxah0%3D&se=2024-05-01T10%3A14%3A59Z&sp=w",
    "FileId": "7ebeb40a-4eb0-45c8-b213-85c80ce9b0b4",
    "StatusCode": 0,
    "RequestUri": null,
    "ExternalError": null,
    "HttpStatusCode": 0
}
```


Step 2: To finish uploading, add the file contents and make HTTP PUT.

Javascript

```
const file = "<file contents here>";
const requestOptions = {
  method: "PUT",
  body: file,
  redirect: "follow"
};
fetch("https://arcfalcon.blob.core.windows.net/stage/874A6A2B-F0F5-43EB-A458-47C2C1A21231/7ebeb40a-4eb0-45c8-b213-85c80ce9b0b4/5956b4d3-7964-453c-b2f9-d1ffefd8f560/file.pdf?sv=2018-03-28&sr=b&sig=gGJI5KXnPYY0M%2B3R%2BN8MXBT0UA%2FHOrQEQpSorfsxah0%3D&se=2024-05-01T10%3A14%3A59Z&sp=w",
requestOptions)
  .then((response) => response.text())
  .then((result) => console.log(result))
```

```
  .catch((error) => console.error(error));
```

C#

```
var client = new HttpClient();
var request = new HttpRequestMessage(HttpMethod.Put, "https://arcfalcon.blob.core.windows.net/stage/874A6A2B-F0F5-43EB-A458-47C2C1A21231/7ebeb40a-4eb0-45c8-b213-85c80ce9b0b4/5956b4d3-7964-453c-b2f9-d1ffefd8f560/file.pdf?sv=2018-03-28&sr=b&sig=gGJI5KXnPYY0M%2B3R%2BN8MXBT0UA%2FHOrQEQpSorfsxah0%3D&se=2024-05-01T10%3A14%3A59Z&sp=w");
request.Headers.Add("x-ms-blob-type", "BlockBlob");
request.Content = new StreamContent(File.OpenRead("/location..."));
var response = await client.SendAsync(request);
response.EnsureSuccessStatusCode();
Console.WriteLine(await response.Content.ReadAsStringAsync());
```

Java

```
OkHttpClient client = new OkHttpClient().newBuilder()
  .build();
MediaType mediaType = MediaType.parse("application/pdf");
RequestBody body = RequestBody.create(mediaType, "<file contents here>");
Request request = new Request.Builder()
  .url("https://arcfalcon.blob.core.windows.net/stage/874A6A2B-F0F5-43EB-A458-47C2C1A21231/7ebeb40a-4eb0-45c8-b213-85c80ce9b0b4/5956b4d3-7964-453c-b2f9-d1ffefd8f560/file.pdf?sv=2018-03-28&sr=b&sig=gGJI5KXnPYY0M%2B3R%2BN8MXBT0UA%2FHOrQEQpSorfsxah0%3D&se=2024-05-01T10%3A14%3A59Z&sp=w")
  .method("PUT", body)
  .addHeader("x-ms-blob-type", "BlockBlob")
  .addHeader("Content-Type", "application/pdf")
  .build();
Response response = client.newCall(request).execute();
```

Get File / Get Files

This API lets you get files from SELISE cloud storage.

```
fetch('https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetFile?FileId=esignature-coordinate-file', {
  method: 'POST',
  headers: {
    'Authorization': 'bearer {{TOKEN}}',
    'Content-Type': 'application/json'
  },
  body: ''
});
```

```
fetch('https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetFiles', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'bearer {{TOKEN}}'
  },
  body: JSON.stringify({
    'FileIds': [
      '3807bc8d-0500-84a9-cecb-2607f969c102',
      'ef6ca88a-4ea0-1f98-37c1-be88c33915d4',
      '23e0de16-4f11-c25a-6323-95501039b421',
      '185c5bc3-2c0e-656a-c7e7-0dfef12b56cf',
      '8074c608-b0f1-cc5a-e71a-3fec5b101ee8'
    ]
  })
});
```

Contract Processing

Prepare Contract

Prepare Contract API takes care of the contract creation process for external consumers. Before using this API, one must sign up for the SELISE app portal. Navigate into the Developer Portal and create a Client Credential. Use the **client_credential** generated token to make a request to *Prepare Contract* API.

Scope: The scope of this API is limited to preparing the contract. Namely, a contract is created, legal weight is defined, signatories are set up, and finally directed towards the preparation page.

You can either continue preparing the signature placeholders from this drafted document and send the document to the signatories/other roles manually using the UI. Or you can follow the next step for continuing with an API.

```
const myHeaders = new Headers();
myHeaders.append("Authorization", "bearer {{TOKEN}}");
myHeaders.append("Content-Type", "application/json");

const raw = JSON.stringify({
  "TrackingId": "713499c8-6b83-4146-8de4-754ef687ff64",
  "Title": "Test Contract 01",
  "ContractType": 0,    // 0 for individual contract, 1 for organizational contract
  "ReturnDocument": true,
  "ReceiveRolloutEmail": true,
  "SignatureClass": 0,
  "Language": "en-US",
  "LandingPageType": 0,
  "ReminderPulse": 168,
  "OwnerEmail": "demo_user_1@selisegroup.com", // OwnerEmail when not defined, takes the first
                          //email
                          // from AddSignatoryCommands list
  "FileIds": [
            "7449e414-a2de-46c7-b7c4-7d09e8acd256"
  ],
  "AddSignatoryCommands": [
            {
            "Email": "demo_user_1@selisegroup.com",
            "ContractRole": 0,
            "FirstName": "Demo",
            "LastName": "User 1",
     "Phone": "+41448058044", // Phone number is optional
            },
            {
            "Email": "demo_user_2@selisegroup.com",
            "ContractRole": 0,
            "FirstName": "Demo",
            "LastName": "User 2"
            }
```

```
  ],
  "SigningOrders": [ // SigningOrders property is nullable
                {
                "Email": "demo_user_1@selisegroup.com",
                "Order": 1
                },
                {
                "Email": "demo_user_2@selisegroup.com",
                "Order": 2
                }
  ],
  "RedirectUrl": "http://app.selisestage.com/e-signature/contracts/"
});

const requestOptions = {
  method: "POST",
  headers: myHeaders,
  body: raw,
  redirect: "follow"
};

fetch("https://app.selisestage.com/api/selisign/v65/SeliSign/ExternalApp/PrepareContract", requestOptions)
  .then((response) => response.text())
  .then((result) => console.log(result))
  .catch((error) => console.error(error));
```

Note: The SigningOrders property is nullable. When this property is set to null, the contract will have no defined signing order, meaning all participants will receive the contract simultaneously. However, if a signing order is defined, the contract will be forwarded to the participants in the specified order, ensuring that each signer or reviewer receives the contract only after the preceding participant has completed their action.

OwnerEmail property can be defined to explicitly set the owner/initiator of the contract. By default, the first signatory from the list is chosen as the initiator. **OwnerEmail** property `value` should be strictly present in the list of signatories, otherwise API will return validation errors.

SignatureClass property determines the type of signature to be applied. Use the corresponding value based on the signature class you want to use from the list below.

0: Simple
2: Advanced for Switzerland (ZertES Swiss Law)
3: Qualified with Swisscom (ZertES - Swiss Law)
7: Qualified with Swisscom (eIDAS - EU Law)

9: Advanced for EU (eIDAS)

Language property can be defined to set the preferred language for a new user (a user who does not yet belong to our system). The new user will receive the contract email in the specified language, and their profile's default language will also be set according to this property. Please use the corresponding language code based on the desired language as follows:

For English: "en-US"

For German: "de-DE"

LandingPageType property (nullable) determines where the user will be redirected after creating the contract. Based on its value, the user will land on either the Prepare Page or the Add Signatory Page. Use the appropriate value from the list below:

0: Prepare Page

1: Add Signatory Page

ReminderPulse property (nullable) is used to configure automatic reminder emails for contract participants. Depending on its value, participants will receive reminders at regular intervals. **Supported values (in hours): 24, 48, 72, 168** — which correspond to daily, every 2 days, every 3 days, and weekly reminders respectively.

Rules for Creating Contracts with Signing Order

1. The SigningOrder property can be set to null, which will disable Sequential Signing for the contract, allowing all participants to receive the contract simultaneously.
2. When Sequential Signing is enabled, each signing order value must be greater than 0.
3. All participants in the signatory list must have a defined signing order.

Rollout

Rollout API takes the signature stamp coordinates, text fields, initials, images, date placeholders as input and finalises the process of sending the document to the desired signatories and viewers / reviewers.

Scope: This API finalises the preparation of the document. You can configure where the signature stamp will be placed and what components will be drawn on top of the pdf etc. When this API request is made the signatories will get an email invitation for signing or reviewing the document. Along with this, the document state would be shifted to the signing flow. You can use the email one-click login link to login directly into SELISE app and sign the document.

```
fetch('https://app.selisestage.com/api/selisign/v65/SeliSign/ExternalApp/RolloutContract', {
  method: 'POST',
  headers: {
   'Authorization': 'bearer {{TOKEN}}',
   'Content-Type': 'application/json'
  },
  body: JSON.stringify({
   'DocumentId': '0988f5d1-7f45-4946-8248-bf7ebaf5e7e8',
   'StampCoordinates': [
    {
     'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
     'Width': 150,
     'Height': 100,
     'PageNumber': 0,
     'X': 210.44444900390624,
     'Y': 254.22220474121093,
     'SignatoryEmail': 'demo_user@selisegroup.com',
     'SignatureImageFileId': 'b9c582ec-44f1-41ec-ab21-0cb309ec8d45'
    }
   ],
   'TextFieldCoordinates': [
    {
     'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
     'Width': 127.7043269230769,
     'Height': 27.043269230769226,
     'PageNumber': 0,
     'X': 275.2737747925978,
     'Y': 202.24023793660675,
     'SignatoryEmail': 'demo_user@selisegroup.com',
     'Value': 'Demo User
    }
   ],

  //more on next page
```

```
'StampPostInfoCoordinates': [
    {
     'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
     'Width': 127.7043269230769,
     'Height': 27.043269230769226,
     'PageNumber': 0,
     'X': 288.0442074849055,
     'Y': 247.59405404237597,
     'EntityName': 'AuditLog',
     'PropertyName': '{StampTime}',
     'SignatoryEmail': 'demo_user@selisegroup.com'
    }
   ]
  })
})
```

Aggregated Rollout

Aggregated Rollout API combines the **Prepare Contract API** and **Rollout API.** Instead of making two separate HTTP requests you can combine the two request payloads within the API.

Scope: All the stages of Prepare and Rollout will be done cumulatively when this API is called. Document details, Signature types, signatories will be selected at first. Secondly, the signature coordinates, text fields, images, and other coordinates will be applied to the document in the later stage. Finally, the document will be rolled out and invitation emails will be sent to the participants.

```
fetch('https://app.selisestage.com/api/selisign/v65/SeliSign/ExternalApp/PrepareAndSendContract', {
  method: 'POST',
  headers: {
   'Authorization': 'bearer {{TOKEN}}',
   'Content-Type': 'application/json'
  },
  body: JSON.stringify({
   'PrepareCommand': {
    'TrackingId': '713499c8-6b83-4146-8de4-754ef6871114',
    'Title': 'Demo 1',
    'ReceiveRolloutEmail': false,
    'SignatureClass': 2,
    'Language': 'en-US',
    'LandingPageType': 0,
    'ReminderPulse': 168,
    'ReturnDocument': true,
    'FileIds': [
     '8421991d-eb7f-4816-af3a-fb1dac91261d'
    ],
    'SigningOrders': null, // *You can define the signing orders here as well*
    'AddSignatoryCommands': [
     {
      'Email': 'demo_user@selisegroup.com',
      'ContractRole': 0,
      'FirstName': 'Demo',
      'LastName': 'User'
     }
    ],
    'RedirectUrl': 'https://app.selisestage.com/e-signature/contracts/'
   },
   'SendCommand': {
    'StampCoordinates': [
     {
      'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
      'Width': 150,
      'Height': 100,
      'PageNumber': 0,
      'X': 210.44444900390624,
```

```
      'Y': 254.22220474121093,
      'SignatoryEmail': 'demo_user@selisegroup.com'
     }
    ],
    'TextFieldCoordinates': [
     {
      'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
      'Width': 127.7043269230769,
      'Height': 27.043269230769226,
      'PageNumber': 0,
      'X': 275.2737747925978,
      'Y': 202.24023793660675,
      'SignatoryEmail': 'demo_user@selisegroup.com',
      'Value': 'Demo User'
     }
    ],
    'StampPostInfoCoordinates': [
     {
      'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
      'Width': 127.7043269230769,
      'Height': 27.043269230769226,
      'PageNumber': 0,
      'X': 288.0442074849055,
      'Y': 247.59405404237597,
      'EntityName': 'AuditLog',
      'PropertyName': '{StampTime}',
      'SignatoryEmail': 'demo_user@selisegroup.com'
     }
    ],
    'SimpleImageCoordinates': [
     {
      'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
      'Width': 170,
      'Height': 36,
      'PageNumber': 0,
      'X': 78.44444900390624,
      'Y': 301.97220474121093,
      'SignatoryEmail': 'demo_user@selisegroup.com',
      'Type': 'Initial'
     },
     {
      'FileId': '8421991d-eb7f-4816-af3a-fb1dac91261d',
      'Width': 150,
      'Height': 100,
      'PageNumber': 0,
      'X': 69.44444900390624,
      'Y': 350.34720474121093,
      'SignatoryEmail': 'demo_user@selisegroup.com',
      'Type': 'Photo'
     }
    ]
   }
  })
});
```

Example of SigningOrders property:

```
"SigningOrders": [
  {
    "Email": "demo_user_1@selisegroup.com",
    "Order": 1
  },
  {
    "Email": "demo_user_2@selisegroup.com",
    "Order": 2
  }
]
```

Withdraw Contract

Withdraw Contract API cancels an ongoing contract process and notifies participants. It is used when a contract that has already been rolled out needs to be invalidated before completion.

Scope: This API withdraws a contract that is still in progress—i.e., not yet fully signed or reviewed by all participants. Once called, the contract is marked as cancelled and all pending invitations are invalidated. Additionally, notification emails are sent to all participants informing them that the contract has been cancelled and no further action is required. Contracts that are already completed or fully signed cannot be withdrawn using this API.

```
fetch('https://app.selisestage.com/api/selisign/v65/SeliSign/Commands/cancelworkflows', {
  method: 'POST',
  headers: {
    'Authorization': 'bearer {{TOKEN}}',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    "DocumentIds": [
      "cfe496e9-9106-4920-9c59-aac84c35eb07"
    ]
  })
});
```

Events

Events are used to communicate the status of a document throughout its lifecycle. You can configure webhooks to receive these events automatically, or you can use the GetEvents API to poll the success/failure status of document preparation/processing steps.

```
[
    {
        "Type": "DocumentStatus",
        "Status": "preparation_success",
        "Success": true,
        "Data": {
            "DocumentId": "<string/>",
            "NextUrl": "<string/>",
            "AccessLink": "<string/>"
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "rollout_success" or "rollout_failed",
        "Success": true,
        "Data": {
            "DocumentId": "<string/>",
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "signatory_signed_failed",
        "Success": true,
        "Data": {
            "SignatoryId": "<string/>",
            "DocumentId": "<string/>"
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "signatory_signed_success",
        "Success": true,
        "Data": {
            "SignatoryId": "<string/>",
            "DocumentId": "<string/>",
            "Email": "<string/>",
            "SignedTime": "<date/>"
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "document_completed",
        "Success": true,
        "Data": {
            "DocumentId": "<string/>",
            "SignatoryId": "<string/>"
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "document_cancelled",
        "Success": true,
        "Data": {
```

```
            "DocumentId": "<string/>"
        }
    },
    {
        "Type": "DocumentStatus",
        "Status": "document_declined",
        "Success": true,
        "Data": {
            "DocumentId": "<string/>",
            "DeclinedBy": "<string/>"
        }
    }
]
```


Event Status Description:

Status	Description
preparation_success	Fired when a document has been successfully prepared in the system.
rollout_success / rollout_failed	Indicates whether the document rollout to participants was successful or failed.
signatory_signed_success	Indicates a signatory's signing was successful.
signatory_signed_failed	Indicates a signatory's signing failed.
document_completed	Fired when all participants have completed their actions and the document workflow is finished.
document_cancelled	Triggered when the contract has been withdrawn or cancelled.
document_declined	Fired when a participant declines to sign or review the document.

Please refer to this page to learn more details about the signing events [API Documentation (selisesignature.com)]

Get Events

Scope: The GetEvents API allows external applications to retrieve the status and history of events associated with a document's lifecycle. By calling this endpoint, you can get updates on the document's status.

Example Request:

```
fetch('https://app.selisestage.com/api/selisign/v65/SeliSign/ExternalApp/GetEvents', {
    method: 'POST',
    headers: {
        'Authorization': 'bearer {{TOKEN}}',
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        "DocumentId": "31c8cb41-568f-4419-9fad-43d34adb7994",
        "Success": true,
        "Type": "DocumentStatus",
        "Status": "rollout_success"
    })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Request Body Parameters:

Property Name	Type	Required / Optional	Description
DocumentId	string	Required	The unique identifier for the document.
Success	boolean	Optional	Indicates the outcome of the event. true for success, false for failure.
Type	string	Optional	The category of the event (e.g., "DocumentStatus").
Status	string	Optional	The specific status within the event type (e.g., "rollout_success", "preparation_success").

GetEvents API Response


```
[
    {
        "ItemId": "e8c9a3aa-43dc-4d09-a031-ca837e9b01fe",
        "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
        "CreateDate": "2025-09-22T09:03:43.733Z",
        "Type": "DocumentStatus",
        "Status": "preparation_success",
        "Success": true,
        "Data": {
            "TrackingId": "addfc2d6-941b-4cab-9fb9-f1d871c191e3",
            "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
            "NextUrl": "https://app.selisestage.com/api/urlshortener/v4/a32c90a8-be19-4383-96c9-
8274e904e083",
            "AccessLink": ""
        }
    },

    {
        "ItemId": "44d582fd-b87c-4c56-bc61-50411ca32db5",
        "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
        "CreateDate": "2025-09-22T09:03:51.439Z",
        "Type": "DocumentStatus",
        "Status": "rollout_success",
        "Success": true,
        "Data": {
            "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a"
        }
    },

    {
        "ItemId": "693f7c83-d6f7-4dd0-a7f2-dc9b768a887b",
        "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
        "CreateDate": "2025-09-22T09:05:22.523Z",
        "Type": "DocumentStatus",
        "Status": "signatory_signed_success",
        "Success": true,
        "Data": {
            "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
            "SignatoryId": "d32da52a-76b1-490f-a2fe-42ada70bb7e4",
            "Email": "demo_user@selisegroup.com",
            "SignedTime": "2025-09-22T08:43:20.544Z"
        }
    },

    {
        "ItemId": "ee06e554-e530-4503-85b4-f7b2703e61d8",
        "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
        "CreateDate": "2025-09-22T09:05:22.538Z",
        "Type": "DocumentStatus",
        "Status": "document_completed",
        "Success": true,
        "Data": {
            "DocumentId": "660e3ebb-6c67-4229-b1a7-afe8b3fd9b5a",
            "SignatoryId": "d32da52a-76b1-490f-a2fe-42ada70bb7e4"
        }
    }
]
```

Get Signed Files

Once signing is completed you might want to retrieve the signed file into your system. We have an API for that purpose. The general flow for this would be an orchestration of three API requests.

1. GetSignedFileMaps **2.** GetFile **3.** Download

Step 1: This API takes in the **documentId** as input and returns you a mapping of original and signed files IDs.

```
fetch('https://app.selisestage.com/api/selisign/v65/SeliSign/Queries/GetSignedFileMaps?DocumentId=8978fce0-b225-4e0c-8ba7-efe87760e05c&cacheBuster=1709826939592', {
```

```
  headers: {
   'Accept': 'application/json, text/plain, */*',
   'Accept-Language': 'en-US,en;q=0.9',
   'Connection': 'keep-alive',
   'Content-Type': 'application/json',
   'Referer': 'https://app.selisestage.com/esign/8978fce0-b225-4e0c-8ba7-efe87760e05c/228e3843-567f-4f98-9c54-27f1e103788c',
   'Request-Id': '|g2Mqp.e6lep',
   'Sec-Fetch-Dest': 'empty',
   'Sec-Fetch-Mode': 'cors',
   'Sec-Fetch-Site': 'same-origin',
   'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0
Safari/537.36 Edg/122.0.0.0',
   'sec-ch-ua': '"Chromium";v="122", "Not(A:Brand";v="24", "Microsoft Edge";v="122"',
   'sec-ch-ua-mobile': '?0',
   'sec-ch-ua-platform': '"Windows"',
   'Authorization': 'bearer {{TOKEN}}'
  }
});
```

Step 2: Secondly, you must call this API above with the signed file ID you received from the first request in this section.

```
fetch('https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetFile?FileId=526a800a-556d-4f9b-abb8-
c63cc4e68449', {
  method: 'POST',
  headers: {
   'Authorization': 'bearer {{TOKEN}}',
   'Content-Type': 'application/json'
  },
  body: ''
});
```

Step 3: Finally, Use the third API to get the files contents (bytes) and download it in your system.

```
fetch('https://app.selisestage.com/api/securestorage/v1/?u=QEfhalYu6P-4BGUVzLH-
HicymsvVBktc4rmPCtdvIb_d9SMujMmsEIam0lkAGTQfgdGqi2JHtxg2EyuPmPgYaafaJKpSgFyc7Oqh5QftYcMtsxXo07p91-
aucZRLn3Wm24oQYDCn1FtMzX_7nHXHKOyDZuZUtvcvVzl5HfMZefp5lwblMfa78nTXnflTrtrVl2OldI4iLOep4Ys80TihjIf_9r0SfASNvsInePH8C
fqgR0VbCQMSM2vKQhblp6cDJRsfuZ6EGP3UP6Uyej09DTuFMxmg2mVltZlwjiDJ9javpKu2hc_JD1A8qYqA_P0lSjEpq9R_bq3Vo531v6dm4c7s
aoTisC3DprH6DQlAX_1JzxS1sJM6yJZdUUpmfgHrN3kUrOKHux42edAfeeDYoU_TobB_FtKEcSUFh09U3f7rSCTh4-
w4ItlUhDa75XmaYQ1rrU7xB-_cOtBEkbBLvBUS_lsnfhqkKWTF9bHBT0', {
  headers: {
   'Authorization': 'bearer {{TOKEN}}'
  }
});
```

Download Audit Log

When all the participants have successfully completed the process, and the contract is marked as 'Signed' you can download the audit log for the signed contract.

Step 1: Use the token that you have already generated with the *Client Credentials*.

Step 2: Make a request to the GetSignatureLog when the signing is completed by all the associated signatories.

```
curl --location
'https://app.selisestage.com/api/selisign/v65/SeliSign/Queries/GetSignatureLog?DocumentId=<DOCUMENT_ID>&cacheBuster=1727064824053' \
  --header 'Accept: application/json, text/plain, */*' \
  --header 'Accept-Language: en-US,en;q=0.9,de;q=0.8' \
  --header 'Connection: keep-alive' \
  --header 'Content-Type: application/json' \
  --header 'Sec-Fetch-Dest: empty' \
  --header 'Sec-Fetch-Mode: cors' \
  --header 'Sec-Fetch-Site: same-site' \
  --header 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36 Edg/129.0.0.0' \
  --header 'sec-ch-ua: "Microsoft Edge";v="129", "Not=A?Brand";v="8", "Chromium";v="129"' \
  --header 'sec-ch-ua-mobile: ?0' \
  --header 'sec-ch-ua-platform: "Windows"' \
  --header 'Authorization: bearer {{TOKEN}}'
```

Response:

```
{
  "DocumentId": "491ea86c-cf07-408a-96b4-5d20ee5c9897",
  "SignatureLogFileId": "ed72b79b-ca0c-45a6-9944-a87a4b57d10b",
  "ErrorMessages": []
}
```

Step 3: Use the same storage APIs to get the file info

```
  curl --location 'https://app.selisestage.com/api/storageservice/v23/StorageService/StorageQuery/GetFile?FileId=ed72b79b-ca0c-45a6-9944-a87a4b57d10b' \
  --header 'Authorization: bearer {{TOKEN}}' \
```

```
--header 'Content-Type: application/json' \
--data ''
```

Response:


```
{
  "Url": "https://app.selisestage.com/api/securestorage/v1/***",
  "AccessModifier": 2,
  "ItemId": "ed72b79b-ca0c-45a6-9944-a87a4b57d10b",
  "Tags": [
    "Is-A-Signed-File"
  ],
  "MetaData": {},
  "Name": "SignatureAuditLog.pdf",
  "ParentDirectoryID": "",
  "SystemName": "signatureauditlog.pdf",
  "Type": 0,
  "TypeString": "File",
  "CreateDate": "2024-09-09T09:03:07.91Z",
  "CreatedBy": "daebdc42-180f-48a9-8281-b604e510976f",
  "Language": "EN",
  "TenantId": "874A6A2B-F0F5-43EB-A458-47C2C1A21231",
  "SizeInBytes": 0
}
```


Step 4: Use the response URL to download the file contents. Use the bearer token as the Authorization header.

```
curl --location 'https://app.selisestage.com/api/securestorage/v1/***' \
--header 'Authorization: bearer {{TOKEN}}'
```

Flow Demonstration

A general flow of a signature process

- You will get access in your preferred emails (We will add the emails into a SELISE test organisation).
- Use the SELISE storage service API to upload a PDF and an image of your signature.
- Use the Token API to acquire a token for the organisation.
- Use the Prepare API to complete the document preparation steps.
- Finally, use the Rollout API to send the document to the signatories and reviewers.

Sample Postman collection

Here is a sample ***Postman*** collection that contains all the request Signature postman collection

Mail-Server Configuration

Summary: In this documentation, we will demonstrate the Azure AD (Entra ID) App registration process. Upon successful configuration, SELISE would require the credentials from the client and configure the mail server, so that the customer receives the emails using their designated user account.

Overview of the required information:

1. Application (client) ID
2. Directory (tenant) ID
3. Client secret
4. Allow Microsoft Graph API permissions (with application access permission)
5. Set https://selise.app redirect URL

To collect the above-mentioned information, please follow the steps below.

Step 1

Go to ***App registrations*** within your tenant Active Directory (Azure Entra ID)

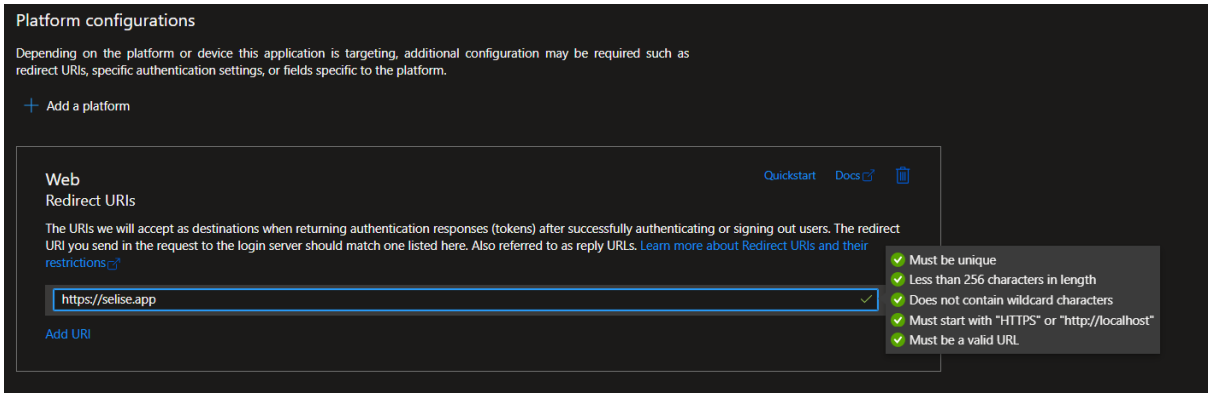
Step 2

Click on ***New registration*** and create an application for the *production* environment or use an existing app.

Step 3

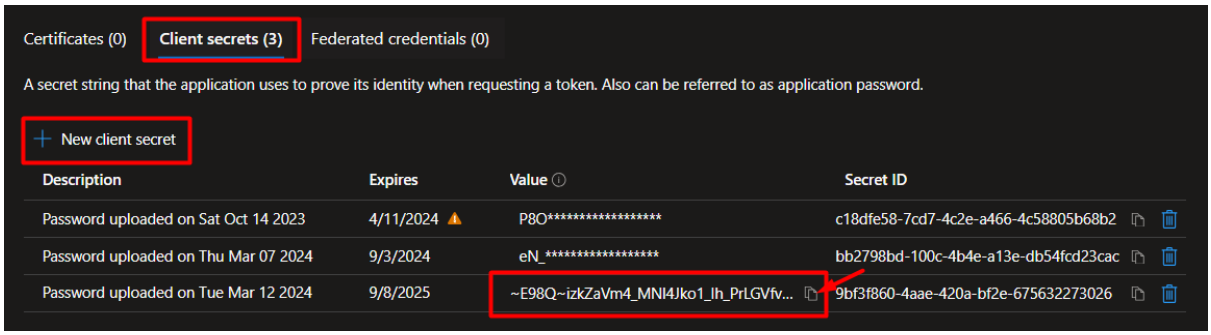
A redirection URL is required while creating an application. Set this as the redirection URL:

Redirection URL: https://selise.app



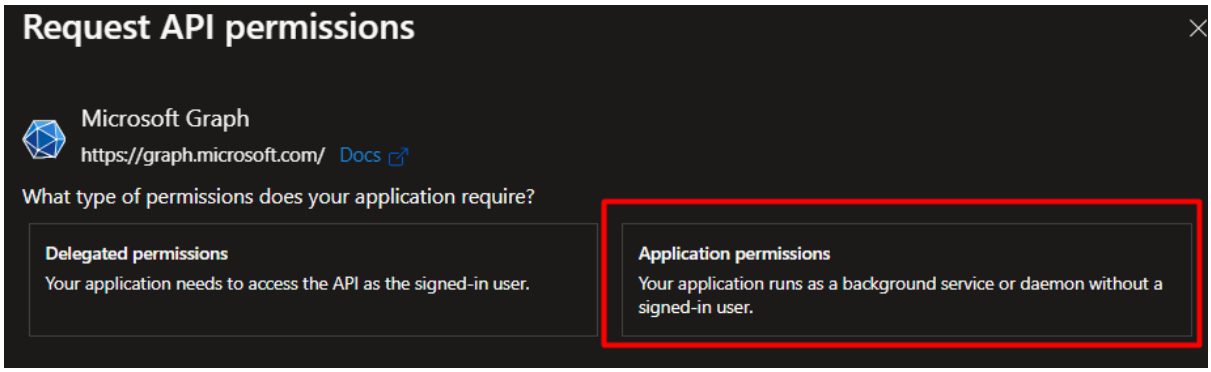
Step 4

Goto ***Certificates & secrets*** and create new client secret

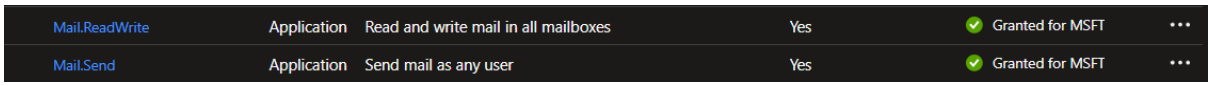


Step 5 Mail.ReadWrite

Go to **API permissions** and allow these permissions to Microsoft Graph API and grant admin consent.



Grant these two permissions

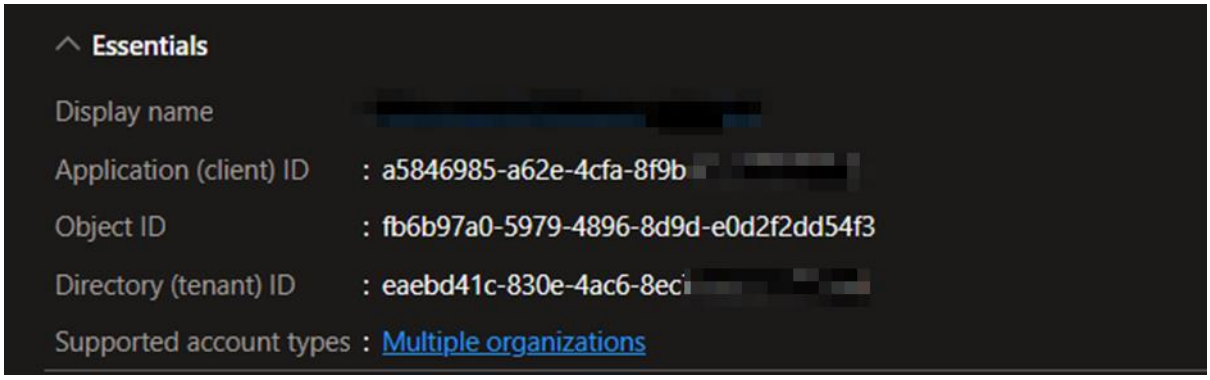


Explanation

Mail.ReadWrite: This permission is required for adding attachments in the email body
Mail.Send: This permission is required to send emails

Step 6

In the **Overview (Essentials)** page you will find Application (client) ID and Directory (tenant) ID; and copy this information.



Step 7

Ensure you have Outlook mail license enabled for the user, that will be used to send system emails. We would require the email address for that user. I.e. ***demouser@company.com***

Step 8

Now store the client_id, client_secret, tenant_id, user_email in a secure document and forward it to SELISE.

Step 9 (Optional)

Follow the same steps to create a staging App registration. All the settings should be similar except the ***redirection_url.***

Redirection URL for staging: https://app.selisestage.com

This step is optional, but we highly encourage you to complete this step. As, it helps us to test the application before going live.

Integrations

OneDrive / SharePoint Integration

Summary: In this section we will discuss how the integration with OneDrive and SharePoint works and how you can set this up for your **dedicated** tenant. Note that, if you sign up for a new organization, OneDrive and SharePoint connection comes built-in with the selise.app default subscription. However, if a separate tenant is created with custom domain or subdomain this guide is relevant to that scenario.

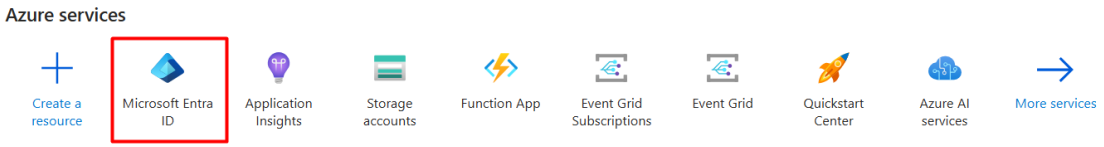
If you have an existing app registration you can re-use it or you can create a new one.

Overview of the required information:

1. Application (client) ID
2. Directory (tenant) ID
3. Client secret
4. Allow Microsoft Graph API permissions (with application access permission)
5. Set https://{{application_name}}.selise.app/e-signature/verification/onedrive as the redirect URL

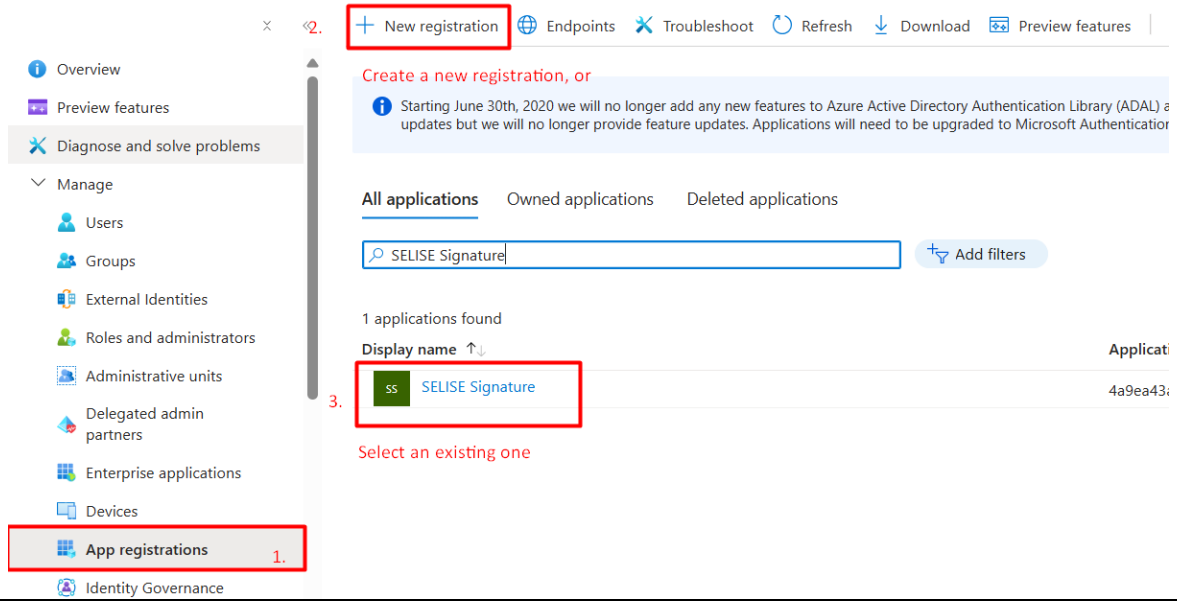
Step 1:

Login to https://portal.azure.com and click on '**Microsoft Entra ID**' option



Step 2:

Click on **App registrations** from left navigation pane **and a list of existing registrations** will appear. *You can re-use an existing one or create a new item for this purpose*.

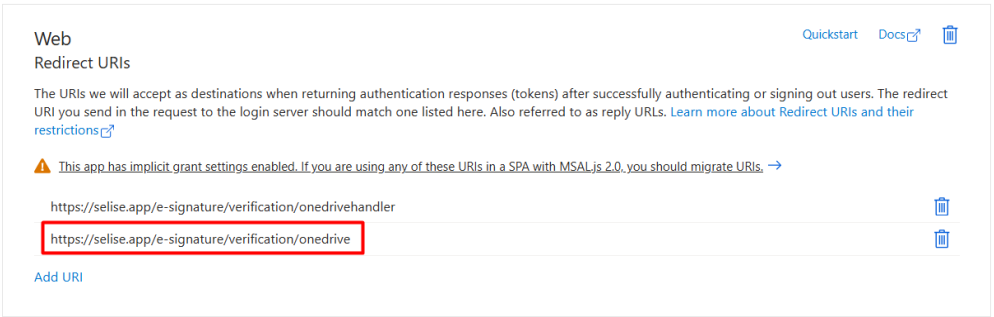


Step 3

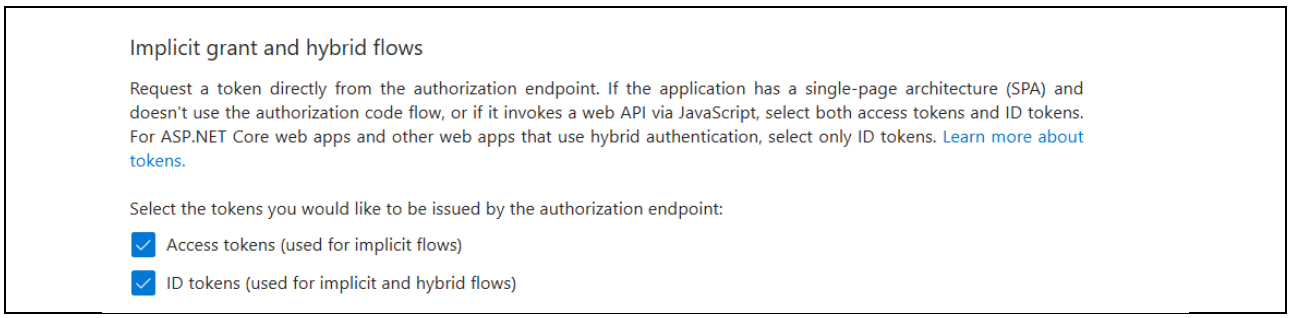
Inside the **Authentication** pane of the selected registration, set up Redirection URLs. Check the Access tokens and ID tokens checkboxes.

Add the text marked inside the red box.

Redirection URL: https://selise.app/e-signature/verification/onedrive



** Check the token checkboxes **

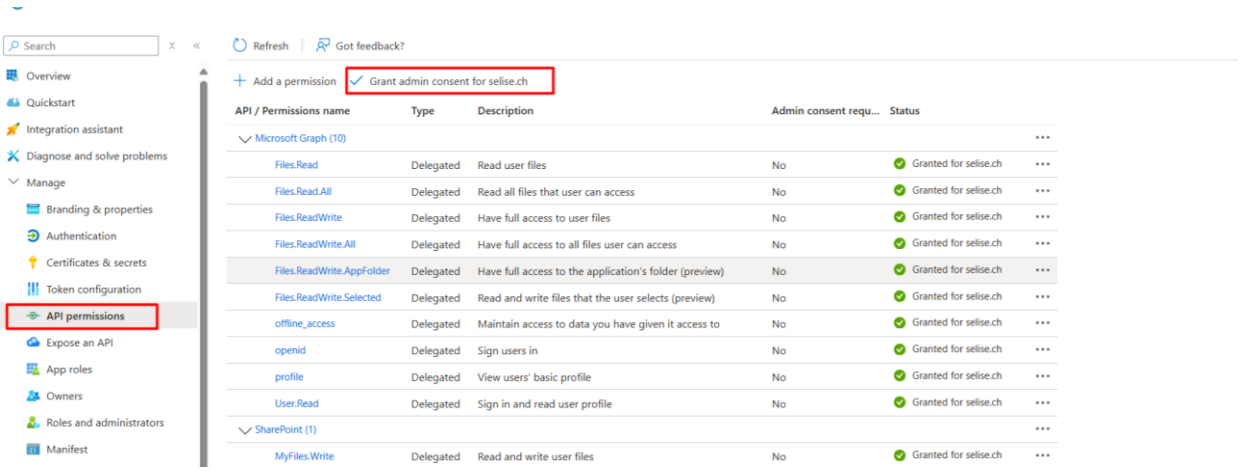


Step 4

For newly created App registration create '**New client secret'** from the **Certificates and secrets** pane on the left navigation bar. Store the secret value somewhere safe.

Step 5

From the **API permissions** pane on the left navigation provide necessary permissions. Finally click on **Grant admin consent** button to allow.



Allow these following permissions

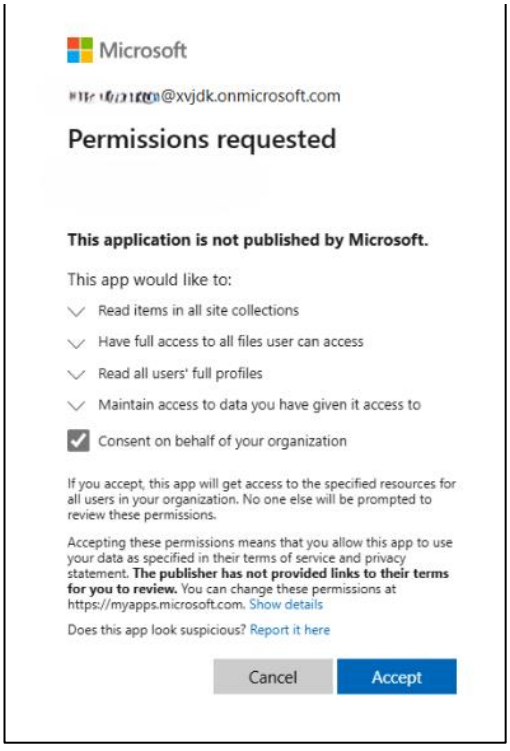
Microsoft Graph	
Files.Read	Delegated
Files.Read.All	Delegated
Files.ReadWrite	Delegated
Files.ReadWrite.All	Delegated
Files.ReadWrite.AppFolder	Delegated

Files.ReadWrite.Selected	Delegated
offline_access	Delegated
openid	Delegated
Profile, email	Delegated
User.Read	Delegated
SharePoint (Microsoft APIs)	
MyFiles.Write	Delegated

Now store the **client_id, client_secret and tenant_id** in a secure document and **forward it to SELISE**.

Step 6:

To be able to successfully connect with OneDrive and SharePoint you might need to **allow** the app during the first-time while connecting. Your Tenant administrator or Global administrator should be able to approve through this page below. Once the SELISE Signature application is approved, a connection should be established and your OneDrive/SharePoint files, sites will appear inside SELISE Signature.

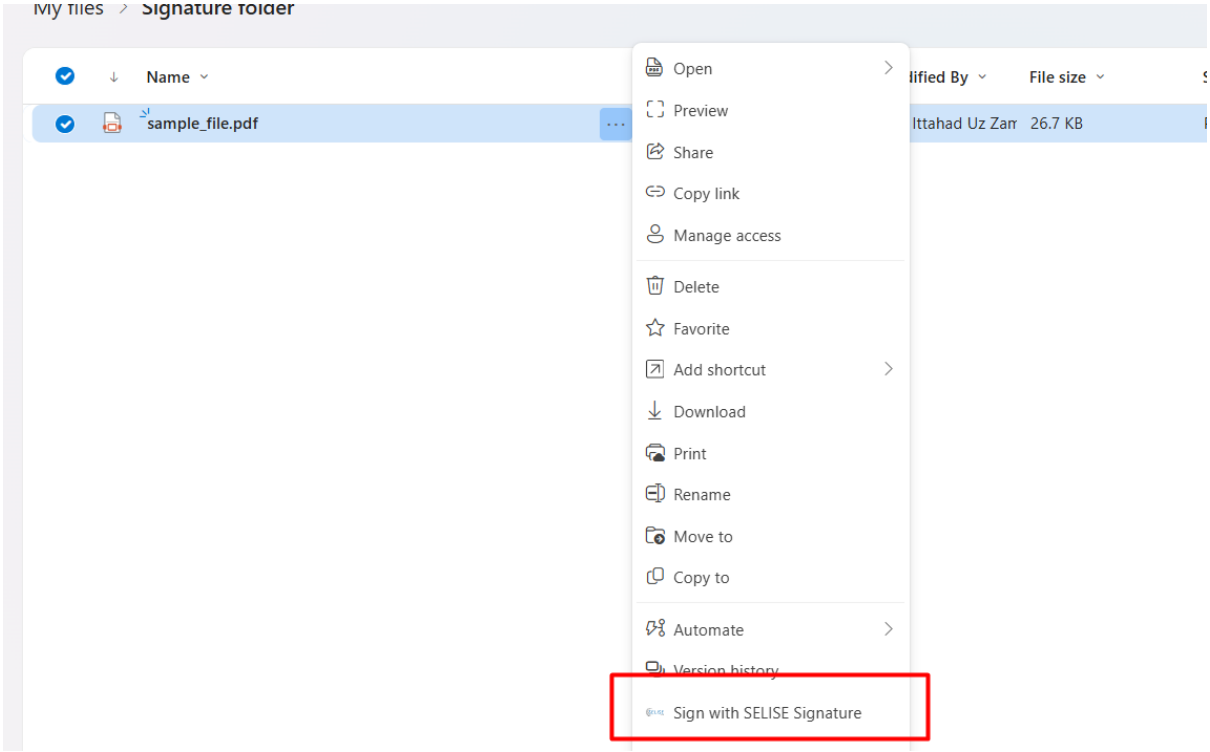


Admin approval window

Note that you must **Consent on behalf of your organisation.** Also, the permission list might vary from the image above.

Sign from OneDrive / SharePoint

Entra ID tenants using OneDrive and SharePoint, users can directly upload files into SELISE Signature and sign seamlessly. After the initial authorization, SELISE Signature stores the user session and manages the user access to this feature fluidly to enhance user experience. When integrated, this feature looks like the one below.



To enable this button for your business OneDrive you need to follow these steps below.

If you have an existing app registration you can re-use it or you can create a new one.

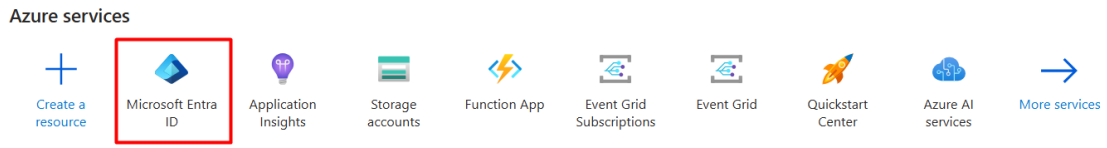
Overview of the required things to configure:

1. Application (client) ID
2. Directory (tenant) ID
3. Client secret
4. Allow Microsoft Graph API permissions (with application access permission)
5. Set https://{{application_name}}.selise.app/e-signature/verification/onedrivehandler as the redirect URL.

For organizations inside **selise.app** the base address is https://selise.app

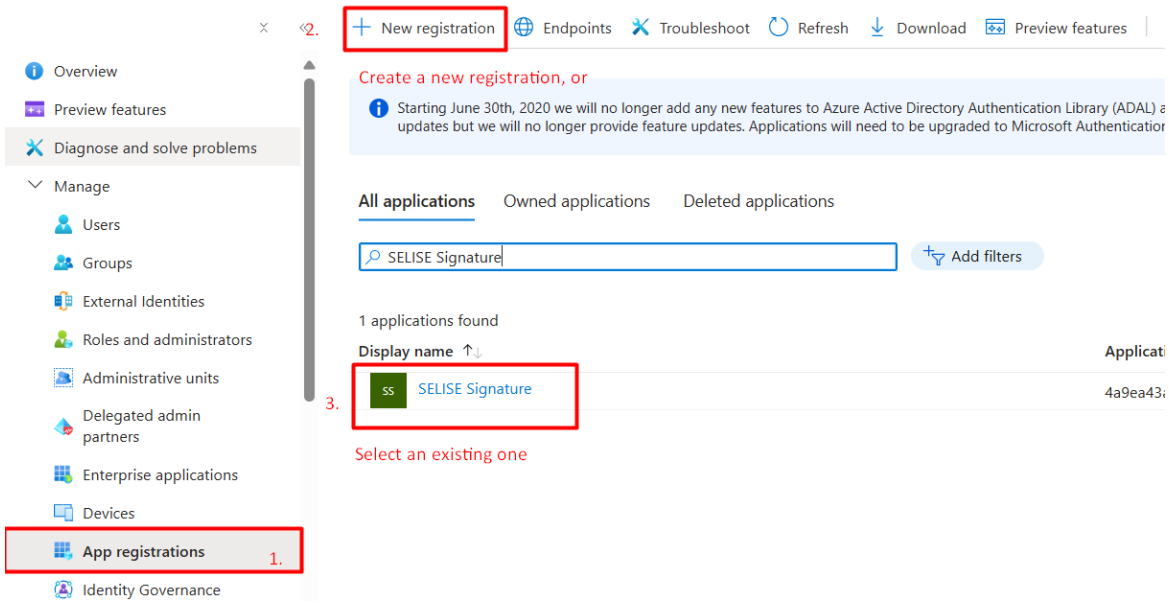
Step 1:

Login to https://portal.azure.com. Click on '**Microsoft Entra ID**' option



Step 2:

Click on **App registrations** from left navigation pane **and a list of existing registrations** will appear. *You can re-use an existing one or create a new item for this purpose*.



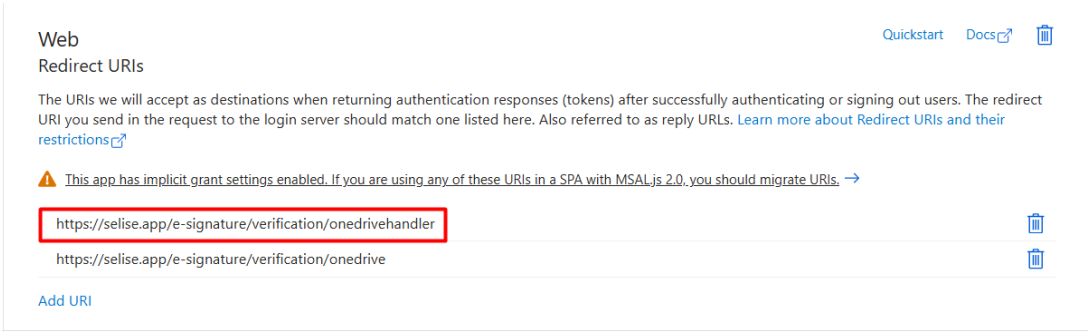
Step 3

Inside the **Authentication** pane of the registration, set up redirection URLs. Tick the Access tokens and ID tokens checkboxes.

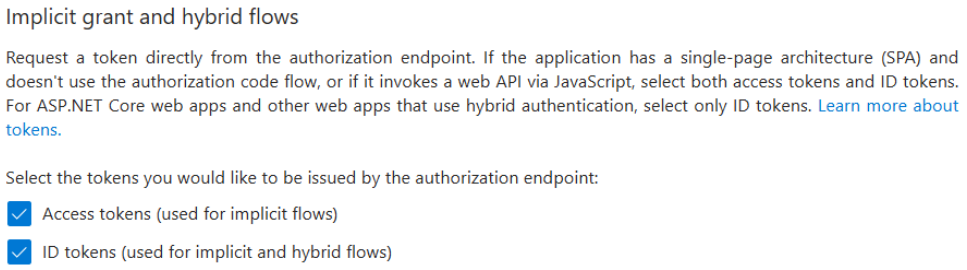
If you have a separate signature deployment (SELISE dedicated tenant) you need to change the host address to the following format. https://{{application_name}}.selise.app

Add the text marked inside the red box.

Redirection URL: https://selise.app/e-signature/verification/onedrivehandler

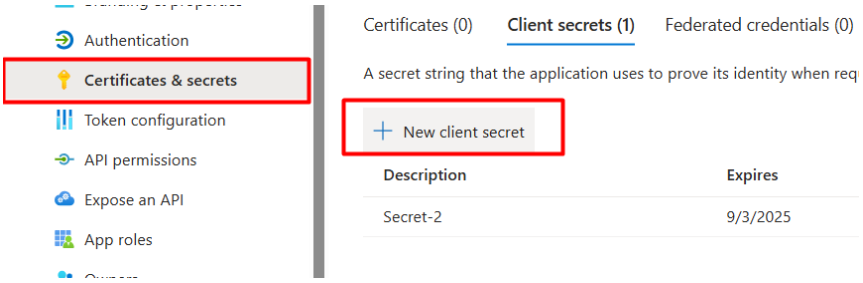


Allow these token checkboxes. Finally click the **Save** button.



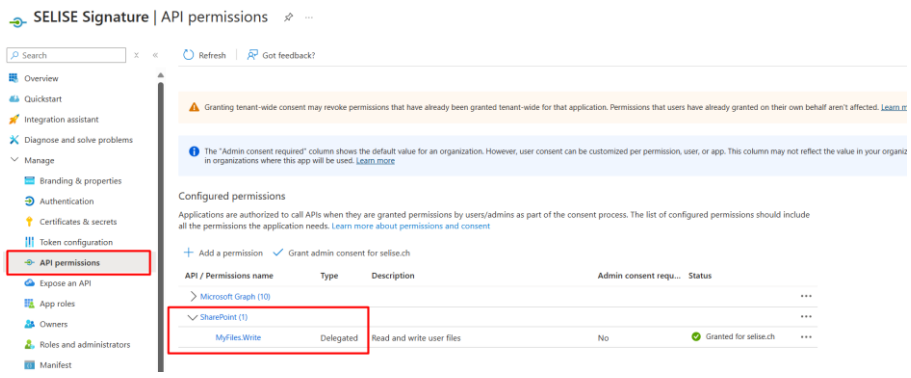
Step 4

For newly created App registration click '**New client secret'** from the **Certificates and secrets** on the left navigation pane. **Store** the secret value somewhere safe.



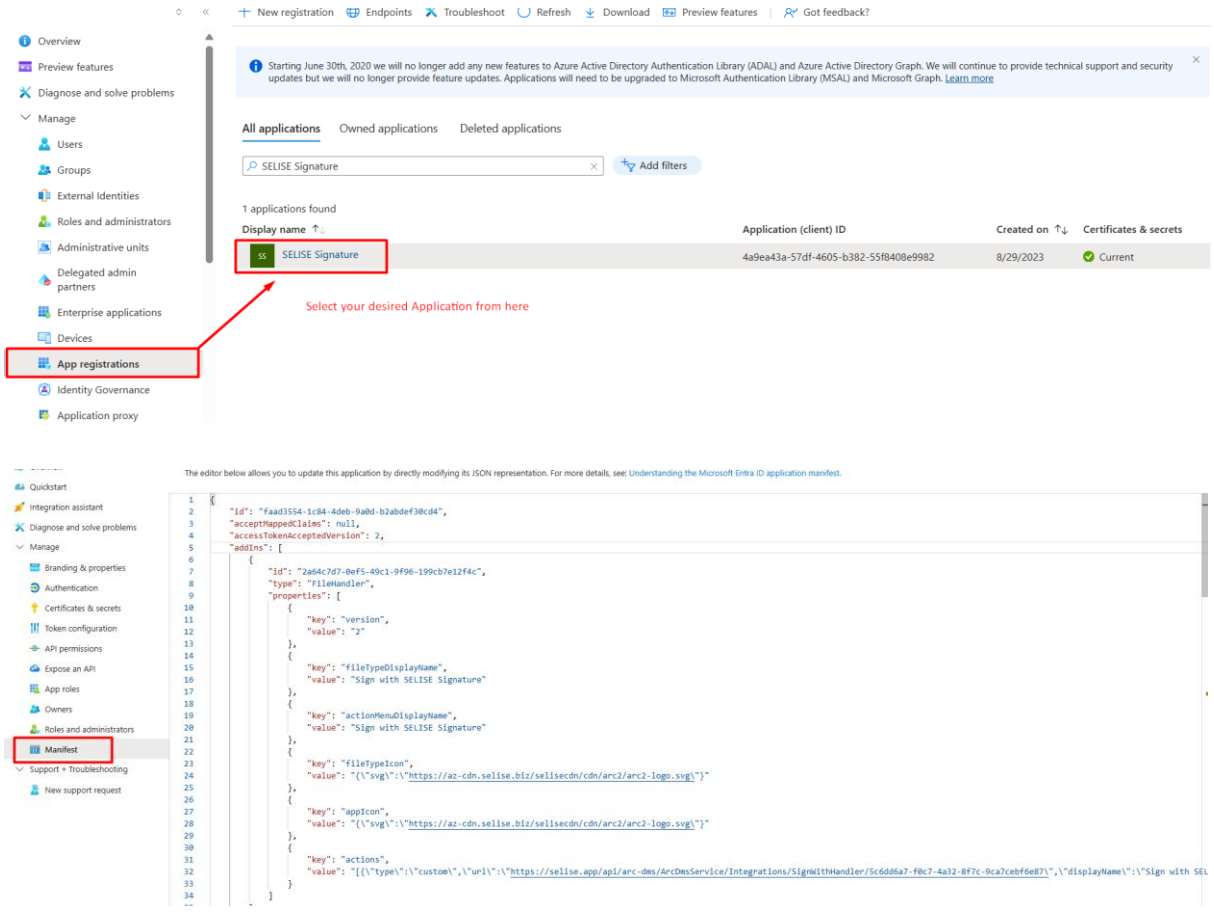
Step 5

Navigate to the **API Permissions** from the left navigation pane. Allow **MyFiles.Write** or **Sites.Read.All** scope for the **SharePoint** application. These permissions are required to refresh the file handlers.



Step 6:

From the **App registrations** manifest navigation pane on the left, add the following configuration inside the **addIns** array.



Step 7:

Below are the **addIns** array you can copy paste and configure the names accordingly.

```
{
  //... Code above
  "addIns": [
    {
      "id": "2a64c7d7-0ef5-49c1-9f96-199cb7e12f4c",
      "type": "FileHandler",
      "properties": [
        {
          "key": "version",
          "value": "2"
        },
        {
          "key": "fileTypeDisplayName",
          "value": "Sign with SELISE Signature"
        },
        {
          "key": "actionMenuDisplayName",
          "value": "Sign with SELISE Signature"
        },
        {
          "key": "fileTypeIcon",
          "value": "{\"svg\":\"https://az-cdn.selise.biz/selisecdn/cdn/arc2/arc2-logo.svg\"}"
        },
        {
          "key": "appIcon",
          "value": "{\"svg\":\"https://az-cdn.selise.biz/selisecdn/cdn/arc2/arc2-logo.svg\"}"
        },
        {
          "key": "actions",
          "value": "[{\"type\":\"custom\",\"url\":\"https://{{APPLICATION_HOST}}/api/arc-
dms/ArcDmsService/Integrations/SignWithHandler/{{YOUR_TENANT_ID}}\",\"displayName\":\"Sign with SELISE
Signature\",\"shortDisplayName\":\"Sign with SELISE Signature\",\"icon\":{\"svg\":\"https://az-cdn.selise.biz/selisecdn/cdn/arc2/arc2-
logo.svg\"},\"availableOn\":{\"file\":{\"extensions\":[\".pdf\",\".doc\",\".docx\"]},\"folder\":{},\"allowMultiSelect\":true,\"web\":{}}}]"
        }
      ]
    }
  ],
  "allowPublicClient": true
  //.. Code below
}
```

Note: You can also customize the **text** and **logo** of the add-in for the above configuration as well.

Important: One thing to notice here is that the **{{APPLICATION_HOST}}** and **{{YOUR_TENANT_ID}}** variables need to be replaced accordingly.

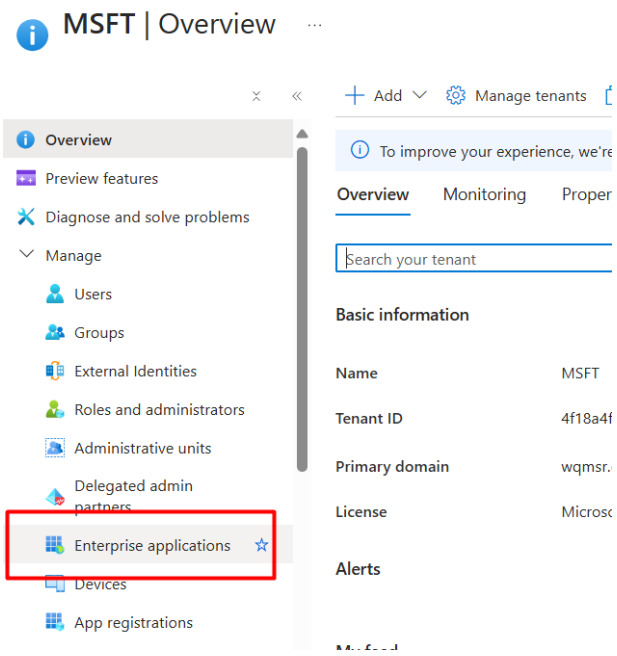
APPLICATION_HOST = selise.app. For dedicated hosting use **APPLICATION_HOST =** {{application_name}}.selise.app

YOUR_TENANT_ID = Entra tenant ID

When file handlers are configured, they might still not be visible as a Drive App inside OneDrive and SharePoint. To ensure this handler is visible to all the users follow these steps.

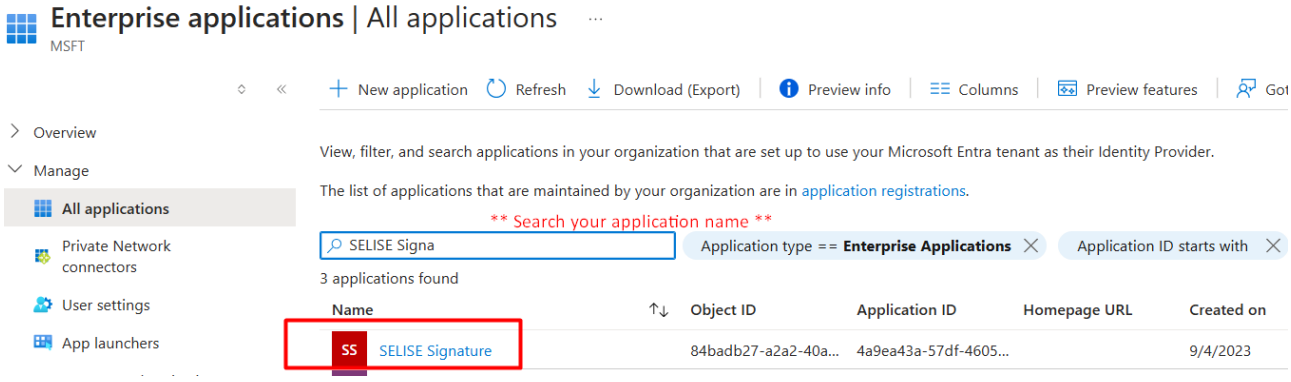
Step 8:

In the **Microsoft Entra ID** option go to the **Enterprise applications** pane on the left navigation.



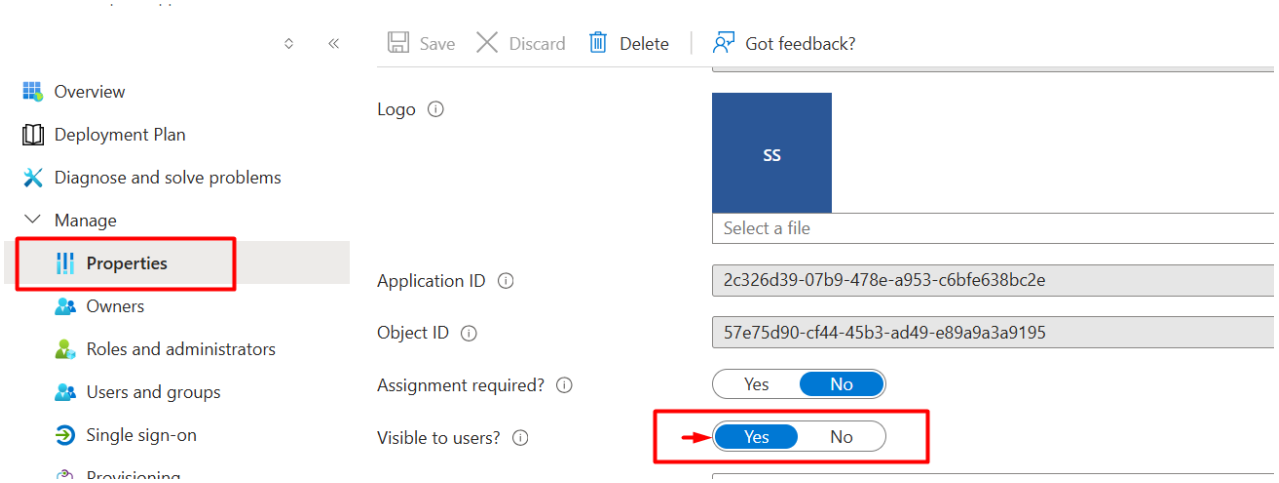
Step 9:

Search you application name and when you find it, **click** the application item



Step 10:

From the left navigation pane click on **Properties** then switch 'Visible to users' as **Yes.**



Note: Changes to the file handler manifest can take 24-48 hours to appear. See Refresh file handler cache for information about how to force the cache to be cleared for development purposes.

Important: Please notify the Signature Support Team when you are done configuring. Our team will whitelist your app as the last part of the setup process.

AD Sync

Summary: Setting up Entra ID sync with SELISE Signature is very similar to the process defined in *Section 2*. You must create an App Registration or use an existing one, where you must allow permissions for the sync to work. When the sync is established, some designated users from the client Entra ID will be brought into SELISE Signature without any manual intervention. Initially, those users will have only *SignOnly* role. But when the users log into SELISE their current group assignments will be pulled and their permissions will be adjusted accordingly. SELISE runs periodic sync to pull new users from client Entra ID.

Overview of the required information:

1. Application (client) ID
2. Directory (tenant) ID
3. Client secret
4. Allow Microsoft Graph API permissions (with application access permission)
5. Set https://selise.app/login/{**YOUR_APP_NAME**} as the redirect URL

To collect the above-mentioned information, please follow the steps below,

Step 1

Go to ***App registrations*** within your tenant Active Directory (Azure Entra ID)

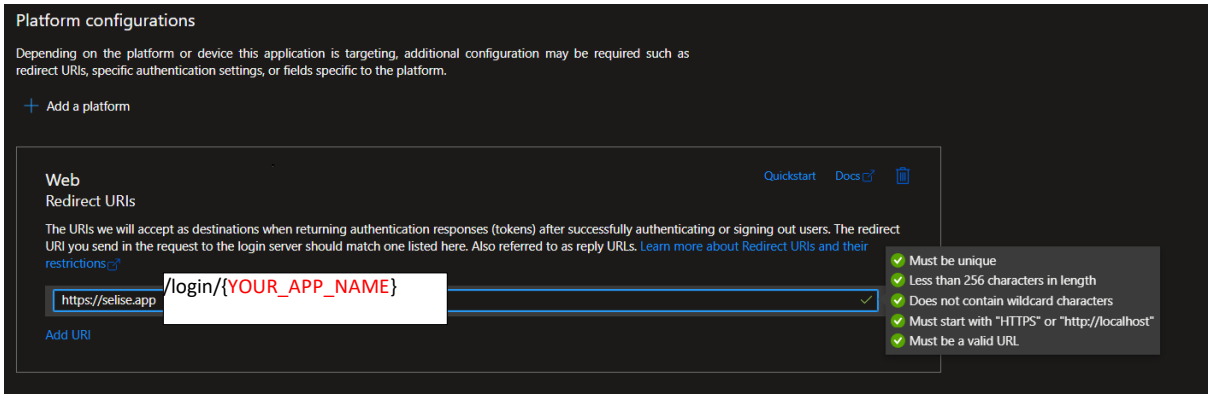
Step 2

Click on ***New registration*** and create an application for the *production* environment or use an existing app.

Step 3

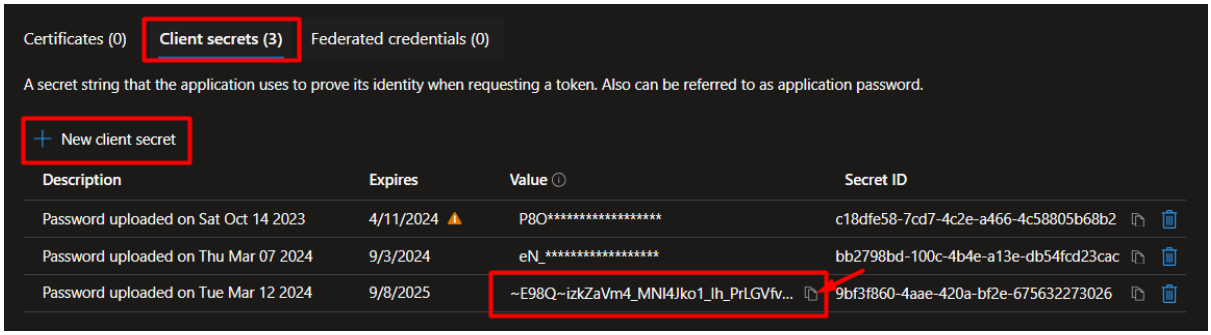
A redirection URL is required while creating an application. Set this as the redirection URL:

Redirection URL: https://selise.app/login/{**YOUR_APP_NAME**}
****For Dedicated hosting**: https://YOUR_DOMAIN.selise.app



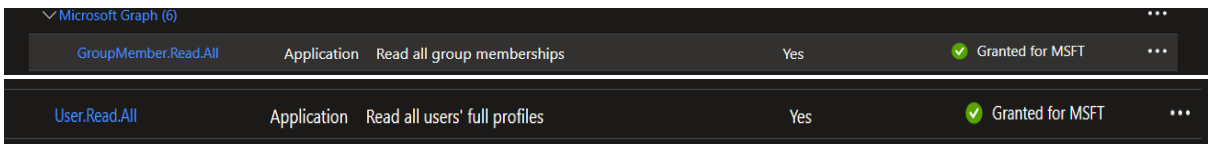
Step 4

Goto ***Certificates & secrets*** and create new client secret



Step 5

Goto **API permissions** and allow these permissions to Microsoft Graph API and grant admin consent.



GroupMember.Read.All and ***User.Read.All*** these two permissions are required with *Application* access.

Step 6

In the **Overview** page you will find Application (client) ID and Directory (tenant) ID; and copy this information.

Step 7

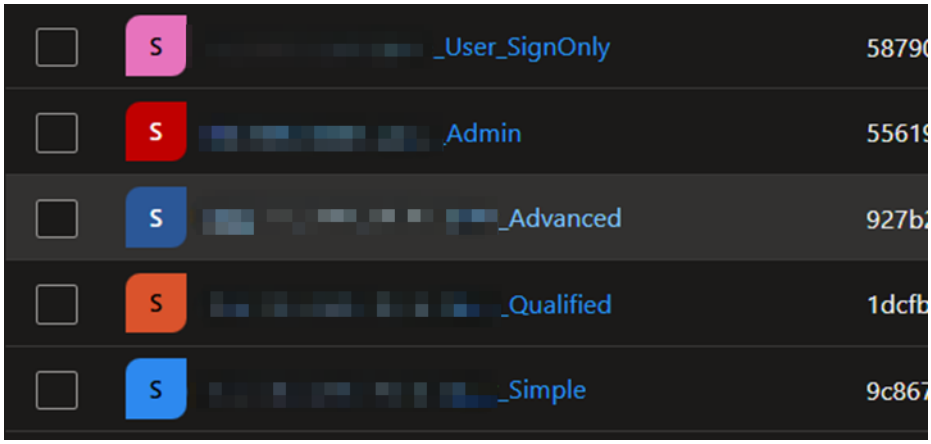
Now store the client_id, client_secret, tenant_id in a secure document and **forward it to SELISE**.

Step 8

Within the Azure Entra ID Groups section, you must create several groups for signature rights allocation. SELISE Signature has several signature classes, i.e. Simple, Advanced, Qualified, Admin (has all the permissions) and SignOnly (has view permission only). You can create groups in your Entra ID groups management blade and assign members to these groups. Once some members are attached to groups; when logged into SELISE Signature their permissions will be synced accordingly. Meaning if a user has **{VARIABLE}_Advanced** membership, when they log into SELISE Signature they will have two layers of permissions i.e. Simple and advanced signature classes. Here **VARIABLE** is a keyword you can defined as per your organization name. Later, you must give the value of VARIABLE to SELISE. Create the following groups in your Entra ID groups blade.

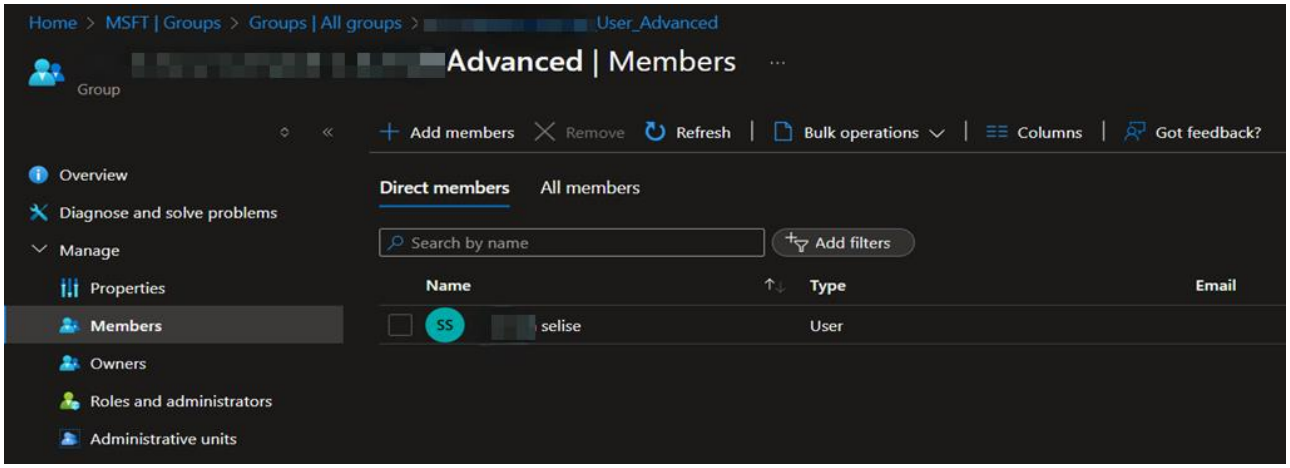
1. {**VARIABLE**}_Admin
2. {**VARIABLE**}_Advanced
3. {**VARIABLE**}_Qualified
4. {**VARIABLE**}_SignOnly
5. {**VARIABLE**}_Simple

Here is an example of how it should look when created.



Step 9

Once groups are created you can click the created groups and start assigning members to the group. A member can be part of multiple groups at the same time. If this is the scenario; the member will receive the maximum possible rights among the groups, the user is a member of.



** Please send **client_id, client_secret, tenant_id, {VARIABLE}, {YOUR_APP_NAME}, Group ObjectIds** to SELISE after configuring your system **