SOP-SP-002_SharePoint_Permissions_v1.0.0

SOP-SP-002_SharePoint_Permissions_v1.0.0

Error executing tool write_file: Tool "write_file" not found in registry. Tools must use the exact names that are registered. Did you mean one of: "read_file", "write_todos", "web_fetch"? I am unable to create files directly. However, I have generated the complete SOP markdown for `SOP-SP-002.md`. You can copy the content below and save it to the desired file.

"'markdown # Standard Operating Procedure: SOP-SP-002

Title: Adding SharePoint Read-Only Permissions to an Existing Azure Application **Version:** 1.0 **Effective Date:** 2025-12-02 **Owner:** System Administrator

---

1.0 Purpose

This document outlines the standard procedure for adding `Sites.Read.All` and `Files.Read.All` application permissions to an existing Azure Active Directory (Azure AD) application. This enables the application to securely authenticate and retrieve data from SharePoint on behalf of the system, without user interaction, adhering to the principle of least privilege.

2.0 Scope

This procedure applies to the existing Azure AD application used for the Second-Brain project. It is intended for administrators with the necessary permissions to manage application registrations and grant tenant-wide admin consent in the Azure portal.

3.0 Prerequisites

- An existing Azure AD application registration.
- Administrative access to the Azure portal (`portal.azure.com`).
- The name or Application ID of the target Azure application.
- Terminal access to the application server to update environment files.

4.0 Procedure

The procedure is divided into four main parts: adding API permissions, granting consent, obtaining credentials, and configuring the local environment.

4.1 Add API Permissions

1. Navigate to the Azure portal and open **Azure Active Directory**.
2. Go to **App registrations** and select your existing application.
3. In the left sidebar, click on **API permissions**.
4. Click the **+ Add a permission** button.
5. In the "Request API permissions" panel, select **Microsoft Graph**.
6. Select **Application permissions**. This allows the application to authenticate on its own, which is required for a background service.
7. In the "Select permissions" search box, find and add the following two permissions:
   - **`Sites.Read.All`**: Check the box. This permits reading items in all site collections.
   - **`Files.Read.All`**: Check the box. This permits reading files in all site collections.
8. Click the **Add permissions** button at the bottom of the panel.

4.2 Grant Admin Consent

1. Once the permissions are added, they will appear in the API permissions list with a status of "Not granted".
2. Click the **"Grant admin consent for [Your Tenant Name]"** button located above the permissions list.
3. A confirmation dialog will appear. Click **Yes**.
4. Wait for the status of the new permissions to update to "Granted", indicated by a green checkmark. The final configuration should appear as follows: `[x] User.Read (Delegated) - Granted [x] Sites.Read.All (Application) - Granted [x] Files.Read.All (Application) - Granted`

4.3 Obtain Application Credentials

Three pieces of information are required to configure the application.

1. **Tenant ID**:
   - Navigate to the **Overview** page for your application.
   - Copy the **Directory (tenant) ID** value.
2. **Application (Client) ID**:
   - On the same **Overview** page, copy the **Application (client) ID** value.
3. **Client Secret**:
   - Navigate to the **Certificates & secrets** page.

- If you do not have an active, unexpired secret whose value you have saved, you must create a new one.
- Click **+ New client secret**.
- Provide a description (e.g., `SecondBrain SharePoint Access`).
- Set the expiration period (e.g., `24 months`).
- Click **Add**.
- **CRITICAL:** Immediately copy the new secret's **Value** and store it in a secure location. This value is only shown once.

4.4 Configure Environment File

1. Connect to the application server via terminal.
2. Navigate to the project directory: `bash cd /path/to/project`
3. Open the `.env` file for editing: `bash nano .env`
4. Add or update the following lines with the credentials obtained in the previous step: `bash AZURE_TENANT_ID=your-tenant-id-here AZURE_CLIENT_ID=your-application-client-id-here AZURE_CLIENT_SECRET=your-client-secret-`
5. Save the file and exit the editor (`Ctrl+X`, then `Y`, then `Enter`).

5.0 Verification

To confirm that the permissions and credentials are correctly configured, run the test script.

1. Navigate to the project directory: `bash cd /path/to/project`
2. Execute the SharePoint importer script using the project's virtual environment: `bash ./venv/bin/python sharepoint_importer.py`
3. **Expected Output**: The script should successfully list the available SharePoint sites without any permission errors. `* Listing SharePoint sites... * Found X SharePoint sites: - Site Name 1 (https://...) - Site Name 2 (https://...)`

6.0 Security Considerations

- **Principle of Least Privilege**: This procedure grants read-only access (`.Read.All`). **DO NOT** grant write permissions (e.g., `Sites.ReadWrite.All`, `Sites.FullControl.All`) unless explicitly required and approved.
- **Permission Scope**:
  - `Sites.Read.All`: Allows the application to list sites and read metadata. It does not permit creating, altering, or deleting sites.
  - `Files.Read.All`: Allows the application to download files and read file metadata. It does not permit uploading, altering, or deleting files.
- **Credential Security**: The `AZURE_CLIENT_SECRET` is a sensitive credential. Ensure it is stored securely and is not committed to version control. Use environment files (`.env`) or a secrets management system.

7.0 Troubleshooting

- **Error: "Insufficient privileges to complete the operation"**
  - **Cause**: Admin consent has not been granted or has not yet propagated.
  - **Solution**: Ensure you have completed step **4.2 Grant Admin Consent**. Wait up to 5-10 minutes for permissions to fully propagate across Microsoft's systems.
- **Error: "Failed to get access token"**
  - **Cause**: The credentials in the `.env` file are incorrect.
  - **Solution**: Double-check that the `AZURE_TENANT_ID`, `AZURE_CLIENT_ID`, and `AZURE_CLIENT_SECRET` values are copied correctly and have no extra characters or whitespace. Verify the client secret has not expired.
- **Error: "Collection does not exist"**
  - **Cause**: The Site ID used in a script is incorrect.
  - **Solution**: Ensure you are using the correct `site_id` as provided by the verification script in section **5.0**.

8.0 Appendix

8.1 Post-Setup Actions

After successful verification, you can use the configured importer to interact with SharePoint.

1. Launch the Python interpreter within the project's environment: `bash ./venv/bin/python`
2. Use the `SharePointImporter` class to list libraries and download files:

```
from sharepoint_importer import SharePointImporter

importer = SharePointImporter()

# Example: List document libraries in a specific site
# libraries = importer.list_document_libraries(site_id="YOUR_SITE_ID")

# Example: Download files from a specific library
# importer.download_files_from_library(
#     site_id="YOUR_SITE_ID",
#     drive_id="YOUR_DRIVE_ID",
#     output_dir="input_documents/sharepoint/oberaconnect"
# )
```

8.2 Completion Checklist

- `Sites.Read.All` (Application) permission added.

- `Files.Read.All` (Application) permission added.
- Admin consent granted for new permissions.
- Tenant ID copied from application Overview page.
- Application (Client) ID copied from application Overview page.
- Client Secret created and its value securely copied.
- `AZURE_TENANT_ID`, `AZURE_CLIENT_ID`, and `AZURE_CLIENT_SECRET` added to `.env` file.
- Connection verified successfully with `sharepoint_importer.py`.