Publish to Microsoft Graph - How to Configure

Step 1. Launch Aspire and open the Content Source Management Page

Launch Aspire (if it's not already running). See:

- Browse to: **http://localhost:50505/**. For details on using the Aspire Content Source Management page, please refer to Aspire Architecture.

Step 2. Add a new Content Source

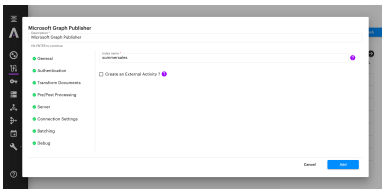
- For this step please follow the step from the Configuration Tutorial of the connector of your choice, please refer to Connector list

Step 3. Add a new Microsoft Graph publisher to the Workflow

To add a Publish to Microsoft Search drag a **Microsoft Graph publisher** rule from the *Workflow Library* a nd drop to the *Workflow Tree* where you want to add it. This will automatically open the Microsoft Graph publisher window for the configuration of the publisher

---

Step 4. Specify General Information

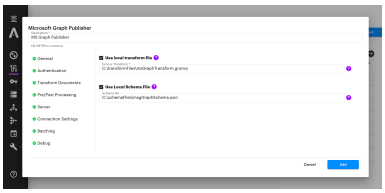


- **Name**: Unique name for the publisher.
- **Index Name**: The name of the connection/index that will be created in MS Graph.
- **Create an External Activity**: check if you are adding an external activity.

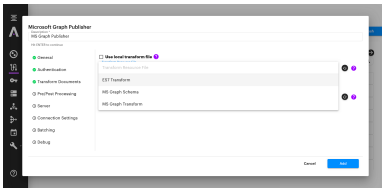
Step 5. Authentication Information

- **Tenant ID**: The tenant ID provided by Microsoft.
- **Client ID**: The client ID generated when the Application was registered in Prerequisites.
- **Client Secret**: The client secret that was generated in Prerequisites.

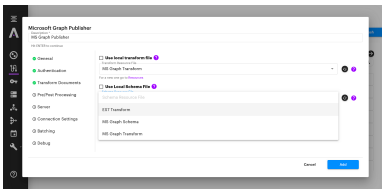
Step 6. Specify Transform Documents



- **Groovy Transform**: The path to a Groovy transformation file that will process the document to make it match the expected structure from either the fixed ExternalFile or the custom ExternalItem data types in MS Graph connection schemas. You can choose between specifying a Local Transform File or picking from a previously uploaded Resource Transform File:
    1. **Local Transform File**: Specify the path to your transform file. You can follow the instructions in JSON Transformation to create a custom transform file.
    2. **Resources Transform File**: pick the appropriate file that was previously uploaded by using Aspire's "Resources" feature.



- **Use custom schema**: Enables the usage of the limited custom schema in MS Graph. You can choose between specifying a Local JSON Schema File or picking from a previously uploaded Resource Schecma File:
    1. **Local Schema File**: If not specified, it will assume ExternalFile schema, otherwise it will be ExternalItem and will require the path to a JSON schema file.
    2. **Resources Schema File**: pick the appropriate file that was previously uploaded by using Aspire's "Resources" feature.



Step 7. Specify Pre/Post Processing Parameters

- **Clear Index Before Publishing:** if checked, it will clear the index previous to publishing.
- **Commit After Publishing**: if checked, the connector will commit changes after publishing.

Other configuration items are common to every publisher component.

ExternalFile vs ExternalItem

Microsoft Graph allows the usage of one of two schemas: fixed ExternalFile or custom ExternalItem. ExternalItem allows limited freedom to define properties to be expected from crawled items. Needless to say, the Groovy transformation file must yield an output that matches the expected schema.

ExternalFile schema

The fixed external file schema expects the following information:

- **acl**: The list of ACLs for the document
- **createdDateTime**: A standard UTC string that represents the creation date and time
- **modifiedDateTime**: A standard UTC string that represents the last modification date and time
- **createdBy**: The author's name
- **lastModifiedBy**: The name of the last person that modified the document
- **title**: The document's title
- **URL**: The document's URL
- **name**: The document's name
- **extension**: The document's file name extension
- **size**: The document size
- **content**: The document's content

This is a sample document in JSON format, as expected by the REST API:

```
{
    "acl": [
        {
            "type": "user",
            "value": "d411eb08-42e2-4316-aab5-2df8e9d9c21b",
            "accessType": "grant",
            "identitySource": "Azure Active Directory"
        }
    ],
    "createdDateTime": "2017-11-08T19:06:17Z",
    "modifiedDateTime": "2017-11-08T19:06:17Z",
    "createdBy": "empty",
    "lastModifiedBy": "empty",
    "title": "sample document",
    "url": "http://the.url.com",
    "name": "name.txt",
    "extension": "txt",
    "size": 10,
    "content": "the content/n"
}
```

ExternalItem schema

As mentioned before, the ExternalItem schema has limited customization capabilities. It expects the following information:

- **acl**: The list of ACLs for the document.
- **properties**: A name/value list of custom properties of predefined types.
- **content**: The document's content.

When configured to use custom schema, the publisher component expects a TXT file with the following structure:

```
{
    "properties": [
        {
            "name": "propertyName",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        }
    ]
}
```

Where:

- **name**: The property name in JSON friendly characters
- **type**: The data type of the property. Can be one of the following:
    - String
    - Int64
    - Double
    - DateTime (as UTC offset: YYYY-MM-DDTHH:mm:ss.nnZ)
    - Boolean
    - Collection(String)
    - Collection(Int64)
    - Collection(Double)
    - Collection(DateTime)
- **isSearchable**: (optional) If true, the property will be indexed as containing searchable content
- **isRetrievable**: (optional) If true, the property can be retrieved as part of search results
- **isQueryable**: (optional) If true, the property can be queried for specific values

This is a the default schema file provided with the component (schemaProperties.json):

```
{
    "properties": [
        {
            "name": "id",
            "type": "String"
        },
        {
            "name": "name",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        },
        {
            "name": "createdDateTime",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        },
        {
            "name": "modifiedDateTime",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        },
        {
            "name": "title",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true"
        },
        {
            "name": "url",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        },
        {
            "name": "description",
            "type": "String",
            "isSearchable": "true",
            "isRetrievable": "true",
            "isQueryable": "true"
        }
    ]
}
```

Groovy Transformation file

The Groovy transformation file makes it easy to output data that is customized to the client's needs and also that can be safely conveyed to Microsoft Graph through the REST API. The output of the transformation must match the expected schema structure.

This is the default Groovy transformation file that is provided with the component (aspireToMicrosoftGraphBulk.groovy):

```
package config.groovy

import java.security.MessageDigest

def connectorSpecificMap = [
    'isContainer': 'is_container'
]
```

```
def getContent(String content) {
  try {
    if (content.getBytes().length > 16777216L) {
      return content.substring(0, 10485760) + "...";
    } else {
      return content
    }
  } catch (Throwable t) {
    return "";
  }
}

def getMD5(String id) {
  MessageDigest digest = MessageDigest.getInstance("MD5")
  String md5name = new BigInteger(1, digest.digest(id.getBytes())).toString
(16)
  return md5name;
}

// Function that process the children of a connector specific field
def getChildren(name, parent) {

  builder."$name"() {
    parent.getChildren().each() { val ->

      def attr = val.getName();

      //if it has other children
      if (val.getChildren().size() > 0) {
        getChildren(attr, val);
      } else {
        builder."$attr"() {

          //All the attributes
          val.getAttributeNames().each() { attrName ->
            "@$attrName" val.getAttribute(attrName);
          }

          //Main content
          if (val?.getText() != null) {
            '_$' val?.getText();
          }
        }
      }
    }
  }
}

// 2021-04-27T11:04:00Z NMS GRAPH DATE TIME FORMAT
def formatBlogDateTime(String strDate) {
  String pattern = "MM/dd/yyyy hh:mm:ss a"
  String outputPattern = "yyyy-MM-dd'T'HH:mm:ss'Z'"

  return Date.parse(pattern, strDate).format(outputPattern)
}

def splitString(String str, String delimeter) {
  return str.split(delimeter)
}

def insertBusinessAddress(String eid) {
  return eid.concat("@accenture.com")
}


def getBody() {
  //Build body
  builder.$object() {

    // ACLs
    acl = null
```

```
    if (doc.acls == null || (doc.acls).equals("") || (doc.acls).isEmpty()
|| (doc.acls).equals("[]")) {
      //Default acls
      builder.acl() {
        $list {
          builder.$object() {
            type "Everyone"
            value "cc4e4bb7-5cce-4b65-80e1-f282b630ca4b"
            accessType "grant"
            identitySource "Azure Active Directory"
          }
        }
      }
    } else {
      builder.acl() {
        $list {
          doc.acls.getChildren().each() { val ->
            $object() {
              value getMD5(val.getAttribute("name")).replaceAll("\\s", "")
              accessType val.getAttribute("access")
              type val.getAttribute("entity")
            }
          }
        }
      }
    }

    // Properties builder
    properties = null
    builder.properties() {
      String newId = "";
      // Get ID
      if (doc.id != null) {
        newId = getMD5(doc.id.getText())
      } else if (doc.fetchUrl != null) {
        newId = getMD5(doc.fetchUrl.getText())
      } else if (doc.url != null) {
        newId = getMD5(doc.url.getText())
      } else if (doc.displayUrl != null) {
        newId = getMD5(doc.displayUrl.getText())
      } else {
        newId = "ID-NOT-PROVIDED"
      }

      'id' newId

      String collectionString = "Collection(String)"
      'blogauthor@odata.type' collectionString

      ArrayList authors = ["new.testemail@accenture.com"];
      'blogauthor' authors
    }

    // Content builder
    content = null
    builder.content() {
      'type' "text"
      if (doc.content != null) {
        'value' doc.content?.getText()
      }
    }
  }
}

//************************************************
//
// Main routine
//

// Action of the job
String action = doc.action.getText();
```

```
if ((action == "add") || (action == "update")) {
  /*****************
   * Add or Update *
   *****************/
  getBody()
} else {
  /**********
   * Delete *
   **********/

  builder.$object() {
    '@search.action' "delete"

    String delId = "";
    // Get ID
    if (doc.id != null) {
      delId = getMD5(doc.id.getText())
    } else if (doc.fetchUrl != null) {
      delId = getMD5(doc.fetchUrl.getText())
    } else if (doc.url != null) {
      delId = getMD5(doc.url.getText())
    } else if (doc.displayUrl != null) {
      delId = getMD5(doc.displayUrl.getText())
    } else {
      delId = "ID-NOT-PROVIDED"
    }

    'id' delId
  }
}
```

Once you've clicked on the *Add* button, it will take a moment for Aspire to download all the necessary components (the Jar files) from the Maven repository and load them into Aspire. Once that's done, the publisher will appear in the Workflow Tree.

For details on using the Workflow section, please refer to Workflow introduction.