Document Integration Guide

About Documents API

The Argus document system provides a hierarchical folder structure **per customer**, with named **folders** which can contain **subfolders** or **documents**.

The Document system uses role based access to control read- and write access to the documents per customer. In addition, each folder or document can be restricted and granularly access controlled using **explicit access control.**
Permission checking is transitive, meaning that to read a folder or document element, a user needs *folder access* to the elements parent folder (recursively, up to the root folder).

The Document system is versioned, so each document update will generate a new **revision** of a document, and the system retains the revision history. It allows a user to lock a document for a period of time, to avoid conflicting updates.

Please read the General Integration Guide to learn the general concepts and common data structures used throughout the Argus API.

Detailed API documentation

The Swagger API documentation is always up-to-date and lets you try out any query with your user session or an API-key.

Integration guide

Documents and Folders by ID

Every document and folder has a unique ID, unique across all document spaces. Operations on document/folder ID is unambiguous, and therefore recommended for most API usage.

```
#fetch the metadata of the folder with ID 99
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/99

#fetch the metadata of the document with ID 101
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/document/101
```

Folders by path

The Document API also supports fetching folders and documents **by path**, i.e. the path of each folder (+ document name for a document) is specified in the URI, relative to the document space root root. The path elements must use valid URL Encoding.
The space root for path endpoints will always be the space of the default customer for the current user , unless otherwise specified in the "customer" query parameter:

```
#fetch the metadata of the folder folder "Reports/Weekly Reports" in the
space of the default customer for the current user
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/path/meta/Reports/Weekly%20Reports

#fetch the metadata of the folder "Reports/Weekly Reports" in the space of
customer "mycustomer"
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/path/meta/Reports/Weekly%20Reports?customer=mycustomer

#fetch the metadata of the document "Reports/Weekly Reports/Report.pdf" in
the space of the default customer for the current user
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/path/meta/Reports/Weekly%20Reports/Report.pdf
```

See the Swagger API documentation for details.

Fetch root folder for customer

To fetch the root folder of a particular customer, use the customer root folder endpoint:

```
#fetch the document space root folder of customer 1
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/customer/1
```

Or by path

```
#fetch the document space root folder of customer "customer"
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/path/meta/?customer=customer
```

This returns a folder descriptor object:

```
{
  "data": {
    "id": 666,
    "parentElements": [],
    "name": "Root folder for mnemonic (mnemonic)",
    "customer": {...},
        ...
    "accessMode": "roleBased",
    "effectiveAccessMode": "roleBased",
    "currentUserAccessLevel": "write",
    "elementType": "folder",
    "flags": [
      "ROOT_FOLDER"
    ]
  },...
}
```

The folderelement structure returned is the same used for all document and folder endpoints.

The `currentUserAccessLevel` allows the user to quickly determine which permissions the user has for this particular element.

See the Swagger API documentation for details.

Listing the contents of a folder

To list the folders and documents contained in a folder, use the *folder content* endpoint:

```
#list the documents/subfolders contained in the folder 666
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/666/content
```

Or by path:

```
#list the documents/subfolders contained in the folder Parent Folder (in
the space of the default customer for the current user)
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/folder/path/content/Parent%20Folder
```

The response from this endpoint returns a list of folders and documents contained in this folder:

```
{
        "data": [
                {
                        "id": 1087,
                        "parentElements": [{"id": 666,"name": "Parent
folder"}],
                        "name": "Folder name",
                         ...
                        "accessMode": "explicit",
                        "effectiveAccessMode": "explicit",
                        "currentUserAccessLevel": "write",
                        "elementType": "folder",
                        "flags": []
                }],
                "size": 1
        ],...
}
```

The `parentElements` field of a document/folder element lists the string of parent folders from this elements immediate parent up to the document root.

See the Swagger API documentation for details.

Fetching a document

To the metadata of a document, use the *document* endpoint:

```
#fetch the metadata of the document 666
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/document/666
```

To fetch the actual contents of the document, use the *document content* endpoint

```
#fetch the contents of the document 666
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/document/666/content
```

Or by path:

```
#fetch the metadata of the document /Parent Folder/Folder/Document.pdf (in
the space of the default customer for the current user)
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/document/path/meta/Parent%20Folder/Folder/Document.pdf

#fetch the contents of the same document
curl -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents/v1
/document/path/content/Parent%20Folder/Folder/Document.pdf
```

See the Swagger API documentation for details.

Creating a document

To create a document, post a new document to the *folder documents* endpoint:

```
#create a new document in folder 666 with name "my document", mimetype
"text/plain" and a plaintext content
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666/documents -d '{
  "name": "my document",
  "text": "contents",
  "mimeType": "text/plain"
}'


#create a new document in folder 666 with name "my image", mimetype "image
/png" and a base64-encoded binary content
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666/documents -d '{
  "name": "my image",
  "data": "MmZkZWFlZTUwYWMyMmRmNzhlOWNjN2MxOWViYjU2NDAK",
  "mimeType": "image/png"
}'
```

To upload by path, use the streaming upload by path endpoint, below.

Upload document as stream

For larger documents, the document API now supports uploading by stream:

```
#create a new document "Document.pdf" in the folder 202 from the local
file /path/to/Document.pdf
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/pdf" https://api.mnemonic.no/documents/v1/folder/202/documents --data-
binary @/path/to/Document.pdf
```

Or by path

```
#create a new document "/Folder/Subfolder/Document.pdf" (in the space of
the default customer for the current user)
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/pdf" https://api.mnemonic.no/documents/v1/document/path/Folder/Subfolder
/Document.pdf --data-binary @/path/to/Document.pdf

#create a new document "/Folder/Subfolder/Document.pdf" (in the space of
the customer "mycustomer")
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/pdf" https://api.mnemonic.no/documents/v1/document/path/Folder/Subfolder
/Document.pdf?customer=mycustomer --data-binary @/path/to/Document.pdf
```

When uploading by path, the endpoint will reject the request if any of the parent folder elements are missing. Use option `createMissing=true` to create missing parent elements.

When posting a document, the endpoint will reject the request if an element with the same name exists in the parent folder. Use option `overwriteExisting=true` to overwrite existing document.
This will fail if the existing element is a subfolder, or if the current user does not have write permissions to the existing element.

See the Swagger API documentation for details.

Updating a document

To update an existing document, perform a PUT to the *document endpoint.*

```
#update document 1027 with new name "new documentname", mimetype "text
/csv" and a plaintext content
curl -XPUT -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/document/1027 -d '{
  "name": "new documentname",
  "text": "a,b,c",
  "mimeType": "text/csv"
}'
```

In both examples, this will create a new revision of document 1027.

To update an existing document with streaming upload, use the streaming upload endpoint above, with option `overwriteExisting=true`

See the Swagger API documentation for details.

Creating a new folder

To create a new folder, use the POST endpoint on the folder element:

```
#create a new subfolder in folder 666 with name "my folder"
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666 -d '{
  "name": "my folder"
}'
```

Or by path

```
#create a new subfolder in folder "MyRootFolder" with name "my folder"
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/path/MyRootFolder -d '{
  "name": "my folder"
}'
```

See the Swagger API documentation for details.

Deleting a document

```
#delete document 1027
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/document/1027
#delete document /Folder/Subfolder/Document.pdf
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/document/path/Folder/Subfolder/Document.pdf
```

See the Swagger API documentation for details.

Deleting a folder

To delete a folder, the default mode is DELETE_IF_EMPTY, meaning that the folder must not contain any non-deleted entries, else the request will fail

```
#delete folder 666
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/folder/666
#delete folder /Folder/Subfolder (in the space of the customer
"mycustomer")
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/folder/path/Folder/Subfolder?customer=mycustomer
```

To delete a folder recursively, use mode DELETE_CASCADE

```
#delete folder 666
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/folder/666?mode=DELETE_CASCADE
```

See the Swagger API documentation for details.

Managing document access

Managing access to documents can be done per document, or on a folder level.

Folder access management can be done by the document owner, or users with the *document manager* role.

A users effective access permissions to an element can be **folder access**, **read access** or **write access**.

- Write access is required to update a document, or add a document or subfolder to a folder.
- Read access is required to read the contents of a document (data) or folder (subfolders /documents)
- Folder access is required to traverse a folder or fetch folder/document metadata.

When listing the contents of a folder which the current user has **folder** level access to, it will only list subfolders or documents which the user also has access to.

	Folder access	Read access	Write access
Fetch folder metadata	X	X	X
Fetch document metadata	X	X	X
List folder contents		X	X
Download document contents		X	X
Add subfolder			X (on parent folder)
Add document to folder			X (on parent folder)
Update document			X (on document)

Role based access to documents provides different access levels for documents/folders depending on their access mode:

Role vs AccessMode	DOCUMENT-VIEWER	DOCUMENT-EDITOR	DOCUMENT-MANAGER	DOCUMENT-ADMIN
roleBased	Read access	Write access	Write access	Write access
writeRestricted	Read access	Read access	Write access	Write access
readRestricted	None	None	Write access	Write access

explicit	None	None	None	Write access

Explicit access control on documents/folders grants the affected user/group an explicit access level (read/write/folder access).

Granting explicit access for a subject to a folder/document will automatically grant **folder access** to the parent folder (recursively to the root folder), unless explicit access is granted to the same subject.

Creating element with restricted access

To create a folder with restricted access

```
#create a new subfolder with explicit access mode
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666 -d '{
  "name": "my folder",
  "accessMode": "explicit"
}'
```

To create a document with restricted access:

```
#create a new document with access mode "explicit"
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/document/666/documents -d '{
  "name": "my document",
  "text": "contents",
  "mimeType": "text/plain",
  "accessMode": "explicit"
}'
#same operation for streaming upload by path
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: text/plain"
https://api.mnemonic.no/documents/v1/document/path/Folder/Subfolder/my%
20document?accessMode=explicit --data-binary @/path/to/document.txt
```

Setting new access mode on an element

```
#set access mode explicit on document 1027
curl -XPUT -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/document/1027/access -d '{
  "accessMode": "explicit"
}'
#set access mode explicit on folder 666
curl -XPUT -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666/access -d '{
  "accessMode": "explicit"
}'
#set access mode explicit on /MyFolder by path
curl -XPUT -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/path/access/MyFolder -d
'{
  "accessMode": "explicit"
}'
```

Granting explicit access to an element

```
#grant write access for subject 130 to document 1027
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/document/1027/access -d '{
  "subjectID": 130,
  "level": "write"
}'
#grant folder access for subject 130 to folder 666
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/666/access -d '{
  "subjectID": 130,
  "level": "folder"
}'
#grant folder access for subject 130 to folder /MyFolder
curl -XPOST -H "Argus-API-Key: my/api/key" -H "Content-Type: application
/json" https://api.mnemonic.no/documents/v1/folder/path/access/MyFolder -d
'{
  "subjectID": 130,
  "level": "folder"
}'
```

Listing explicit access granted to an element

```
#list explicit access granted for document
curl -XGET -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents
/v1/document/1027/access
#list explicit access granted for folder 666
curl -XGET -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents
/v1/folder/666/access
#list explicit access granted for folder /MyFolder
curl -XGET -H "Argus-API-Key: my/api/key" https://api.mnemonic.no/documents
/v1/folder/path/access/MyFolder
```

Revoking explicit access from an element

Use the ID of the access element listed from the *access* endpoint:

```
#revoke explicit access granted to document 1027
curl -XDELETE -H "Argus-API-Key: my/api/key" https://api.mnemonic.no
/documents/v1/document/1027/access/623
```