e-Share REST API Guide

Information on using e-Share REST API to create and manage trusted shares

Table of Contents

Shares API

The Shares API provides a way to create, update and get information about Trusted Shares. Trusted Shares can be created either by the user/owner making the request or on behalf of someone else. An example of the second use case is e-Share's Secure Mail Gateway. Secure Mail Gateway (SMG) is an application that receives e-mails and creates Trusted Shares with the files and body of the e-mail. The application (SMG) makes requests to e-Share server, but such requests in turn create a Trusted Share on behalf of the e-mail sender.

A Trusted Share has three required properties:

1. Recipients – A list of email(s) that the Trusted Share should be send to
2. Shared item – The Item can either be already located in the provider or can be uploaded later
3. Options – Permissions that the recipients will have on the shared item

Recipients are defined with a list of valid emails and for each recipient a separate URL is created by e-Share. Each such URL is called "Collaboration".

The shared item can be already located in the provider, in which case a mount point and an item id are required to create the TS. If these two are not supplied with the request, then e-Share will create a new folder in the provider under a folder called "outbox" to put all files that the user may upload.

Options are the permissions given to the recipients by the user/owner. It's a JSON object with the following properties:

- "can_read" (Boolean): Recipient can view/read items
- "can_create" (Boolean): Recipient can create new folders and files
- "can_edit" (Boolean): Recipient can edit files on-line.
- "can_delete" (Boolean): Recipient can delete files and folders.
- "can_download" (Boolean): Recipient can download files
- "watermark" (Boolean): If the file must be watermarked with recipient's email and ip
- "login_required" (Boolean): The user must login in e-Share prior to accessing the TS
- "enforce_openid" (Boolean): If e-Share detects that the recipient has an OpenID enabled email, enforce to login with OpenID. E-Share checks for GSuite and Office365 accounts.
- "require_terms_of_use" (Boolean): Recipient must agree to the Terms Of Use before accessing the TS
- "show_terms_once" (Boolean): If e-Share should show the terms of use only once if the user has agreed to them.
- "expiration" (integer): The time in seconds this TS should remain accessible.
- "pin_protected" (Boolean): Recipient must enter a pin (security code) before accessing the TS
- "pin" (string): Pin value
- "send_pin_on_email" (Boolean): e-Share will send the pin with an email

- "secure_conversation_enabled" (Boolean): If the Secure Conversation should be enabled for this TS (defaults True)
- "only_message_owner" (Boolean): When Secure Conversations are enabled, recipient messages will be visible to thw TS owner (default False)
- "message" (string): A brief message to be displayed to the recipient.

Authentication

All Shares API requests are authenticated by using the Authorization header with the device token of the user making the request.

Authorization: NCC token=<your_token>

Available Endpoints

- Get My Shares
- Get Shared with Me Shares
- Create a Share
- Get info about a TS
- Add recipients to a TS and

Responses

Status codes are REST compliant, 2xx for successes, 3xx for redirects, 4xx for request errors, 5xx for server errors.

The main object returned in this set of endpoints is a ShareData Object, which holds the attributes of the share. It may include a list of recipients which is represented by a list of ShareRecipient Objects.

ShareData Object

{

- mount_id (string): Unique identifier of the current user's mount to the collaboration
- identity_id (string): Unique identifier of the identity created the TS
- name (string): The collaboration's name
- url (string): URL the current user can view the collaboration online with,
- category (integer): The Share category (3: TS, 4:SM, 5: Bot sharepoint share)
- type (string): The Share type
- is_offline (boolean): Whether the share can be made offline in a client device
- is_swml_share (boolean): Whether the share was created through ShareWithMe functionality
- swml_share_unsubscribed (boolean): Whether the ShareWithMe Share originator used the unsubscribe link to disable it
- is_single_file (boolean): Whether the share is a single file
- created (string): Creation date

- last_modified (string): Modification date
- metadata_revision (string): A unique identifier that changes when the share properties are changed
- id (string): Unique identifier of the share
- recipients (array[ShareRecipient]): Recipient dependent share url
- can_create (boolean): Whether the share has can_create option
- can_read (boolean): Whether the share has can_read option
- can_download (boolean): Whether the share has can_download option

}

ShareRecipient Object

{

- id (string): Unique identifier of the share recipient,
- email (string): Email of the share recipient
- url (string): URL the recipient can view the collaboration online with
- last_accessed (string): Last access date
- expires_at (string): Expiration date, null if the share does not expire
- is_active (boolean): Whether the share is active
- communication_status (string): Status of email send to recipient
- communication_log (string): Logs of email sending process
- is_deleted (Boolean): Whether the recipient is permanently removed

}

Endpoints

Get My Shares

Method: GET

URL: /api/3.0/shares/by-me/

Optional query parameters:

- name (string): Return Shares with matching name
- mount_id (string): Return matching mount point id
- offline (Boolean): Whether to return collaborations that are available offline.
- category (integer): Only return Shares of certain category (3: TS, 4: SM, 5: Bot)
- type (string): Only return Shares of certain type:
- TS: Trusted Share

- SM: Secure Mail
- BOT: Bot Share
- BOT-SP: Bot share through SharePoint
- SWML: ShareWithMeLink Share
- BLOB: TS using Blob Storage
- inactive (Boolean): Whether to return inactive collaborations, defaults to false
- shares_revision (string): The current 'shares_revision'. If unchanged, no shares are returned
- include_url (Boolean): Include a CWP URL in response
- include_recipients (boolean): Include ShareRecipient objects in response

The shares_revision parameter can be used to gat incremental listing of the user's shares. When given without any value a value for this will be returned in the response. If this value is included in a subsequent request, only the new shares will be returned and a new shares_revision identifier

Response:

{

- shares (array[ShareData]): Shares owned by me. Not returned if the passed shares_revision is the latest,
- shares_revision (string): An identifier that changes whenever the active list of my shares is modified. This value concerns all my shares, not just the ones currently filtered by this query

}

Get Shared With Me Shares

Method: GET

URL: /api/3.0/shares/with-me/

Optional query parameters:

- name (string): Return Shares with matching name
- mount_id (string): Return matching mount point id
- offline (Boolean): Whether to return collaborations that are available offline.
- category (integer): Only return Shares of certain category (3: TS, 4: SM, 5: Bot)
- type (string): Only return Shares of certain type:
- TS: Trusted Share
- SM: Secure Mail
- BOT: Bot Share
- BOT-SP: Bot share through SharePoint
- SWML: ShareWithMeLink Share
- inactive (Boolean): Whether to return inactive collaborations, defaults to false

- shares_revision (string): The current 'shares_revision'. If unchanged, no shares are returned
- include_url (Boolean): Include a CWP URL in response
- can_download (Boolean): Whether to return only collaborations that allow downloads
- watermarked (Boolean): Whether to return only collaborations that apply watermarking

The shares_revision parameter can be used to gat incremental listing of the user's shares. When given without any value a value for this will be returned in the response. If this value is included in a subsequent request, only the new shares will be returned and a new shares_revision identifier

Response:

{

- shares (array[ShareData]): Shares owned by me. Not returned if the passed shares_revision is the latest,
- shares_revision (string): An identifier that changes whenever the active list of my shares is modified. This value concerns all my shares, not just the ones currently filtered by this query

}

Create Trusted Share

Creating a TS is a complex process and may include many steps until a TS is created. A separate document describes all the use cases and how to this endpoint. Here we give a brief description of the attributes being used.

Method: POST

URL: /api/3.0/shares/

Body:

{

- action (string): One of: 'create' (default),'initialize', 'send' and 'finalize'
- recipients (array Email): Required. A list of recipient emails
- options (Options Object): Required. The options to be used
- identity_id (string): Required. The guid of the sender's identity
- sharing_policy_id (integer): The sharing policy id used to verify the given options, if not present the options will be verified against the default Organization policy
- mount_id(integer): If not set (as in the thin client), server will generate the mount point, creating any container folders as necessary
- item_id (string): If set, an existing item located in the provider defined from mount_id will be shared
- share_id (string): This is the TS guid. If not set by client, server will generate it. Required in case the action is 'send' or 'finalize'

- share_name (string): A name for this share. If not present the TS will be named after the folder or the file being shared
- share_message_id (integer): Id of first shared message. It is used during finalization of TS in order to insert items metadata in message
- category (string): TS or SM default is TS

}

The response varies upon the arguments given and the action being done. The most use case is “action”: “create” with mount_id and Item_id. In this case the following object is returned.

{

- share (ShareData Object): The share object created
- mount_point (string): String respresantation of the mount point
- item_id (string): The item_id that is shared

}

Example:

Request: POST /api/3.0/shares/

Body:

```
{
"recipients": ["<recipient_email>"],
"mount_id": <mount_point_id>,
"item_id": "<item_id>",
"options": {
            "can_read": true,
            "can_download": true,
            "expiration": 86400
                    },
"identity_id": "<identity_guid"
}
```

- mount_id: Can be found by getting the list of proivder mounts with /api/3.0/mounts/list?category=1
- item_id: Can be found by getting a list of items with /api/3.0/mounts/<mount_id>/items
- identity_id: Can be found by issuing a GET request to /api/3.0/identities

Response:

```
{
   "share": {
      "mount_id": "2587",
      "identity_id": "Cebw2e6kj6zY",
```

```
        "name": "local_folder1",
        "url": "……",
        "category": 3,
        "type": "Trusted Share",
        "is_offline": false,
        "is_swml_share": false,
        "swml_share_unsubscribed": false,
        "is_single_file": false,
        "created": "2020-01-09T13:41:22Z",
        "last_modified": "2020-01-09T13:41:22Z",
        "metadata_revision": "2e1f59e1-0558-4876-9547-b6e53d1c066f",
        "id": "a6RYpKx8bqZM",
        "recipients": [
          {
            "id": "4647",
            "email": "sv11jsb@yopmail.com",
            "url": "……",
            "last_accessed": null,
            "expires_at": "2020-01-10T13:41:22Z",
            "is_active": true,
            "communication_status": "in_flight",
            "communication_log": "",
            "is_deleted": false
          }
        ]
      },
      "mount_point": "NnZV82oS /local_folder1/",
      "item_id": "/local_folder1/"
    }
```


Get information about a Share

Method: GET

URL: /api/3.0/shares/<share_id>

Optional query parameters:

- include_recipients(Boolean): Return recipient info
- include_options (Boolean): Return options info

Response:


```
    {
```

- "share" (ShareData Object): A share data object, with optionally recipient and options info

}

Events API

e-Share provides an Events API which can be used to get events of Trusted Shares, such as file uploads, file deletions etc.

Description

Authentication of the request is performed through the Authorization header, having the device token of the user that owns the TS. The user must first enable publishing for specific shares and then poll the Events endpoint to get any new events.

Cursor

The endpoint provides a 'cursor' for each event it sends. The user must save the last cursor and send it in a subsequent request to get only the new events that have happened since the last request, otherwise all events, from the time the shared got published, will be returned.

Event types

The user can supply the event types that wants to get events for. The following types are supported:

- file_add – New file added to the TS
- file_updated – File updated
- file_edit – Edited online
- file_open – Opened/Viewed
- file_download - Downloaded
- file_delete – Deleted
- file_rename – Renamed

Limit

The user can limit the amount of events to be returned from the endpoint. Default is 100 events. Maximum is 500. If there are more events to be returned than the limit, then the "has_more" attribute will be true. The user can get the next bunch of events by supplying the latest cursor value.

Enable publishing of events for a specific share

Headers: Authorization: NCC token=<device_token>

Method: POST

URL: /api/3.1/shares/<share_id>/publish-events

Response:

Content-type: application/json
Status code: 200
Body: {}

Disable publishing of events for a specific share

Headers: Authorization: NCC token=<device_token>

Method: POST

URL: /api/3.1/shares/<share_id>/unpublish-events

Response:

Content-type: application/json
Status code: 200
Body: {}

Get events

Headers: Authorization: NCC token=<device_token>

Method: GET

URL: /api/3.1/events

Query Params:

share_id: comma separated list of share ids the user wants events from
event_type: comma separated list of event types
since: last cursor value
limit: number if events to return, default 100, max 500.

Response:

Content-type: application/json
Status code: 200
Body: {

- "events": [<list_of_event_objects>],
- "has_more": <boolean>

}

Event Object:

- id (string): Unique identifier of the event,
- type (choice) = The event type,
- created_at (string): Creation timestamp of the event, as recorded by,
- recorded_at (string): Timestamp when the event was received and recorded by the server,
- actor (EventActorObject),
- context (EventContextObject),
- cursor (string): A cursor identifying the event sequence at this event.

Event Actor Object:

- email (string): The email of the user that generated the event,
- user (EventActorUserObject): User information available if the event actor is a user,
- identity (EventActorIdentityObject): Identity information if the event actor has an identity,
- organization (EventActorOrganizationObject): Organization information if the event actor belongs to an organization

Event Actor User Object:

- id (string): Unique identifier of the user that generated the event,
- name (string): The user's name

Event Actor Identity Object:

- id (string): Unique identifier of the user's identity,
- name (string): The user's identity name

Event Actor Organization Object:

- id (string): Unique identifier of the actor's organization,
- name (string): The organization's name

Event Context Object:

- file (FileEventContextObject)

File Event Context Object

- id (string): Unique file identifier,
- name (string): The file name,
- etag (string): The file Etag,
- version (string): The file version, can be used to download that version of the file,
- path_at_provider (string): The full path of the file at the provider,
- mounts (array[FileMountObject]): The mounts and shares the file belongs to, may be more than one,
- created (string),
- last_modified (string),

- size (integer)

File Mount Object:

- mount_id (string): Unique identifier of the mount the file belongs to,
- share_id (string): Unique identifier of the share the file belongs to,
- path_at_mount (string): The path of the file at this mount

Example

Request

GET
/api/3.1/events?share_id=jpwUMHCuxnkB,k8D4nSrPrDJy&event_type=file_add,file_edit&since=eyJ2IjogMSwgInAiOiBbeyJwIjogOTAsICJvIjogMzF9XX0=

Response


```
{
   "events": [
      {
         "id": "qBe7yZQ5ZfLZ",
         "type": "file_add",
         "created_at": "2019-12-31T09:41:18Z",
         "recorded_at": "2019-12-31T09:41:18Z",
         "actor": {
            "email": "sv11jsb@yopmail.com"
         },
         "context": {
            "file": {
               "id": "015ZE4P2RV4LHXD5G66BA3KCRD63JCC2M6",
               "name": "500shares_production.png",
               "etag": "\"{71CFE235-DEF4-41F0-B50A-23F6D221699E},1\"",
               "version": "1.0",
               "path_at_provider": "/ppe_events/500shares_production.png",
               "mounts": [
                  {
                     "mount_id": "164034",
                     "share_id": "jpwUMHCuxnkB",
                     "path_at_mount": "/500shares_production.png"
                  }
               ],
               "created": "2019-12-31T09:41:17Z",
               "last_modified": "2019-12-31T09:41:17Z",
               "size": 23664
            }
```

```
        },
        "cursor": "eyJ2IjogMSwgInAiOiBbeyJwIjogOTAsICJvIjogMjN9XX0="
      },
      {
        "id": "a4iQHieTUVAN",
        "type": "file_add",
        "created_at": "2019-12-31T10:19:35Z",
        "recorded_at": "2019-12-31T10:19:35Z",
        "actor": {
          "email": "sv11jsb@yopmail.com"
        },
        "context": {
          "file": {
            "id": "015ZE4P2QPOORWLBJRYRGJUFMC42EZGCP6",
            "name": "accounts1.png",
            "etag": "\"{65A3730F-3185-4CC4-9A15-82E6899309FE},1\"",
            "version": "1.0",
            "path_at_provider": "/ppe_events2/accounts1.png",
            "mounts": [
              {
                "mount_id": "164035",
                "share_id": "k8D4nSrPrDJy",
                "path_at_mount": "/accounts1.png"
              }
            ],
            "created": "2019-12-31T10:19:34Z",
            "last_modified": "2019-12-31T10:19:34Z",
            "size": 197082
          }
        },
        "cursor": "eyJ2IjogMSwgInAiOiBbeyJwIjogOTAsICJvIjogMjR9XX0="
      },
    ],
    "has_more": false
}
```


Remarks

In this example the user asked for events of shares *jpwUMHCuxnkB* and *k8D4nSrPrDJy*, for event types *file_add* and *file_edit*, since the *eyJ2IjogMSwgInAiOiBbeyJwIjogOTAsICJvIjogMzF8FFg=* cursor.

The response includes one event per share, both are file_add events and information about who uploaded the file and about the new file.

It is assumed that a publish request has already been done for each share, example is below.

POST /api/3.1/shares/jpwUMHCuxnkB/publish-events
POST /api/3.1/shares/k8D4nSrPrDJy/publish-events

The only information about the 'uploader' that is returned is the email, because the user does not have an account with e-Share. If the has an account in e-Share application, the user's name, identity and probably organization objects would had been returned.

All timestamps are returned in UTC time.

Mount API

Description

e-Share provides a consistent way to work with your files regardless where these files are stored. The user needs to provide a Mount Point Id. Mount Point is an e-Share term which is used to describe a specific folder located in a specific provider. e-Share will contact the provider to get information or data thus making the whole process transparent to the user. The Mount Point ID can be found in the response of a Create Share request, in the response of the Events request or can be provided by e-Share.

Authentication

All Mount API requests are authenticated by using the Authorization header with the device token of the user making the request.

Authorization: NCC token=<your_token>

Content type

All requests expect and return json format data. An exception is the upload request which must be multipart/form-data because the binary content of the file will be in the request body.

Available actions

- List all mounts
- List files and folders of mount point root folder
- Create new folder in mount point's root folder
- List files and folders of a specific folder
- Create new folder in a folder
- Get metadata of a folder or file
- Delete files or folders
- Copy files
- Move files
- Download files

- Upload files

Responses

Status codes are REST compliant, 2xx for successes, 3xx for redirects, 4xx for request errors, 5xx for server errors.

When listing files or folders a **FileOrFolderMetadata** object is returned for each one.

FileOrFolderMetadata:

- cloud_provider (string),
- name (string),
- original_name (string),
- guid (string),
- id (string),
- is_shared (boolean),
- full_path (string),
- type (choice) = ['folder' or 'google_team_drive' or 'file']: The type of items,
- created (string): RFC 3339 timestamp of the item creation date, may be missing,
- last_modified (string): RFC 3339 timestamp of the item modification date, may be missing for folders,
- view_url (string): The URL through which the item is viewable on the web client,
- mimetype (string),
- revision (string): The file's Etag, can be used to determine if the file has been modified and for conditional requests. Might be different than the file's version depending on the provider.,
- version (string): The file's version, which can be used to download specific versions of a file. Might be different than the Etag depending on the provider,
- size (integer),
- is_zip (boolean),
- version_id (string): The current version of the folder,
- is_versioned (boolean): Whether this folder is versioned,
- version_date (string): When the current version was published,
- version_update_topic_id (string): Topic id for this folder's version updates notifications,
- version_is_outdated (boolean): Is current folder state from cloud provider different from latest published state,
- is_folder (boolean): Whether the item is a folder

There also two stripped-down versions of this object representing either a file or folder.

FolderMetadata

- cloud_provider (string),
- name (string),

- original_name (string),
- guid (string),
- id (string),
- is_shared (boolean),
- full_path (string),
- type (choice) = ['folder' or 'google_team_drive' or 'file']: The type of items,
- created (string): RFC 3339 timestamp of the item creation date, may be missing,
- last_modified (string): RFC 3339 timestamp of the item modification date, may be missing for folders,
- view_url (string): The URL through which the item is viewable on the web client,
- version_id (string): The current version of the folder,
- is_versioned (boolean): Whether this folder is versioned,
- version_date (string): When the current version was published,
- version_update_topic_id (string): Topic id for this folder's version updates notifications,
- version_is_outdated (boolean): Is current folder state from cloud provider different from latest published state

FileMetadata

- cloud_provider (string),
- name (string),
- original_name (string),
- guid (string),
- id (string),
- is_shared (boolean),
- full_path (string),
- type (choice) = ['folder' or 'google_team_drive' or 'file']: The type of items,
- created (string): RFC 3339 timestamp of the item creation date, may be missing,
- last_modified (string): RFC 3339 timestamp of the item modification date, may be missing for folders,
- view_url (string): The URL through which the item is viewable on the web client,
- mimetype (string),
- revision (string): The file's Etag, can be used to determine if the file has been modified and for conditional requests. Might be different than the file's version depending on the provider
- version (string): The file's version, which can be used to download specific versions of a file. Might be different than the Etag depending on the provider,
- size (integer),
- is_zip (boolean)

The most important fields are:

- id – Will be used in a subsequent request as <item_id>

- name – The name of file or folder
- is_folder – Boolean
- mimetype – The mimetype of the file
- size – Size of file

Endpoints

List all mounts

This endpoint returns a list of all Secure Views owned by the user. A Secure View is the object containing information about a Mount Point and a Cloud Provider.

Method: GET

URL: /api/3.0/mounts/list

Optional query parameters:

- Category: Return only that category mounts. Valid categories are:
  - 1 – Provider mounts
  - 2 – Not used, deprecated
  - 3 – Trusted Share
  - 4 – Secure Mail
  - 5 – Bot share
- provider: Return only providers with this name

Response:

```
{
  "pagination" (PagintionObject): Information about the paged results
  "results" (arraySecureViewObject): List of mounts
}
```

Pagination Object:

```
{
  "results" (integer): Total number of results
  "pages" (integer): Number of pages needed to display all results
  "links" (object): {
      "next" (string): Next page URL,
      "previous" (string): Previous page URL
   }
}
```

Secure View Object:

{
id (string): Secure View ID
is_corporate (Boolean): If this a corporate entity or personal
mount_points (array[MountPointsObject]): list of Mount Points
last_modified (string): Timestamp
created (string): Timestamp
guid (string): Secure View GUID
slug (string): Unique name of Secure View
name (string): Name of Secure View
category (choice): ['1' or '2' or '3' or '4' or '5'],
active (Boolean): If the Secure View is active
hidden (Boolean): If the Secure View is hidde
identity_id (string): Owner identity of this secure view,
type_id (string): Type of identity
}

Mount Point Object:

{
id (string): Mount Point ID
created (string) Timestamp
last_modified (string): Timestamp
is_folder (Boolean): If this represents a folder
encrypted (Boolean): If it's encrypted
cloud_provider_auth (CloudProviderObject)
}

Cloud Provider Object:

{
id (string): Provider ID
provider_slug (string): Unique name of provider
}

List files and folders of mount point root folder

Method: GET

URL: /api/3.0/mounts/<mount_id>/items

Optional query parameters:

- name: Return only metadata for a specific item
- is_folder: If True, return only folder items. If False, return only file items. If not present, then both folder and file items are returned

Response:

```
{
  "items": [
    FileOrFolderMetadata object
  ]
}
```

List files and folders of a specific folder

Method: GET

URL: /api/3.0/mounts/<mount_id>/items/<item_id>/items

Response:

```
{
  "items": [
    FileOrFolderMetadata object
  ]
}
```

Optional query parameters:

- name: Return only metadata for specific item
- is_folder: If True, return only folder items. If False, return only file items. If not present, then both folder and file items are returned

Create new folder in a folder

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/<item_id>/items

Body:

{

- folder_name(string): New folder's name
- folder_type(string): One of "public", "private", "shared"

}

Response:

{

- item_id(string): The Id of the new folder

- full_path(string): The full path if it's available (maybe is the same as item_id)
- guid(string): The guid of the new folder (maybe is the same as item_id)

}

Error response codes:

- 409 – Conflict. There is already a file or folder named the same
- 422 – Unprocessable. The item_id given in the url is not a folder

Get metadata of specific folder or file

Method: GET

URL: /api/3.0/mounts/<mount_id>/items/<item_id>

Response:

{

- metadata (FileOrFolderMetadata): Metadata for the specified item,
- subfolders (array[FolderMetadata]): Metadata for the specified item's subfolders,
- files (array[FileMetadata]): Metadata for the specified item's files

}

Optional query parameters:

- include_subfolders: whether to include metadata of subfolders

Rename files or folders

Method: PATCH

URL: /api/3.0/mounts/<mount_id>/items/<item_id>

Body:

{

- rename_to(string): New name of the folder or file

}

Response:

{

- metadata (FileOrFolderMetadata): Metadata for the specified item

}

Delete files or folders

Method: DELETE

URL: /api/3.0/mounts/<mount_id>/items/<item_id>

Response:

{

- item(string): The item id that was deleted

}

Warning: This method performs a direct request to the cloud storage provider to delete the item. There is no confirmation step. If the item is a folder, all its contents will be deleted.

Copy files

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/copy

Body:

{

- source_items(array): List of items to copy
- destination_id(string): Destination folder id

}

Response:

{

- task_id (string): ID of started Celery task,
- task_url (string): Url for checking status of started Celery task

}

Description:

The source_items field accepts an array of item objects. The item object must contain at least the fields item_id and name.

The response has a task_id which can be queried with the Tasks API to get the progress of the copy process.

Move files

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/move

Body:

{

- source_items(array): List of items to move
- destination_id(string): Destination folder id

}

Response:

{

- task_id (string): ID of started Celery task,
- task_url (string): Url for checking status of started Celery task

}

Description:

The source_items field accepts an array of item objects. The item object must contain at least the fields item_id and name.

The response has a task_id which can be queried with the Tasks API to get the progress of the move process

Downloading and uploading files

Download and upload are two step processes when working with e-Share.

For download, the requested file needs to be downloaded to e-Share server and maybe decrypt it and then the user can download it locally. For upload, the file first needs to be uploaded to e-Share server, maybe encrypt it and then upload it to the provider space. It is recommended to break up large files (>30MB) into chunks and upload each chunk separately. If a chunk is not uploaded correctly it can be retried.

Initialize download

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/download

Body:

{

- items(array[Item]): List of items to download
- is_download(boolean): If this is a download request and no conversion will take place.

}

Response:

{

- tasks (array[Task]): List of task info objects

}

Task object:

{

- item_id (string): Id of item.,
- ready (boolean): If the latest item revision is present in our cache.,
- task_id (string): Id of started Celery task.,
- task_url (string): The URL to check for task status,
- revision (string): Revision of the item to be downloaded,
- counts (Counts): Counts of items enumerated for the download.,
- download_url (string): url to download the item once it is ready.,
- preview_url (string): url to preview the item once it is ready.,
- office_url (string): url to start edit the item in Office Online once it is ready.,
- office_preload_url (string): url to preload Office Online resources.,
- error_type (choice) = ['password-protected' or 'sensitive-document' or 'download-prohibited-by-policy' or 'permissions-denied' or 'key-access-forbidden' or 'download-size-limit-exceeded' or 'directory-is-empty']: Error code.,
- error_message (string): Friendly error message.,
- error_details (string): Extra information related to the error.

}

Count object:

{

- folders (integer): Number of folders to download.,

- subfolders (integer): Number of subfolders to download.,
- files (integer): Number of files to download.,
- total_size (integer): Total downloaded size in bytes.

}

Description

The array of Item to send in the request must include at least the field item_id. The file is downloaded and stored in e-Share's cache. The file can be retrieved with the Downlaod content request.

The response has a task_id which can be queried with the Tasks API to get the progress of the move process.

Download content

Method: GET

URL: /api/3.0/mounts/<mount_id>/items/<item_id>/content

Response: Binary content of the file requested

Description: The file must be already located in e-Share's cache, or else it will fail.

Initialize upload

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/upload

Body:

{

- mount_id(string): Required – Id of mount point
- parent_id(string): The id of the folder to upload into. If not specified, the upload will take place into the root folder of the mount.
- file_name(string): Required – File name
- size(integer): Required – Size of file in bytes
- overwrite(bool): Overwrite existing file. Default is false

}

Response:

{

- upload_id (string): New upload id.
- parent_id (string): The id of the item the file will be uploaded to.

}

Upload chunk

Method: POST

URL: /api/3.0/mounts/<mount_id>/items/upload/<upload_id>/

Type: multipart/form-data

Body form data:

- mount_id: Required – Id of mount point
- file_name: Required - File name to create
- chunk: Required Binary – Contents of chunk
- checksum: Required – MD5 checksum of chunk's contents
- chunk_order: Required Integer - The sequence order of this chunk, zero indexed
- is_last_chunk: Boolean – True if this is the last chunk

Response:

If the uploaded chunk is not the last one:

{

- size(integer): The size of the received chunk

}

If the chunk is the last one:

{

- task_id(string): The task_id to query the Tasks API for results.

}

Get information about upload status

During or after the upload of chunks you can request information about the upload status. It's simpler to use this url to get information instead of the Tasks API.

Method: GET

URL: /api/3.0/mounts/<mount_id>/items/upload/<upload_id>/

Response:

{

- upload (UploadObject): The upload session

}

Upload Object:

- id (string),
- type (choice) = ['copy' or 'encrypt' or 'folder_transfer' or 'move' or 'decrypt' or 'upload'],
- status (choice) = ['failed' or 'finished' or 'receiving_chunks' or 'in_progress' or 'initialized'],
- reason (string): Failure reason,
- mount_id (string),
- parent_id (string): The id of the folder to upload into,
- filename (string): File name to create,
- file_created (string): The creation date of the file,
- file_last_modified (string): The last modification date of the file,
- overwrite (boolean): Whether to overwrite an existing file,
- encrypt (string),
- created (string),
- expired (boolean),
- size_received (integer): Size in bytes of the file chunks received so far,
- item (string): JSON with the item metadata,
- share_message (string): Shared message

The upload is successful when all chunks are uploaded to e-Share and the Upload Sessions status field is finished.

Tasks API

Description

Many of our process can be completed with one-pass. For example, if a user requests to view a file that is being watermark protected, the file must first be downloaded from the provider, get decrypted, apply the watermark and then sent to browser for viewing. Depending on the file size this whole process can take several seconds. In order to complete it, a task is fired for every step. When a task has completed its job, it fires the next task that must be executed and so on.

Getting the results usually results in a new task that must be queried to get results from. When there is no next task to query the whole process has been finished.

Authentication

All Tasks API requests are authenticated by using the Authorization header with the device token of the user making the request.

Authorization: NCC token=<your_token>

Task results

Method: GET

URL: /api/3.1/tasks/<task_id>

Response:

{

- id (string): The task unique identifier,
- status (choice) = ['RECEIVED' or 'RETRY' or 'REVOKED' or 'SUCCESS' or 'STARTED' or 'FAILURE' or 'PENDING']: The task status,
- result (object): The task result if it is available

}

The result object may vary if the first has started one or many parallel tasks.

1. If there are many parallel tasks then an array of Task Info objects are returned.

   {

   - tasks_info(arrayTaskInfo): List of tasks that have been started

   }

   Task Info object:

   {

   - item_id(string): Item Id
   - task_id(string): Task id to query for results
   - is_folder(bool): If the Item is a folder
   - name(string): Name of the item

   }

2. If a single task has been started then the result will be:

{

- next_task(string): The next task id to query
- progress(integer): An integer indicating the progress of the whole process. 100 means there is nothing else pending to do.

}

3. If there is no next_task, then the result is:

{

- next_task: null
- result(objectFileOrFolderMetadata): Object with information about the performed action

}

Azure Blob Usage API

Description

Depending on the storage provider used by an organization, use of traditional cloud storage services like OneDrive, SharePoint, Google Drive, etc. may not be suitable for mass usage. Especially organizations using OneDrive/SharePoint as a storage provider may have experienced throttling from Microsoft's Graph API with high number of file transactions.

To address the issue, e-Share also supports API endpoint by using Azure's Storage as a provider for creation of Trusted Shares. Additionally, usage of API endpoint for Azure Blob is simpler and more suitable for sending large number of single files in each Trusted Share. Instead of issuing two or three requests to create, upload and send the Trusted Share, only one is needed with this API endpoint. The Azure Blob storage API endpoint requires a one-time setup and then can be used as detailed in the Usage section below.

Usage

The Azure Blob API endpoint accepts content type multipart/form-data and not application/json. This is because the file to be send will be also included in the request.

URL: /api/3.0/shares/blob/

Method: POST

Authorization:

When using 3.0: Authorization: NCC token=<your_token>
When using 3.1: Authorization: NCCAPP token=<your_token>
NCC-IDENTITY-EMAIL: <email_of_the_sender>

Fields:

recipients: email of recipient – Required
recipients_cc: email of cc recipient
recipients_bcc: email of bcc recipient

Due to the type of the request, you can include more than 'recipients' fields to have the same file sent to multiple recipients.

share_name: Your own name for the share
share_id: Your own id for the share
message: Short message
identity_id: required when version 3.0 is used.
pin: required if the policy has 'pin protected' option set.
file: the content of the file to be send. max. size 10MB - Required

Response:

The response will be a Share object containing information about the created share, and about the recipients, e.g.


```
{
  "share": {
    "mount_id": "2582",
    "identity_id": "hbU9sVPafdqG",
    "name": "payslip201912-4",
    "url": null,
    "category": 3,
    "type": "Trusted Share (through Blob Storage)",
    "is_offline": false,
    "is_swml_share": false,
    "swml_share_unsubscribed": false,
    "is_single_file": false,
    "created": "2019-12-03T09:10:27Z",
    "last_modified": "2019-12-03T09:10:27Z",
    "metadata_revision": "30d906ec-280e-4023-843b-fb6d6d3cdcb2",
    "id": "ebad3b8e-e31d-483b-bf58-df784cd8d304",
    "recipients": [
     {
       "id": "4630",
       "email": "<recipient_email>",
       "url": "<unique_recipient_url>",
```

```
        "last_accessed": null,
        "expires_at": "2020-02-03T09:10:27Z",
        "is_active": true,
        "communication_status": "not_sent",
        "communication_log": "",
        "is_deleted": false
      }
    ]
  }
}
```


The authorization, response and the field names are the same as the general API endpoints described above. The main difference is the type of the request and that only one request is required to create and send a Trusted Share.