Programmieren für ELO

ELO REST

Inhaltsverzeichnis

ELO Indexserver REST API

Introduction

> **Beachten Sie**
>
> Diese Dokumentation ist nur auf Englisch verfügbar.

The Indexserver REST API allows to call the Indexserver API in other environments than Java, JavaScript or .NET.

This documentation describes how to archive that and demonstrates the usage in some examples.

Contents

- Design principles
- Authentication for browser based applications
- Authentication for services
- Enable CORS
- Examples

> **Information**
>
> ELO Server provides information about Entrypoint of the Indexserver REST API and the OpenAPI Definition.
>
> - The Indexserver REST API is available at: `https://<ixhost>:<port/path>/rest`.
> - An OpenAPI document is available at: `https://<ixhost>:<port/path>/rest/openapi.json`

Design Principles

Paths

The URL path of a REST operation contains the interface name and the function name separated by slash.

Examples:

```
http://server:port/ix-repository/rest/IXServicePortIF/checkoutSord
http://server:port/ix-repository/rest/FeedService/findFirstActions
```

HTTP Methods and Content Types

To invoke an operation, send a POST request to the destination path and pass the parameters encapsulated in a JSON object in the request body.

While the OpenAPI document declares the operations only with application/json, requests can also be sent as multipart/form-data. This allows to send objects and streams in a single request.

The character encoding of the JSON data must be UTF-8.

POST with application/json

A POST request with application/json sends the interface and function name as path parameters in the URI. The function parameters are sent in a single JSON object in the request body. The element names in this parameter object must match the parameter names declared in the reference documentation.

Example Function createSord is declared as

```
EditInfo createSord(ClientInfo ci, java.lang.String parentId, java.lang.String maskId,
                    EditInfoZ editInfoZ) throws byps.RemoteException
```

The invocation path is as follows:

```
/IXServicePortIF/createSord
```

The parameters have to be wrapped by a JSON object. To simplify the usage, the ci parameter can be omitted. Furthermore, the element selector editInfoZ can be specified as comma separated field names in a list. Example:

```
{
   "parentId":"1",
   "maskId" : "0",
   "editInfoZ":"sord.lean,maskName,mask"
}
```

The entire HTTP request is as follows:

```
POST /ix-elo200/rest/IXServicePortIF/createSord HTTP/1.1
Host: localhost:8084
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:95.0)
            Gecko/20100101 Firefox/95.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
elo-approved: D5510B368C3081DA8D9529C200E93986
Content-Type: application/json
Content-Length: 105
Origin: http://localhost:8084
Connection: keep-alive
Referer: http://localhost:8084/ix-elo200/js-api-examples/rest/insert-folder.html
Cookie: JSESSIONID=7428BF67CEB824DCF3C72B6EDAA05C40.ELO-PCIMIGELO21-1;
        ELO-Approved=D5510B368C3081DA8D9529C200E93986
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{"parentId":"1","maskId":"(E10E1000-E100-E100-E100-E10E10E10E31)",
    "editInfoZ":"sord.lean,maskNames,mask"}
```

POST with application/json to upload streams (files)

To checkin a document or write BLOB data into map values, a POST request encoded as multipart/form-data is recommended, see POST with multipart/form-data

If this is not possible, the file or BLOB data can be uploaded into a temporary storage on the server before a subsequent checkin request attaches the temporary file with the repository object. The destination path for the POST request is:

```
/BUtility/upload
```

The server returns a JSON object with a random stream ID as follows:

```
{
  "result" : {
    "streamId" : "6596943602124994030"
  }
}
```

Use this stream ID in the following checkin request (e.g. checkinDocEnd):

```
/IXServicePortIF/checkinDocEnd
...
```

```
   "document":{
      "docs":[
         {
            "version":"1.0",
            "comment":"Version uploaded via REST call.",
            "contentType":"text/plain",
            "fileData":{
               "stream":{
                  "streamId":"6596943602124994030"
               }
            }
         }
      ]
   },
...
```

POST with application/json inclusive BLOB data

Files or large byte arrays can be sent as BASE64 encoded data within a checkin call. The BASE64 string has to be set in the "data" element of the FileData object.

```
/IXServicePortIF/checkinMap
...
      {
         "key":"bufferValue",
         "blobValue":{
            "contentType":"text/plain",
            "data":"U23DuHJyZWJyw7hkIGZvciAw4oKs"
         }
      },
...
```

> **Information**
>
> ELO Server delivers an internal documentation with examples via the following url scheme: `https://<ixhost>:<port/path>/js-api-examples/<function>`.
>
> Example: `https://srvpdoksrvint02vm:9093/ix-EXTEN/js-api-examples/rest/index.html#read_file_and_blob`

While a BLOB can be written as BASE64 encoded data along with the parameter object, it cannot be returned in a result object this way. For reading a BLOB, a separate GET request is required, see section *Read File and BLOB Data*.

POST with multipart/form-data

The content type multipart/form-data allows to send the parameter object and the streams (files to upload) in a single request. The parameter object has to be sent as form field named

"data". The streams can be sent with any other field name which need not to be unique. In case of multiple streams use the same field name, the order of the streams is important when referencing the streams in the parameter object.

The following example shows for a checkinDocEnd request that uploads a document version and attachment how to reference the streams in the parameter object. Both streams are sent with field name "files" which requires to supply an index in their reference. The first stream is assigned to the document version at "streamId" : "files[0]". The attachment is bound to the second stream at "streamId" : "files[1]".

```
{
   "ci":{...},
   "sord":{...},
   "sordZ":{...},
   "document":{
      "docs":[
         {
            "version":"1.0",
            "comment":"Version uploaded via REST call.",
            "contentType":"text/plain",
            "fileData":{
               "stream":{
                  "streamId":"files[0]"
               }
            }
         }
      ],
      "atts":[
         {
            "version":"1.0",
            "comment":"Attachment uploaded via REST call.",
            "contentType":"text/plain",
            "fileData":{
               "stream":{
                  "streamId":"files[1]"
               }
            }
         }
      ]
   },
   "unlockZ":{ ... }
}
```

The HTTP request with multipart/form-data is as follows:

```
POST /ix-elo210/rest/IXServicePortIF/checkinDocEnd HTTP/1.1
Host: localhost:8084
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0)
            Gecko/20100101 Firefox/87.0
```

```
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
elo-approved: E222F2145581E9C07AC5CC6804166BAA
Content-Type: multipart/form-data;
              boundary=---------------------------254483314546095894235331156 98
Content-Length: 6308
Origin: http://localhost:8080
Connection: keep-alive
Referer: http://localhost:8080/ix-elo210/js-api-examples/rest/insert-doc-att.html
Cookie: JSESSIONID=E8C5E7051D88837961C7D55E4544B3F3.ELO-PCIMIGELO21-1;
        ELO-Approved=E222F2145581E9C07AC5CC6804166BAA
Pragma: no-cache
Cache-Control: no-cache

----------------------------254483314546095894235331156 98
Content-Disposition: form-data; name="data"

{"ci":{"country":"DE","language":"de","ticket":"de.elo.ix.client.ticket_from_cookie",
        "timeZone":"UTC","options":0},
 "sord":{"acl":"2-7A+PYJA","att":0,"childCount":0,"doc":0,
         "guid":"(65368C89-8D3E-503E-8522-34622DD9E8DC)","histCount":0,"id":-1,
         "info":0,"key":0,"kind":0,"lockId":-1,"mask":0,"ownerId":0,"parentId":1,
         "path":1,"type":254,"vtRep":0,"IDateIso":"","XDateIso":"","access":127,
         "aclItems":[{"access":63,"id":9999,"name":"Everyone","type":0}],
         "delDateIso":"","deleted":false,
         "details":{"archivingMode":2001,
                    "encryptionSet":0,
                    "fulltext":false,
                    "sortOrder":1999,
                    "arcReplEnabled":false,
                    "fulltextDone":false,
                    "replRoot":false,
                    "linked":false,
                    "incomplete":false,
                    "limitedReleaseDocument":false,
                    "linkedPermanent":false,
                    "documentContainer":false,
                    "translateSordName":false,
                    "inheritAclDisabled":false,
                    "workspace":false},
         "lockName":"",
         "objKeys":[],
         "ownerName":"Administrator",
         "maskName":"Basic entry",
         "lockIdSord":-1,
         "lockIdDoc":-1,
         "lockNameSord":"",
         "lockNameDoc":"",
```

```
         "deleteUser":-1,
         "regionId":-1,
         "name":"Smørrebrød"},

 "sordZ":{"bset":"2196631817761587199"},
 "document":{"docs":[{"version":"1.0",
                      "comment":"Version uploaded via REST call.",
                      "contentType":"text/plain",
                      "fileData":{"stream":{"streamId":"files[0]"}}}],
            "atts":[{"version":"1.0",
                     "comment":"Attachment uploaded via REST call.",
                    "contentType":"text/plain",
                    "fileData":{"stream":{"streamId":"files[1]"}}}]},
 "unlockZ":{"bset":"0"}}
----------------------------254483314546095894235331156 98
Content-Disposition: form-data; name="files"; filename="APACHE_LICENSE.txt"
Content-Type: text/plain

 ... first file content ...
----------------------------254483314546095894235331156 98
Content-Disposition: form-data; name="files"; filename="readme.txt"
Content-Type: text/plain

 ... second file content ...
----------------------------254483314546095894235331156 98--
```

Read file and BLOB data

File and BLOB data can be read by sending a GET request to the relative URL received in a FileData object. This URLs are valid for 1 minute by default and can be downloaded only once. For BLOBs provided by checkoutMap, checkoutNotes, processOcr etc., this restrictions are negligible. For document versions or attachments provided in DocVersion objects, there is also an URL element that gives access to the content for 10 minutes (defined by Indexerver option documentUrlLifetimeSeconds) as long as the session is valid.

Example:

```
/IXServicePortIF/checkoutSord Response

...
 "docVersion": {
    "accessDateIso": "20210418101911",
    "comment": "Version uploaded via REST call.",
    "createDateIso": "20210418101900",
    "deleted": false,
    "ext": "txt",
    "contentType": "text/plain",
    "fileData": {
      "stream": {
```

```
        "url": "getstream?serverid=0&messageid=0&streamid=3625747495282349347"
      }
    },
    ...
    "url": "http://PCIMIGELO21:8084/ix-elo210/ix?cmd=readdoc1&id=233&objid=429...",
    ...
},
...
```

The stream URL is relative to the endpoint of the REST API. A GET request to download the file can look like:

```
GET /ix-elo210/rest/getstream?serverid=0&messageid=0&streamid=3625747495282349347 HTTP/1.1
Host: localhost:8084
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101 Firefox/87.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
elo-approved: 672DBD1CF4BE7646015B9C5C64EB632D
Connection: keep-alive
Referer: http://localhost:8080/ix-elo210/js-api-examples/rest/insert-document.html
Cookie: JSESSIONID=1186F3AEA08E2F309018F8A73190FC7D.ELO-PCIMIGELO21-1;
        ELO-Approved=672DBD1CF4BE7646015B9C5C64EB632D
```

Return values and Errors

An operation returns a JSON object either with member "result" or "exception". On success, the "result" member is set and contains the return value of the operation. If an error occurs, the exception element is set to a String that contains an error code in part "[ELOIX:errorcode]".

> **Information**
>
> ELO Server delivers an internal documentation with examples via the following url scheme: `https://<ixhost>:<port/path>/js-api-examples/<function>`.
>
> Example: `https://srvpdoksrvint02vm:9093/ix-EXTEN/js-api-examples/rest/index.html#contants`

Error codes are defined in IXExceptionC. They are provided in the "constants" object in element IXEXCEPTION.

If the operation cannot be parsed by the server, the response body is undefined and a HTTP status code is set to an error value (4xx or 5xx).

An exception to this rule is a GET request to download a file or BLOB. If this download request fails, the error is always defined by a HTTP status code.

Success response example:

```
{
   "result":{
      "clientInfo":{
         "country":"DE",
         "language":"de",
         "ticket":"de.elo.ix.client.ticket_from_cookie",
         "timeZone":"UTC",
         "options":0
      },
      "ticketLifetime":486,
      "user":{
         "desc":"",
         "flags":2013265919,
         "groupList":[
            9998,
            9999
         ],
         ...
}
```

Error response example

```
{
   "exception":"[TICKET:de.elo...][ELOIX:1002]Keine Lizenz für den Indexserver vorhanden.
               [NO-DETAILS]"
}
```

Constants

Many operations use constants in their parameters. To get the definition of this constants, call /IXServicePort/getConstants. This function can be called without authentication.

Since the response is quite large (60KB JSON text) and its values do not change for an installation , it should be cached by the client.

```
{
   "result":{
      "STREAM_VERSION":"21.00.000.015",
      "ACCESS":{
         "FLAG_ADMIN":1,
         "FLAG_EDITCONFIG":2,
         "FLAG_EDITSTRUCTURE":4,
         "FLAG_EDITDOCS":8,
         "FLAG_CHANGEPW":16,
...
      },
      ...
      "DOC_MASK":{
```

```
         "GUID_BASIC":"(E10E1000-E100-E100-E100-E10E10E10E30)",
         "GUID_EMAIL":"(E10E1000-E100-E100-E100-E10E10E10E32)",
         "GUID_STRUCTURE_ELEMENT":"(E10E1000-E100-E100-E100-E10E10E10E31)",
...
         "mbName":"2",
         "lnName":40,
      ...
      },
      ...
      "IXEXCEPTION":{
        "SERVER_ERROR":1000,
        "UNSUPPORTED_PROTOCOL_VERSION":1001,
        "INVALID_LICENSE":1002,
        "INVALID_PARAM":2000,
        "INVALID_SESSION":2001,
        "TEMP_PROBLEM_OLD":2002,
        "TEMP_PROBLEM":2003,
        "INVALID_CRYPT_KEY":2004,
        "SYNTAX_ERROR":2007,
        "PASSWORD_DENIED":3007,
        "ALREADY_EXISTS":5005,
        "LOCKED":5022,
        "NOT_FOUND":5023,
      ...
   }
 }
}
```

Authentication

Authentication for Browser-Based Applications

The session cookie JSESSIONID in conjunction with the header "ELO-Approved" is used to authenticate requests and maintain session information. Both values have to be sent in each request. The "ELO-Approved" header protects the server from CSRF attacks. Although the server returns the "ELO-Approved" value as cookie too (for some clients), it is only accepted if sent as header value.

Sessions are automatically invalidated after 10 minutes idle time. The server returns the error code "[EIX:2001]" if the session is unknown.

A session can be created by calling login or loginAdmin.

Since the passwords are transmitted in plain text, this functions should only be used in HTTPS connections.

Login example:

```
/IXServicePortIF/login

{
   "ci":{
      "language":"de",
      "country":"DE",
      "timeZone":"UTC",
      "ticket":"de.elo.ix.client.ticket_from_cookie"
   },
   "userName":"Administrator",
   "userPwd":"elo",
   "clientComputer":"REST Client",
   "runAs":""
}
```

Response:

```
HTTP/1.1 200
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: OPTIONS, HEAD, GET, POST, PUT
Access-Control-Allow-Headers: Authorization, Content-Type, Content-Length, Content-Dispositio
Set-Cookie: JSESSIONID=65934F6BC05C0CB725FFE2870F9EE1A3.ELO-PCIMIGELO21-1;
            Path=/ix-elo200; HttpOnly
Set-Cookie: ELO-Approved=616FABF4E701805574A4A3013192F545; Path=/ix-elo200
ELO-Approved: 616FABF4E701805574A4A3013192F545
Content-Type: application/json;charset=UTF-8
Content-Length: 1473
Date: Mon, 20 Dec 2021 22:56:33 GMT
```

```
Keep-Alive: timeout=20
Connection: keep-alive

{"result":{"clientInfo":{"callId":"",
                         "country":"DE",
                         "language":"de",
                         "ticket":"de.elo.ix.client.ticket_from_cookie",
                         "timeZone":"UTC","options":0},
            "ticketLifetime":486,
            "user":{"desc":"",
                    "flags":2013265919,
                    "groupList":[9998,9999],
                    "id":0,
                    "keylist":[
    ...
```

Authentication for Services

Authenication can also be performed by sending a Basic-Authorisation header or a Bearer-Header containing a session ticket. This authentication methods are discouraged in a browser because there is no CSRF protection. If an Authorisation header is provided, the first call checks the credentials (or ticket) and creates a connection which is internally cached in IX. The connection is reused for all subsequent calls from the same client IP address and the same Authorization header. An explicit login can be omitted. Furthermore it is not required to send a ClientInfo parameter in each call. A ClientInfo for language=en, country=US, timezone=UTC is used by default.

Enable CORS

To enabled CORS add the "Access-Control-Allow-*" headers as entries in the Indexserver config.xml. At least "Access-Control-Allow-Origin" is requried. Example:

```
<entry key="Access-Control-Allow-Origin">*</entry>
```

Examples

Examples

> **Information**
>
> ELO Server delivers an internal documentation with examples via the following url scheme: `https://<ixhost>:<port/path>/js-api-examples/<function>`.
>
> Example: `https://srvpdoksrvint02vm:9093/ix-EXTEN/js-api-examples/rest/index.html#read_file_and_blob`

login.html	Shows how to create a session.
insert-folder.html	FolderInsert a folder entry into the repository hierarchy.
insert-path.html	PathCreate a path in the repository.
insert-document.html	Insert a document entry into the repository hierarchy.
insert-document-basic-auth.html	Insert a document entry under basic authentication.
insert-doc-att.html	Insert a document entry inclusive attachment in one multipart/form-data POST.
esearch.html	Use FindByElastic to search in fulltext index.
write-map.html	Write and read Map Data with simple values and BLOB values.
invoke-rf.html	Invoke a Registered Function and pass an object serialized in JSON.

Example: Login

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login via REST Call</title>

<script src="rest-call.js"></script>

<script>

var xhr = new XHR(window);

// ClientInfo object with session attributes.
var clientInfo = {
   language : 'en',
   country : 'US',
   timeZone : 'UTC',
};
```

```
// This function shows how to login and logout.
async function onLogin() {

   var loginUser = document.getElementById('name').value;
   var loginPwd = document.getElementById('password').value;

   // Login
   try {
      var loginResult = await xhr.restcall('IXServicePortIF/login', {
         ci: clientInfo,
         userName: loginUser,
         userPwd: loginPwd,
         clientComputer: 'REST Client',
         runAs: ''
      });

      // Show login result in the browser's console.
      console.dir(loginResult);

      // Print the received result.
      document.getElementById('loginResult').value =
                JSON.stringify(loginResult, null, 2);

   }
   catch (exception) {
      console.dir(exception);
      document.getElementById('loginResult').value =
                exception.message;
   }

   // Logout
   await xhr.restcall('IXServicePortIF/logout', {
      ci : clientInfo
   });

}

</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Login with
         <a href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#login(de.elo.ix.client.ClientInfo,java.lang.String,
               java.lang.String,java.lang.String,java.lang.String)">
         login</a>
      </tr>
```

```
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">User Credentials</td>
      </tr>
      <tr>
         <td>Name</td>
         <td><input id="name" type="text" value="Administrator"></td>
      </tr>
      <tr>
         <td>Password</td>
         <td><input id="password" type="password" value="elo"></td>
      </tr>
      <tr>
         <td> </td>
         <td><button onclick="onLogin()">Login</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>loginResult</td>
         <td><textarea id="loginResult" cols="100" rows="20"
            spellcheck="false"></textarea></td>
      </tr>
   </table>
</body>
</html>
```

Example: Insert folder

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert Folder via REST Call</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);
```

```
// Testuser
const loginUser = 'Administrator';
const loginPwd = 'elo';

// ClientInfo object with session attributes.
var clientInfo = {
   language : 'de',
   country : 'DE'
};

async function onInsertFolder() {

   // Read constant values.
   // This object should be read only once since it does
   //  not change and contains a large number of items.
   const CONST = await
   xhr.restcall('IXServicePortIF/getConstants');

   // Login
   var loginResult = await
   xhr.restcall('IXServicePortIF/login', {
      ci : clientInfo,
      userName : loginUser,
      userPwd : loginPwd,
      clientComputer : 'REST Client',
      runAs : ''
   });

   // Initialize Sord with given parent ID and default folder mask.
   var parentId = document.getElementById('parentId').value;
   var editInfo = await
   xhr.restcall('IXServicePortIF/createSord', {
      parentId : parentId,
      maskId : CONST.DOC_MASK.GUID_FOLDER,
      editInfoZ : "sord.lean,maskNames,mask"
   });

   // Assign Sord properties.
   var sord = editInfo.sord;
   {
      // Name and date
      sord.name = document.getElementById('name').value;
      sord.XDateIso = document.getElementById('xdate').value;

      // Access
      sord.aclItems = [ {
         id : loginResult.user.id,
         access : CONST.ACCESS.LUR_ALL
```

```
         }, {
            id : CONST.USER_INFO.ID_EVERYONE,
            access : CONST.ACCESS.LUR_READ
         } ];

         // Index fields
         sord.objKeys[0].data = [ 'elo-index-123' ];
      }

      // Store Sord object.
      var objId = await
      xhr.restcall('IXServicePortIF/checkinSord', {
         sord : sord,
         sordZ : "name,xDateIso,aclItems,objKeys"
      });

      // Read Sord object.
      var editInfoR = await
      xhr.restcall('IXServicePortIF/checkoutSord', {
         objId : objId,
         editInfoZ : "sord.lean",
      });

      // Show Sord values on website.
      var sordR = editInfoR.sord;
      console.dir(sordR);
      document.getElementById('objId').value = sordR.id;
      document.getElementById('idate').value = sordR.IDateIso;
      document.getElementById('sord').value = JSON.stringify(sordR, null, 2);

      // Delete Sord object
      await xhr.purgeSord(objId);

      // Logout.
      await xhr.restcall('IXServicePortIF/logout', {});

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Insert Folder with <a
            href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinSord(de.elo.ix.client.ClientInfo,
                  de.elo.ix.client.Sord,de.elo.ix.client.SordZ,de.elo.ix.client.LockZ)"
            >checkinSord</a></td>
      </tr>
```

```
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Folder Properties</td>
      </tr>
      <tr>
         <td>Parent ID</td>
         <td><input id="parentId" type="text" value="1"></td>
      </tr>
      <tr>
         <td>Name</td>
         <td><input id="name" type="text" value="Smørrebrød"></td>
      </tr>
      <tr>
         <td>Date</td>
         <td><input id="xdate" type="text" value="2021-05-15-17-00-00"></td>
      </tr>
      <tr>
         <td> </td>
         <td><button onclick="onInsertFolder()">Insert Folder</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value=""></td>
      </tr>
      <tr>
         <td>Filing Date</td>
         <td><input id="idate" type="text" value=""></td>
      </tr>
      <tr>
         <td>Sord</td>
         <td><textarea id="sord" cols="100" rows="20" spellcheck="false"></textarea></td>
      </tr>
   </table>
</body>
</html>
```

Example: Insert Folder with checkinSordPath

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Create a Path in the Repository</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   async function onInsertPath() {

      // Read constant values.
      // This object should be read only once since it does not change
      // and contains a large number of items.
      const CONST = await
      xhr.restcall('IXServicePortIF/getConstants');

      // Login
      var loginResult = await
      xhr.restcall('IXServicePortIF/login', {
         userName : loginUser,
         userPwd : loginPwd,
         clientComputer : 'REST Client',
         runAs : ''
      });

      // Split given path at forward slashes into an array.
      // Create a Sord object for each path element.
      var pathNames = document.getElementById('path').value;
      var sords = [];
      pathNames.split('/').forEach(name => {
         sords.push({ name : name });
      });

      // Create the path.
      var ids = await   xhr.restcall('IXServicePortIF/checkinSordPath', {
         parentId : 1,
         sords,
         sordZ : CONST.EDIT_INFO.mbSord
      });
      var objId = ids[ids.length-1];
```

```
      // Read Sord object.
      var editInfoR = await xhr.restcall('IXServicePortIF/checkoutSord', {
         objId : objId,
         editInfoZ : CONST.EDIT_INFO.mbSord,
      });

      // Show Sord values on website.
      var sordR = editInfoR.sord;
      console.dir(sordR);
      document.getElementById('objId').value = sordR.id;
      document.getElementById('pathR').value = sordR.refPaths[0].pathAsString;
      document.getElementById('sord').value = JSON.stringify(sordR, null, 2);

   // Delete Sord object
   await xhr.purgeSord(objId);

      // Logout.
      await
      xhr.restcall('IXServicePortIF/logout', {});

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Insert Folder with <a
            href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinSordPath
               (de.elo.ix.client.ClientInfo,java.lang.String,
               de.elo.ix.client.Sord%5B%5D,de.elo.ix.client.SordZ)"
            >checkinSordPath</a></td>
      </tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Folder Properties</td>
      </tr>
      <tr>
         <td>Path</td>
         <td><input id="path" type="text"
         value="[illegible]">
         </td>
      </tr>
      <tr>
```

```
         <td> </td>
         <td><button onclick="onInsertPath()">Create Path</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value=""></td>
      </tr>
      <tr>
         <td>Repository Path</td>
         <td><input id="pathR" type="text" value=""></td>
      </tr>
      <tr>
         <td>Sord</td>
         <td><textarea id="sord" cols="100" rows="20" spellcheck="false">
         </textarea></td>
      </tr>
   </table>
</body>
</html>
```

Example: Insert Document with checkinDocEnd

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert Document</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   async function onInsertDocument() {

      // File to be uploaded.
```

```
var files = document.getElementById('file').files;
var versionFile = files[0];
if (!versionFile) {
   alert('Select a file to upload');
   return;
}

// Read constant values.
// This object should be read only once since it does not change
// and contains a large number of items.
const CONST = await
xhr.restcall('IXServicePortIF/getConstants');

// Login
var loginResult = await
xhr.restcall('IXServicePortIF/login', {
   userName : loginUser,
   userPwd : loginPwd,
   clientComputer : 'REST Client',
   runAs : ''
});

// Initialize Sord with given parent ID and default document mask.
var parentId = document.getElementById('parentId').value;
var editInfo = await
xhr.restcall('IXServicePortIF/createDoc', {
   parentId : parentId,
   maskId : CONST.DOC_MASK.GUID_DOCUMENT,
   editInfoZ : CONST.EDIT_INFO.mbSord
});

// Assign Sord properties.
var sord = editInfo.sord;
sord.name = document.getElementById('name').value;
sord.XDateIso = document.getElementById('xdate').value;

// Assign original file name.
var okeyFName = {
   id : CONST.DOC_MASK_LINE.ID_FILENAME,
   name : CONST.DOC_MASK_LINE.NAME_FILENAME,
   data : [versionFile.name]
};
sord.objKeys.push(okeyFName);

// Provide a DocVersion object and assign the stream reference.
var version = {
   version : '1.0',
   comment : 'Version uploaded via REST call.',
   contentType : versionFile.type,
```

```
   fileData : {}
};

// Upload using multipart/form-data?
var uploadViaFormData =
          document.getElementById('uploadStrategy').options[0].selected;
if (uploadViaFormData) {

   /* Reference the file by the field name
   used in the FormData object (see restcallWithStreams). */
   version.fileData.stream = { streamId : 'files[0]' };

   // Checkin document.
   var doc = await
   xhr.restcallWithStreams('IXServicePortIF/checkinDocEnd', {
      sord : sord,
      sordZ : CONST.SORD.mbAll,
      document : {
         docs : [ version ]
      },
      unlockZ : CONST.LOCK.NO
   }, files);                // pass files here

   objId = doc.objId;

}
else {
   // Upload the file to a temporary storage on the server.
   // It is kept there for a few minutes.
   var streamReference = await   xhr.upload(versionFile);
   version.fileData.stream = streamReference;

   // Checkin document.
   var doc = await xhr.restcall('IXServicePortIF/checkinDocEnd', {
      sord : sord,
      sordZ : CONST.SORD.mbAll,
      document : {
         docs : [ version ]
      },
      unlockZ : CONST.LOCK.NO
   });

   objId = doc.objId;
}

// Read Sord object.
var editInfoR = await xhr.restcall('IXServicePortIF/checkoutSord', {
   objId : objId,
   editInfoZ : CONST.EDIT_INFO.mbSordContentStream,
```

```
      });
      console.dir(editInfoR);

      // Download file into blob to be able to view it in an IFRAME
      // afte the sord has been deleted.
      var fileBlob = await xhr.download(editInfoR.sord.docVersion.fileData.stream.url);
      var fileUrl = URL.createObjectURL(fileBlob);
      document.getElementById('fileContent').src = fileUrl;

      // Show Sord values on website.
      var sordR = editInfoR.sord;
      document.getElementById('objId').value = sordR.id;
      document.getElementById('idate').value = sordR.IDateIso;
      document.getElementById('sord').value = JSON.stringify(sordR, null, 2);

      // Delete sord finally.
      await xhr.purgeSord(objId);

      // Logout.
      await xhr.restcall('IXServicePortIF/logout', {});

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Insert Document with <a
            href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinDocEnd(de.elo.ix.client.ClientInfo,
               de.elo.ix.client.Sord,de.elo.ix.client.SordZ,
               de.elo.ix.client.Document,de.elo.ix.client.LockZ)"
            >checkinDocEnd</a></td>
      </tr>
      <tr>
         <td>Upload strategy</td>
         <td><select id="uploadStrategy">
               <option>FormData</option>
               <option>Separate Upload</option>
         </select></td>
      </tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Document Properties</td>
      </tr>
```

```
      <tr>
         <td>Parent ID</td>
         <td><input id="parentId" type="text" value="1"></td>
      </tr>
      <tr>
         <td>Name</td>
         <td><input id="name" type="text" value="Smørrebrød"></td>
      </tr>
      <tr>
         <td>Date</td>
         <td><input id="xdate" type="text" value="2021-05-15-17-00-00"></td>
      </tr>
      <tr>
         <td>File</td>
         <td><input id="file" type="file" ></td>
      </tr>
      <tr>
         <td> </td>
         <td><button onclick="onInsertDocument()">Insert
               Document</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value=""></td>
      </tr>
      <tr>
         <td>Filing Date</td>
         <td><input id="idate" type="text" value=""></td>
      </tr>
      <tr>
         <td>File Content</td>
         <td><iframe id="fileContent"></iframe></td>
      </tr>
      <tr>
         <td>Sord</td>
         <td><textarea id="sord" cols="100" rows="20" spellcheck="false"></textarea></td>
      </tr>
   </table>

   <br />
   <div id="ticket"></div>
```

```
</body>
</html>
```

Example: Insert Document under BASIC authentication

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert Document under BASIC authentication</title>

<script src="rest-call-basic-auth.js"></script>

<script>
   // Indexserver URL, e.g. http://localhost:9090/ix-elo230/ix
   const ixUrl = [ "/", window.location.pathname.split("/")[1], "/ix"].join("");

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   var xhr = new XHR(ixUrl, loginUser, loginPwd);

   // ClientInfo object with session attributes.
   // var clientInfo = not requried, default: language=en, country=US, timeZone=UTC

   async function onInsertDocument() {

      // File to be uploaded.
      var files = document.getElementById('file').files;
      var versionFile = files[0];
      if (!versionFile) {
         alert('Select a file to upload');
         return;
      }

      // Read constant values.
      // This object should be read only once since it does not change
      // and contains a large number of items.
      const CONST = await
      xhr.restcall('IXServicePortIF/getConstants');

      // Initialize Sord with given parent ID and default document mask.
      var parentId = document.getElementById('parentId').value;
      var editInfo = await
      xhr.restcall('IXServicePortIF/createDoc', {
         parentId : parentId,
         maskId : CONST.DOC_MASK.GUID_DOCUMENT,
```

```
      editInfoZ : CONST.EDIT_INFO.mbSord
   });

   // Assign Sord properties.
   var sord = editInfo.sord;
   sord.name = document.getElementById('name').value;
   sord.XDateIso = document.getElementById('xdate').value;

// Assign original file name.
var okeyFName = {
  id : CONST.DOC_MASK_LINE.ID_FILENAME,
  name : CONST.DOC_MASK_LINE.NAME_FILENAME,
  data : [versionFile.name]
};
sord.objKeys.push(okeyFName);

   // Provide a DocVersion object and assign the stream reference.
   var version = {
      version : '1.0',
      comment : 'Version uploaded via REST call.',
      contentType : versionFile.type,
      fileData : {}
   };

   // Upload using multipart/form-data?
   var uploadViaFormData =
            document.getElementById('uploadStrategy').options[0].selected;
   if (uploadViaFormData) {

      /* Reference the file by the field name
     used in the FormData object (see restcallWithStreams).*/
      version.fileData.stream = { streamId : 'files[0]' };

      // Checkin document.
      var doc = await
      xhr.restcallWithStreams('IXServicePortIF/checkinDocEnd', {
         sord : sord,
         sordZ : CONST.SORD.mbAll,
         document : {
            docs : [ version ]
         },
         unlockZ : CONST.LOCK.NO
      }, files);                // pass files here

      objId = doc.objId;

   }
   else {
      // Upload the file to a temporary storage on the server.
```

```
         // It is kept there for a few minutes.
         var streamReference = await   xhr.upload(versionFile);
         version.fileData.stream = streamReference;

         // Checkin document.
         var doc = await xhr.restcall('IXServicePortIF/checkinDocEnd', {
            sord : sord,
            sordZ : CONST.SORD.mbAll,
            document : {
               docs : [ version ]
            },
            unlockZ : CONST.LOCK.NO
         });

         objId = doc.objId;
      }

      // Read Sord object.
      var editInfoR = await xhr.restcall('IXServicePortIF/checkoutSord', {
         objId : objId,
         editInfoZ : CONST.EDIT_INFO.mbSordContentStream,
         lockZ : CONST.LOCK.NO
      });
      console.dir(editInfoR);

      // Download file into blob to be able to view it in an IFRAME
      // afte the sord has been deleted.
      var fileBlob = await xhr.download(editInfoR.sord.docVersion.fileData.stream.url);
      var fileUrl = URL.createObjectURL(fileBlob);
      document.getElementById('fileContent').src = fileUrl;

      // Show Sord values on website.
      var sordR = editInfoR.sord;
      document.getElementById('objId').value = sordR.id;
      document.getElementById('idate').value = sordR.IDateIso;
      document.getElementById('sord').value = JSON.stringify(sordR, null, 2);

      // Delete sord finally.
      await xhr.purgeSord(objId);
   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Insert Document under BASIC authentication with <a
            href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinDocEnd(de.elo.ix.client.ClientInfo,
```

```
                                                de.elo.ix.client.Sord,
                                                de.elo.ix.client.SordZ,
                                                de.elo.ix.client.Document,
                                                de.elo.ix.client.LockZ)"
      >checkinDocEnd</a></td>
</tr>
<tr>
   <td>Upload strategy</td>
   <td><select id="uploadStrategy">
         <option>FormData</option>
         <option>Separate Upload</option>
   </select></td>
</tr>
<tr>
   <td colspan="2"> </td>
</tr>

<tr>
   <td colspan="2">Document Properties</td>
</tr>
<tr>
   <td>Parent ID</td>
   <td><input id="parentId" type="text" value="1"></td>
</tr>
<tr>
   <td>Name</td>
   <td><input id="name" type="text" value="Smørrebrød"></td>
</tr>
<tr>
   <td>Date</td>
   <td><input id="xdate" type="text" value="2021-05-15-17-00-00"></td>
</tr>
<tr>
   <td>File</td>
   <td><input id="file" type="file" ></td>
</tr>
<tr>
   <td> </td>
   <td><button onclick="onInsertDocument()">Insert
         Document</button></td>
</tr>

<tr>
   <td colspan="2"> </td>
</tr>

<tr>
   <td colspan="2">Result</td>
</tr>
```

```
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value=""></td>
      </tr>
      <tr>
         <td>Filing Date</td>
         <td><input id="idate" type="text" value=""></td>
      </tr>
      <tr>
         <td>File Content</td>
         <td><iframe id="fileContent"></iframe></td>
      </tr>
      <tr>
         <td>Sord</td>
         <td><textarea id="sord" cols="100" rows="20" spellcheck="false">
             </textarea></td>
      </tr>
   </table>

   <br />
   <div id="ticket"></div>
</body>
</html>
```

Example: Fulltext Search

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Fulltext Search</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   /**
    * Simple search for terms over title, fulltext, etc.
    */
   async function onSearchSimple() {

      // Search terms
      var query = document.getElementById('query').value;
```

```
   // Where to search for the given terms.
   // Can be specified as integer value with a combination of ESearchParamsC values.
   // Furthermore, a comma separated list of ESearchParamsC values is accepted too.
   var searchIn = "";
if (document.getElementById('searchIn1').checked) searchIn += "TITLE,";
if (document.getElementById('searchIn2').checked) searchIn += "FULLTEXT,";
if (document.getElementById('searchIn3').checked) searchIn += "INDEX_FIELDS,";
if (document.getElementById('searchIn4').checked) searchIn += "EXTRA_TEXT,";
if (document.getElementById('searchIn5').checked) searchIn += "FEED,";

var findInfo = {

     // FindByESearch
  findByESearch : {

     // ESearchOptions
     searchOptions : {
   },

   // ESearchParams
   searchParams : {
      query : query,
      searchIn : searchIn
   }
  }
};

   onSearch(findInfo);
}

/**
 * Search over index fields.
 */
async function onSearchMails() {

   // Search criteria
   var query = document.getElementById('queryContent').value;
var mailFrom = document.getElementById('mailFrom').value;
var mailTo = document.getElementById('mailTo').value;

var findInfo = {
   findByESearch : {

      // ESearchOptions
     searchOptions : {
     },

     // ESearchParams
```

```
      searchParams : {

         query: query,
         searchIn: "FULLTEXT",

         // QueryOperator
       queryOperator : {

          // Element operatorType helps to decide which class has to
          // be deserialized on the server side:
          // AndOperator, OrOperator, or NotOperator. This element is not
          // declared in the API and is only required for REST calls.
         operatorType : 'AND',

         // List<QueryObject> queryObjectList
         queryObjectList : [

           // QueryFilter for mailFrom
          {
           searchField : "INDEXFIELD",   // SearchFieldE
           indexFieldKey : "ELOOUTL1",   // Index field name
           fieldType : "tokenized",     // FieldTypeE
           value : { value : mailFrom }
           // FilterValue, here: deserialized as StringSingleValue
          }
          ,
          // QueryFilter for mailTo
          {
             searchField : "INDEXFIELD",
             indexFieldKey : "ELOOUTL2",
             fieldType : "tokenized",
             value : { value : mailTo }
          }
         ]
       }

      }

   }
};

onSearch(findInfo);
}

/**
 * Perform the search and show the results.
 */
async function onSearch(findInfo) {
```

```
      // Login
      var loginResult = await
      xhr.restcall('IXServicePortIF/login', {
         userName : loginUser,
         userPwd : loginPwd,
         clientComputer : 'REST Client',
         runAs : ''
      });

      // Store the received clientInfo object.
      // The clientInfo object has to be supplied as parameter 'ci' in every request.
      clientInfo = loginResult.clientInfo;

      // Search
      try {

         var findResult = await xhr.restcall('IXServicePortIF/findFirstSords', {
            findInfo : findInfo,
            max : 100,
            sordZ : "name,xDateIso,aclItems,objKeys",
         });

         // Pretty-print the result
         document.getElementById('resultObject').value = JSON.stringify(findResult, null, 2);
      }
      catch (ex) {
         alert(ex.message);
      }

      // Logout.
      await
      xhr.restcall('IXServicePortIF/logout', {
         ci : clientInfo
      });

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Fulltext Search</td></tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

   <tr>
     <td colspan="2">Simple</td>
```

```
</tr>
<tr>
  <td>Query term(s)</td>
  <td><input id="query" type="text" value="ELOprofessional Requirements"></td>
</tr>
<tr>
  <td>Search in</td>
  <td>
   <input type="checkbox" id="searchIn1" /><label for="searchIn1">TITLE</label><br>
   <input type="checkbox" id="searchIn2" /><label for="searchIn2">FULLTEXT</label><br>
   <input type="checkbox" id="searchIn3" /><label for="searchIn3">INDEX_FIELDS</label><br>
   <input type="checkbox" id="searchIn4" /><label for="searchIn4">EXTRA_TEXT</label><br>
   <input type="checkbox" id="searchIn5" /><label for="searchIn5">FEED</label><br>
  </td>
</tr>
<tr>
  <td> </td>
  <td><button onclick="onSearchSimple()">Search</button></td>
</tr>

<tr>
  <td colspan="2"> </td>
</tr>
<tr>
  <td colspan="2">Search for mails</td>
</tr>
<tr>
  <td>About</td>
  <td><input id="queryContent" type="text" value="ELOprofessional Requirements">
  </td>
</tr>
<tr>
  <td>From</td>
  <td><input id="mailFrom" type="text" value="Alice">
  </td>
</tr>
<tr>
  <td>To</td>
  <td><input id="mailTo" type="text" value="Bob">
  </td>
</tr>
   <tr>
      <td> </td>
      <td><button onclick="onSearchMails()">Search</button></td>
   </tr>

   <tr>
      <td colspan="2"> </td>
   </tr>
```

```
      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Result Object</td>
         <td><textarea id="resultObject" cols="100" rows="20" spellcheck="false">
         </textarea></td>
      </tr>
   </table>
</body>
</html>
```

Example: Insert Document and Attachment

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert Document and Attachment</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   async function onInsertDocument() {

      // Files to be uploaded.
      var versionFile = document.getElementById('versionFile').files[0];
      var attachmentFile = document.getElementById('attachmentFile').files[0];
      var files = [versionFile, attachmentFile];
      if (!versionFile || !attachmentFile) {
         alert('Select files to upload');
         return;
      }

      // Read constant values.
      // This object should be read only once since it does not change
      // and contains a large number of items.
      const CONST = await
      xhr.restcall('IXServicePortIF/getConstants');

      // Login
```

```
var loginResult = await
xhr.restcall('IXServicePortIF/login', {
   userName : loginUser,
   userPwd : loginPwd,
   clientComputer : 'REST Client',
   runAs : ''
});

// Initialize Sord with given parent ID and default document mask.
var parentId = document.getElementById('parentId').value;
var editInfo = await
xhr.restcall('IXServicePortIF/createDoc', {
   parentId : parentId,
   maskId : CONST.DOC_MASK.GUID_DOCUMENT,
   editInfoZ : CONST.EDIT_INFO.mbSord
});

// Assign Sord properties.
var sord = editInfo.sord;
sord.name = document.getElementById('name').value;

// Provide DocVersion objects with stream reference
var version = {
   version : '1.0',
   comment : 'Version uploaded via REST call.',
   contentType : versionFile.type,
   fileData : {
      stream : { streamId : 'files[0]' }
   }
};
var attachment = {
   version : '1.0',
   comment : 'Attachment uploaded via REST call.',
   contentType : attachmentFile.type,
   fileData : {
      stream : { streamId : 'files[1]' }
   }
};

// Checkin document.
var doc = await
xhr.restcallWithStreams('IXServicePortIF/checkinDocEnd', {
   sord : sord,
   sordZ : CONST.SORD.mbAll,
   document : {
      docs : [ version ],
      atts : [ attachment ]
   },
   unlockZ : CONST.LOCK.NO
```

```
      }, files);                // pass files here

      objId = doc.objId;

      // Read Sord object.
      var editInfoR = await xhr.restcall('IXServicePortIF/checkoutDoc', {
         objId : objId,
         editInfoZ : CONST.EDIT_INFO.mbSordDocAtt,
         lockZ : CONST.LOCK.NO
      });
      console.dir(editInfoR);

      // Show Sord values on website.
      document.getElementById('objId').value = editInfoR.sord.id;
      document.getElementById('versionSize').value = editInfoR.document.docs[0].size
      document.getElementById('attachmentSize').value = editInfoR.document.atts[0].size;
      document.getElementById('editInfo').value = JSON.stringify(editInfoR, null, 2);

      // Delete sord finally.
      await xhr.purgeSord(objId);

      // Logout.
      await xhr.restcall('IXServicePortIF/logout', {});

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Insert Document and Attachment with <a
            href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinDocEnd(de.elo.ix.client.ClientInfo,
               de.elo.ix.client.Sord,de.elo.ix.client.SordZ,
               de.elo.ix.client.Document,de.elo.ix.client.LockZ)"
            >checkinDocEnd</a></td>
      </tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Document Properties</td>
      </tr>
      <tr>
         <td>Parent ID</td>
         <td><input id="parentId" type="text" value="1"></td>
      </tr>
```

```
      <tr>
         <td>Name</td>
         <td><input id="name" type="text" value="Smørrebrød"></td>
      </tr>
      <tr>
         <td>Version File</td>
         <td><input id="versionFile" type="file" ></td>
      </tr>
      <tr>
         <td>Attachment File</td>
         <td><input id="attachmentFile" type="file" ></td>
      </tr>
      <tr>
         <td> </td>
         <td><button onclick="onInsertDocument()">Insert
               Document</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value=""></td>
      </tr>
      <tr>
         <td>Version Size</td>
         <td><input id="versionSize" type="text" value=""></td>
      </tr>
      <tr>
         <td>Attachment Size</td>
         <td><input id="attachmentSize" type="text" value=""></td>
      </tr>
      <tr>
         <td>Sord/Document/Attachment</td>
         <td><textarea id="editInfo" cols="100" rows="20" spellcheck="false">
             </textarea></td>
      </tr>
   </table>

   <br />
   <div id="ticket"></div>
</body>
</html>
```

Example: Read/Write Map Data via REST Calls

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Read/Write Map Data via REST Calls</title>

<script src="rest-call.js"></script>
<script src="../base64.js"></script>

<script>
   var xhr = new XHR(window);

   // Testuser
   const loginUser = 'Administrator';
   const loginPwd = 'elo';

   async function onWriteMap() {

      // Read constant values.
      // This object should be read only once since it does not change
      // and contains a large number of items.
      const CONST = await   xhr.restcall('IXServicePortIF/getConstants');

      // Maximum length of a map value if not stored as BLOB
      var maxValueLength = CONST.MAP_DATA.lnValue;

      // Maximum length of a map value stored as BLOB
      var maxBlobLength = CONST.FILE_DATA.MAX_BLOB_LENGTH;

      // Get the values to be used as map data from the input fields

      // The simpleValue will be passed in String member MapValue.value
      var simpleValue = document.getElementById('simpleValue').value;
      if (simpleValue.length >= maxValueLength) {
         throw new Error('simpleValue exceeds ' + maxValueLength);
      }

      // The bufferValue will be passed as Base64 encoded BLOB value.
      // Before encoding to Base64, String values used in BLOBs should be encoded
      // to UTF-8 to retain accented characters.
      var bufferValue = document.getElementById('bufferValue').value;
      var bufferBase64 = Base64.encode(bufferValue); // converts String -> UTF-8 -> Base64
      var bufferUtf8Length = bufferBase64.length * 2 / 3;
      if (bufferUtf8Length * 2 / 3 > maxBlobLength) {
         throw new Error("bufferValue exceeds " + maxBlobLength);
      }
```

```
// The fileValue will be passed as stream.
var fileValue = document.getElementById('fileValue').files[0];
if (!fileValue) {
   alert('Select a file to upload');
   return;
}
if (fileValue.size > maxBlobLength) {
   throw new Error("fileValue exceeds " + maxBlobLength);
}

// Login
var loginResult = await xhr.restcall('IXServicePortIF/login', {
   userName : loginUser,
   userPwd : loginPwd,
   clientComputer : 'REST Client',
   runAs : ''
});

// Insert a Sord to which the map data will be associated.
var parentId = document.getElementById('parentId').value;
var objId = await xhr.restcall('IXServicePortIF/createSord', {
   parentId : parentId,
   maskId : CONST.DOC_MASK.GUID_FOLDER,
   editInfoZ : CONST.EDIT_INFO.mbSord
}).then(editInfo => {
   editInfo.sord.name = 'Write Map ' + new Date();
   return xhr.restcall('IXServicePortIF/checkinSord', {
      sord : editInfo.sord,
      sordZ : CONST.SORD.mbAll,
      unlockZ : CONST.LOCK.NO
   });
});
document.getElementById('objId').value = objId;

// Create a MapValue object for each value, blob, stream to be stored.
var mapValues = [
   {
      key : 'simpleValue',
      value : simpleValue,
   },
   {
      key : 'bufferValue',
      blobValue : {
         contentType : 'text/plain',
         data : bufferBase64
      }
   },
   {
      key : 'fileValue',
```

```
            blobValue : {
               contentType : fileValue.type,
               stream : { streamId : 'files[0]' }

               // As an alternative to a multipart/form-data request,
               // the file content can also be uploaded in a temporary storage
               // on the server before checkinMap is called:
               // stream : await xhr.upload(fileValue)

            }
         },
      ];

      // Insert the map data
      await xhr.restcallWithStreams('IXServicePortIF/checkinMap', {
         domainName : CONST.MAP_DOMAIN.DOMAIN_SORD,
         id : objId,
         objId : objId,
         data : mapValues
      }, [fileValue]);

      // Read the map data
      var mapData = await xhr.restcall('IXServicePortIF/checkoutMap', {
         domainName : CONST.MAP_DOMAIN.DOMAIN_SORD,
         id : objId,
         keyNames : ['simpleValue', 'bufferValue', 'fileValue']
      });
      document.getElementById('mapData').value = JSON.stringify(mapData, null, 2);

      // Display simpleValue
      var simpleValueR = mapData.mapItems['simpleValue'].value;
      document.getElementById('simpleValueR').value = simpleValueR;

      // Download the BLOB value for bufferValue.
      var bufferStream = mapData.mapItems['bufferValue'].blobValue.stream;
      var bufferBlob = await xhr.download(bufferStream.url);
      var bufferValueR = await bufferBlob.text();
      document.getElementById('bufferValueR').value = bufferValueR;

      // Download the BLOB value for fileValue.
      var fileStream = mapData.mapItems['fileValue'].blobValue.stream;
      var fileBlob = await xhr.download(fileStream.url);
      var fileUrl = URL.createObjectURL(fileBlob);
      document.getElementById('fileValueR').src = fileUrl;

      // Delete Sord object
      await xhr.purgeSord(objId);

      // Logout.
```

```
      await xhr.restcall('IXServicePortIF/logout', {});

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Read/Write Map Data via REST Calls
         <a href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkinMap(de.elo.ix.client.ClientInfo,
                                               java.lang.String,
                                               java.lang.String,
                                               int,
                                               de.elo.ix.client.KeyValue%5B%5D,
                                               de.elo.ix.client.LockZ)">checkinMap</a>

         <a href="../../doc/EloixClient-javadoc.jar/de/elo/ix/client/
               IXServicePortIF.html#checkoutMap(de.elo.ix.client.ClientInfo,
                                                java.lang.String,java.lang.String,
                                                java.lang.String%5B%5D,
                                                de.elo.ix.client.LockZ)">checkoutMap</a>
         </td>
      </tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Map Data Properties</td>
      </tr>
      <tr>
         <td>Parent ID</td>
         <td><input id="parentId" type="text" value="1"></td>
      </tr>
      <tr>
         <td>Simple Value</td>
         <td><input id="simpleValue" type="text" value="Øresundsbron"></td>
      </tr>
      <tr>
         <td>Buffer Value</td>
         <td><input id="bufferValue" type="text" value="Smørrebrød for 2,60€" ></td>
      </tr>
      <tr>
         <td>File Value</td>
         <td><input id="fileValue" type="file" ></td>
      </tr>
      <tr>
```

```
         <td> </td>
         <td><button onclick="onWriteMap()">Write Map</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Object ID</td>
         <td><input id="objId" type="text" value="1"></td>
      </tr>
      <tr>
         <td>Simple Value</td>
         <td><input id="simpleValueR" type="text" value=""></td>
      </tr>
      <tr>
         <td>Buffer Value</td>
         <td><input id="bufferValueR" type="text" value=""></td>
      </tr>
      <tr>
         <td>File Value</td>
         <td><iframe id="fileValueR"></iframe></td>
      </tr>
      <tr>
         <td>Map Data</td>
         <td><textarea id="mapData" cols="100" rows="20" spellcheck="false">
            </textarea></td>
      </tr>
   </table>
</body>
</html>
```

Example: Invoke Registered Function

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Invoke a Registered Function</title>

<script src="rest-call.js"></script>

<script>
   var xhr = new XHR(window);
```

```
// Testuser
const loginUser = 'Administrator';
const loginPwd = 'elo';

async function onInvokeRF() {

   // Login
   var loginResult = await
   xhr.restcall('IXServicePortIF/login', {
      userName : loginUser,
      userPwd : loginPwd,
      clientComputer : 'REST Client',
      runAs : ''
   });

   // Store the received clientInfo object.
   // The clientInfo object has to be supplied as parameter 'ci' in every request.
   clientInfo = loginResult.clientInfo;

   // This object is sent to the Registered Function
   var paramObject = {
      member1 : 123,
      member2 : "ABC"
   };

   // Object has to be serialized into a JSON String before it can be sent.
   var paramString = JSON.stringify(paramObject);

   // Invoke the Registered Function
   try {
      var functionName = document.getElementById('rfname').value;
      var resultString = await xhr.restcall(
         'IXServicePortIF/executeRegisteredFunctionString', {
               functionName : functionName,
               param : paramString
      });

      // Deserialize the result object.
      var resultObject = JSON.parse(resultString);

      // Pretty-print the result
      document.getElementById('resultObject').value =
            JSON.stringify(resultObject, null, 2);
   }
   catch (ex) {
      alert(ex.message);
   }
```

```
      // Logout.
      await
      xhr.restcall('IXServicePortIF/logout', {
         ci : clientInfo
      });

   }
</script>

</head>
<body>
   <table>
      <tr>
         <td colspan="2">Invoke Registered Function</td></tr>
      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Folder Properties</td>
      </tr>
      <tr>
         <td>Registered Function</td>
         <td><input id="rfname" type="text" value="RF_echo">
         </td>
      </tr>
      <tr>
         <td> </td>
         <td><button onclick="onInvokeRF()">Invoke</button></td>
      </tr>

      <tr>
         <td colspan="2"> </td>
      </tr>

      <tr>
         <td colspan="2">Result</td>
      </tr>
      <tr>
         <td>Result Object</td>
         <td><textarea id="resultObject" cols="100" rows="20" spellcheck="false">
            </textarea></td>
      </tr>
   </table>
</body>
</html>
```

ELO REST Service

Introduction

> **Beachten Sie**
>
> Diese Dokumentation ist nur auf Englisch verfügbar.

> **Important**
>
> The ELO REST API Service is DEPRECATED.
>
> The ELO Indexserver offers a REST interface for ELO 20 or higher.
>
> For developments in Java and .NET we recommend using the ELO Indexserver client library. Other programming platforms can address the ELO Indexserver REST interface.

The ELO REST Service enables you to easily integrate other systems with ELO via the ELO REST API interface using programming.

This documentation contains the following topics:

- Authentication
- Search
- Files
- Metadata
- Members
- Misc
- Masks/Metadata forms

Overview

Interactive UI

It is highly recommended to experiment with the interactive specification by opening the browser at the following URL:

```
http(s)://<elo-server>/rest-<repo>/doc
```

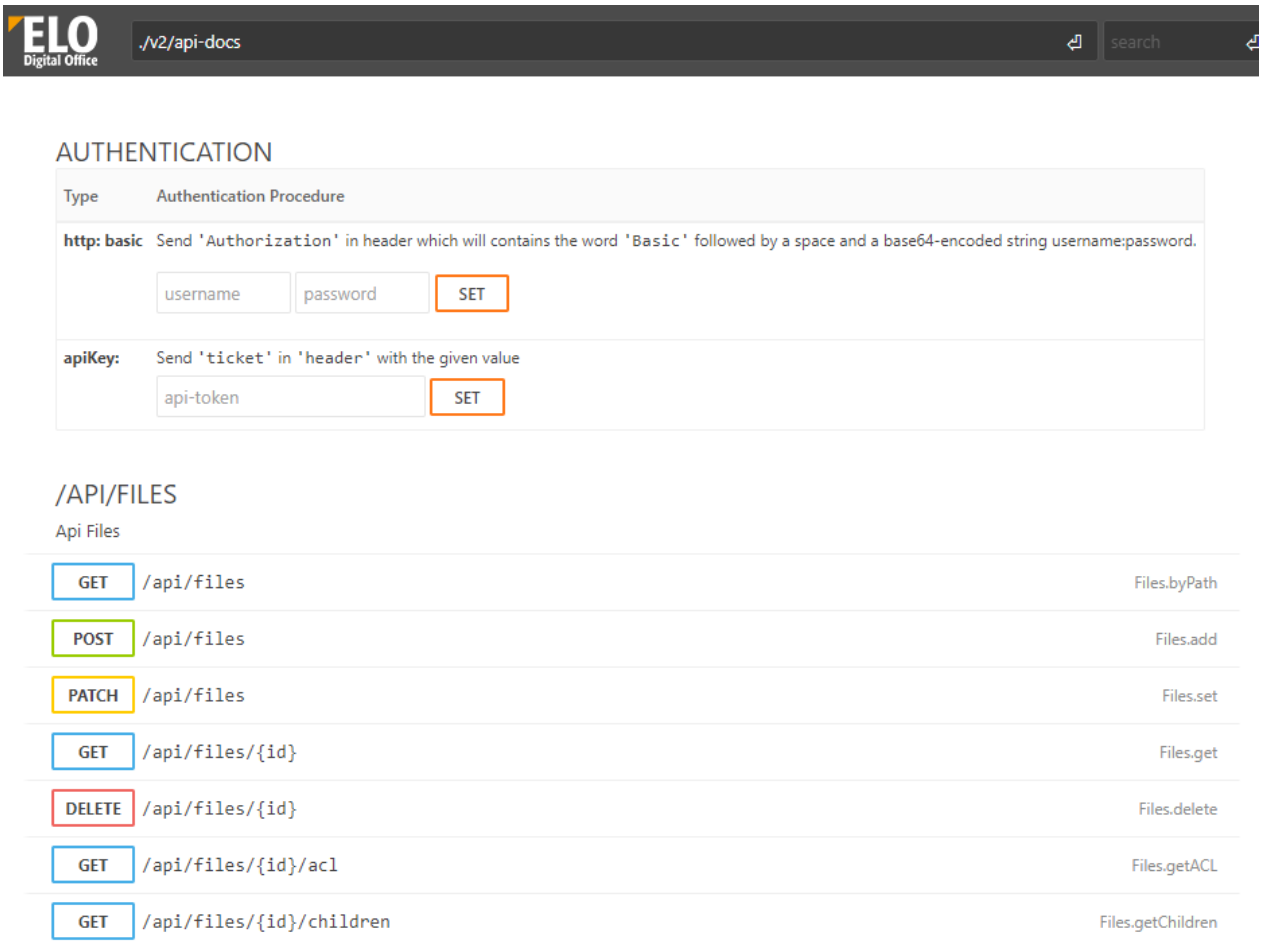

Fig.: ELO REST API specification

You will see the list of endpoints and can try them out directly by sending requests and receiving live responses.

Specification

The specification for the REST API is self generated and follows the *OpenAPI 3* standard. Sometimes it is useful to access the raw JSON specification, for example if used by some tools. This raw JSON specification is available at:

```
http(s)://<elo-server>/rest-<repo>/v3/api-docs
```

Scope

The goal of this API is *not* to cover all the IX's functionality. Rather, it is to offer a simple way to perform common operations.

The API covers the following areas:

- files
  - basic information
  - download/upload
  - metadata *(also known as "keywording")*
    - IX fields
    - map fields
  - child entries *(for directories)*
  - versions *(for documents)*
  - ACL *(read-only)*
- basic search
  - by fulltext
- by metadata
- members
  - users
  - groups
- system
  - masks
  - colors
  - registered functions (IX)

Design

Most endpoints follow a similar pattern:

- `GET /api/.../{id}`
  - Gets an item.
  - Both `id` or `guid` can be used in the URL path.
- `POST /api/...`
  - Adds a new item.
  - The returned value will be of the form `{"id":..., "guid":..., "name":...}`.
- `PATCH /api/...`
  - Updates an item.
  - The `id` or `guid` *must* be present in the body.
- `DELETE /api/.../{id}`
  - Deletes an item.
  - Both `id` or `guid` can be used in the URL path.

For every endpoint containing `{id}` *in the URL*, both `id` or `guid` can be used interchangeably. Sometimes, where the IX internally allows it, the identifying "name" can be used too. This applies for example for user names, mask names, color names...

CORS

If the web app is not hosted on the same Tomcat, setting up CORS will be necessary to enable remote websites to access the API.

Instead of configuring CORS for the whole Tomcat, it can be set explicitly for the REST module.

In the `config.properties`, simply add:

```
allowedOrigins = https://example.com, https://another.net
```

> **Please note**
>
> Using the * wildcard is possible but should only be enabled during development/testing.
>
> It is highly discouraged in production because of obvious security concerns.

Authentication

Backend apps authentication

All API endpoints require authentication, this can be either done using:

- *Basic authentication* with the ELO user name/password
- a valid *ticket* in either of:
  - the query (adding &ticket=...)
  - a "ticket" header
  - a "ticket" cookie

"External" Browser apps - login/logout

Instead of providing authentication each time, it is also possible to login directly to establish a session between the browser and the API.

Browser sessions are disabled by default. To enable them, edit the configuration located at:

```
<elo-install-path>/config/rest-.../<server-name>/config.properties
```

Update it with following values:

```
allowSessions = true
allowedOrigins = https://example.com, https://another.net
```

Then restart the REST API using the Tomcat Manager.

*Note that for security reasons, `allowSessions` is not compatible with the "wildcard" origin (`allowedOrigins = *`).

> **Information**
>
> If you want to enable browser sessions externally, you have to explicitly list the allowed origins.

Once properly configured, users can login:

```
POST  /login
```

```
{
    "username":"Alice",
    "password":"secret",
    "lang":"en"
}
```

This session follows the usual rules of the Tomcat. It typically expires after 10 minutes of inactivity.

The `lang` parameter specifies the language to connect with, which is the two-letter code of any of the supported client languages. This influences translation keys and error messages.

To logout, invoke:

```
POST  /logout
```

"Internal" Browser apps - sharing the Web Client session

If you invoke the API through `https://.../ix-MyRepo/plugin/de.elo.ix.plugin.proxy/rest/api/...`, the session of the Web Client will be shared and authentication will be automatically performed.

In other words, you do not have to provide credentials or call `/login` by using such URLs if the user is already logged in using the Web Client (or another ELO web application).

Note that this solution is *not* compatible with cross origin use cases. This only works if the web app is configured to run behind the IX Proxy Plugin, like "ELO Apps".

Alternatively, you could deploy a classic web app on the Tomcat, and edit the IX Proxy Plugin configuration located at:

```
<elo-install-path>/config/ix-.../<server-name>/de.elo.ix.plugin.proxy.properties
```

There, you can add a forwarding URL to your web app:

```
yourapp=http(s)://any-valid-url
```

After restarting the IX, all requests to `https://.../ix-MyRepo/plugin/de.elo.ix.plugin.proxy/yourapp/...` would be forwarded to your web app URL and calls to the REST API would share the Web Client session.

Search

Search "anywhere"

> **Information**
>
> This example searches all documents and folders containing "elo" anywhere:
>
> the name, the document's content or the associated metadata/keywording.

Request:

```
GET  /api/search?words=elo
```

Response:


```
[
    {
        "guid": "(25A7C0C1-DA8C-176C-50EC-778215E6D69C)",
        "id": 31,
        "name": "elo.profile",
        "type": 3,
        "isDir": true,
        "desc": "",
        "lock": "",
        "ownerName": "ELO Service",
        "access": "RWD-LP",
        "parentId": 30,
        "dateArchived": "2020-10-26T10:09:00Z",
        "dateCustom": null,
        "dateModified": "2020-10-26T09:09:44Z"
    },
    ...
    {
        "guid": "(C1F90D1D-2C4B-D03E-475E-3C8ADCB26AD4)",
        "id": 40,
        "name": "ELO Service",
        "type": 23,
        "isDir": true,
        "desc": "",
        "lock": "",
        "ownerName": "ELO Service",
        "access": "RWD-LP",
        "parentId": 28,
        "dateArchived": "2020-10-26T10:09:00Z",
        "dateCustom": null,
        "dateModified": "2020-11-12T16:14:02Z"
```

```
        }
    ]
```

It is possible to restrict this, for example by using one of the following:

```
GET  /api/search?words=elo&where=TITLE
GET  /api/search?words=elo&where=DOCUMENT
GET  /api/search?words=elo&where=KEYWORDING
```

Please take into consideration that the response is fixed to at most 1000 results (without any possibility to alter it).

This implies that responses can be rather large for vague terms. It also is not suited to walk through all the documents of a repository. It is best used with precise search terms.

Search by keywording

Search all documents of category ABC regarding contract XYZ.

```
POST  /api/search/keywording
```

```
{
    "CATEGORY":"ABC",
    "CONTRACT_ID":"XYZ"
}
```

Like the previous search, it will return the list of found documents and folders.

Note that you currently cannot specify the mask you are looking for. This feature might come in a future version. Also, in ELO, only "index fields" can be searched, not "map" fields.

Files

Get a folder

Gets all information about `/Administration/ELOapps/Icons`.

The following requests are equivalent:

```
GET  /api/files?path=/Administration/ELOapps/Icons
GET  /api/files/86
GET  /api/files/(8615AC76-DBE4-4907-03E5-F974F6C9F13A)
```

The response contains all information about the folder:


```
{
  "info": {
    "guid": "(8615AC76-DBE4-4907-03E5-F974F6C9F13A)",
    "id": 86,
    "name": "Icons",
    "type": 3,
    "isDir": true,
    "desc": "",
    "lock": "",
    "ownerName": "Administrator",
    "access": "RWD-LP",
    "parentId": 73,
    "dateArchived": "2020-10-26T10:09:00Z",
    "dateCustom": null,
    "dateModified": "2020-10-26T09:09:55Z"
  },
  "acl": [...],
  "children": [...],
  "keywording": {...},
  "versions": null,
  "content": null
}
```


Get a document

Gets all information about `/Administration/ELOapps/Icons/tile-Add.ico`.

The following requests are equivalent:

```
GET  /api/files?path=/Administration/ELOapps/Icons/
GET  /api/files/126
GET  /api/files/(7E2739B6-3EE5-8296-7C35-C07500621A8A)
```

The response contains all information about the document:


```
{
  "info": {
    "guid": "(7E2739B6-3EE5-8296-7C35-C07500621A8A)",
    "id": 126,
    "name": "tile-Add.ico",
    "type": 284,
    "isDir": false,
    "desc": "",
    "lock": "",
    "ownerName": "ELO Service",
    "access": "RWDE-P",
    "parentId": 86,
    "dateArchived": "2020-10-26T10:09:00Z",
    "dateCustom": null,
    "dateModified": "2020-10-26T09:09:57Z"
  },
  "acl": [...],
  "children": [],
  "keywording": {...},
  "versions": [...],
  "content": {...}
}
```


Performance

In both previous examples, all the information about the document/directory is retrieved.

For folders, this includes the list of all child files, and for documents, this includes the list of versions and basic information about the content.

Naturally, such an operation is more resource consuming than retrieving only what is actually needed.

Instead of retrieving everything using:

```
GET  /api/files/{id}
```

It is also possible to only get the subset of interest:

```
GET  /api/files/{id}/acl
GET  /api/files/{id}/children
GET  /api/files/{id}/content (*)
GET  /api/files/{id}/info
GET  /api/files/{id}/keywording
GET  /api/files/{id}/versions
```

This reduces traffic and is more efficient. Like the other endpoinds, both `id` and `guid` can be used interchangeably.

> **Information**
>
> (*) Calling `GET /api/files/{id}/content` actually downloaded the document instead of providing content metadata in the ELO 12 Beta Version. It was deprecated in REST API v20.02 and might be replaced by the content metadata in ELO 21.

Downloading a document

This example downloads the "tile-Add.ico" icon previously listed.

The following requests are equivalent:

```
GET  /api/files/126/download
GET  /api/files/(7E2739B6-3EE5-8296-7C35-C07500621A8A)/download
```

Response:

```
<binary-data> of the tile-Add.ico file
```

> **Information**
>
> In REST API prior to v20.02, `GET /api/files/{id}/content` should be used to download the document.

Listing files in a folder

This example lists the files in the "/Administration/ELOapps/Icons" folder previously obtained.

The following requests are equivalent:

```
GET  /api/files/86/children
GET  /api/files/(8615AC76-DBE4-4907-03E5-F974F6C9F13A)/children
```

Response:


```
    [
        {
          "guid": "(7E2739B6-3EE5-8296-7C35-C07500621A8A)",
          "id": 126,
          "name": "tile-Add.ico",
          "type": 284,
          "isDir": false,
          "desc": "",
          "lock": "",
```

```
        "ownerName": "ELO Service",
        "access": "RWDE-P"
      },
      ...
      {
        "guid": "(DE239939-F00C-69C2-9AF0-89C7E72D3519)",
        "id": 238,
        "name": "tile-WorldMap.ico",
        "type": 284,
        "isDir": false,
        "desc": "",
        "lock": "",
        "ownerName": "ELO Service",
        "access": "RWDE-P"
      }
  ]
```

Creating a folder

Request:

```
POST  /api/files
```

```
  {
      "info": {
          "name": "Example Folder",
          "parentId": 123,
      }
  }
```

Response:

```
  {
      "id": 1234,
      "guid":"(...)",
      "name":"Example Folder"
  }
```

Uploading a new document

To create/upload a new document using Json, the content has to be encoded as base64 as follows.

```
POST  /api/files
```

```
{
    "content": {
        "filename": "MyImage.png",
        "contentType" : "image/png",
        "data": "document-raw-content-as-base64-encoded"
    },
    "info": {
        "name": "MyImage",
        "parentId": 123
    }
}
```

This way, the keywording metadata can also be set directly. Note however that ACLs are read-only and inherit the rights of the parent folder by default.

Uploading a new document using an URL

The following example will upload the page `https://www.wikipedia.org` in ELO

Request:

```
POST  /api/files
```

```
{
    "content": {
        "filename": "index.html",
        "url": "https://www.wikipedia.org"
    },
    "info": {
        "name": "Wikipedia Homepage",
        "parentId": 123,
    }
}
```

Response:

```
{
    "id": 1234,
    "guid":"(...)",
    "name":"Wikipedia Homepage"
}
```

> **Information**
>
> - the filename extension determines the document type and icon in ELO
> - do not provide an `id` nor `guid` since it is a new file
> - many properties are read-only and setting them has no effect
> -

> the `guid` and `name` in the response are only in version 20+

Uploading a new document in the browser

This example shows a HTML snippet able to upload a document from the browser.

The following example assumes that the user is already authenticated.

Please consult the chapter Authentication for further information on how to achieve this in browser use cases.

```
<form id="sampleUpload">
    <input type="text" name="name">
    <input type="file" name="file">
    <input type="text" name="versionComment">
    <input type="submit">
</form>

<script>
    formElem.onsubmit = async (e) => {
        e.preventDefault();
        var parentId = 123;
        let response = await fetch('.../api/files/' + parentId, {
            method: 'POST',
            body: new FormData(formElem)
        });
        let result = await response.json();
        alert("Uploaded document id: " + result.id);
    };
</script>
```

Uploading a new document *version*

Uploading a new version of an existing document can only be done using the `POST /api/files/{id-or-guid}/content` endpoint and is similar to the previous examples.

Moving a file

This example moves the file `126` to the folder `/Administration` (having the id `2`)

Moving a file to another folder is actually the same as updating its `parentId`.

Request:

```
PATCH  /api/files/126/info
```

```
{
    "parentId": 2
}
```

In this case, the `parentId` should be the `id` of the target parent folder and a `guid` is not allowed.

Updating the short description

Request:

```
PATCH  /api/files/126/info
```

```
{
    "desc": "Here is an updated short description!"
}
```

Here as well, the `guid` could be used instead.

Getting or updating metadata

See Metadata

Metadata

Reading metadata/keywording

Request:

```
GET  /api/files/{id-or-guid}/keywording
```

Response:

```
    {
        "maskId": 2,
        "maskNameOriginal": "E-mail",
        "fields": {
            "ELOOUTL1": "example@noreply",
            "ELOOUTL2": "something else...",
            ...
        },
        "map": {
            "FOO": "BAR"
        }
    }
```

Note that the metadata is always key/value strings. Even if the field is defined as number, date or something else, the value is always handled as a string.

Updating metadata/keywording

Request:

```
PATCH  /api/files/{id-or-guid}/keywording
```

```
    {
        "fields": {
            "ELOOUTL1":"pseudomail@noreply"
        },
        "map": {
            "FOO":"BAR"
        }
    }
```

This will only update the provided fields, and leave the others unchanged.

Note that metadata are always key/value strings. Even if the field is defined as a number, date or something else, a string should always be provided.

Change mask

Request:

```
PATCH  /api/files/{id-or-guid}/keywording
```

```
{
    "maskNameOriginal": "E-Mail"
}
```

or

```
PATCH  /api/files/{id-or-guid}/keywording
```

```
{
    "maskId": 2
}
```

Change mask and metadata/keywording

Both operations can be combined into one.

Request:

```
PATCH  /api/files/{id-or-guid}/keywording
```

```
{
    "maskNameOriginal": "E-Mail",
    "fields": {
        "ELOOUTL1":"pseudomail@noreply"
    },
    "map": {
        "FOO":"BAR"
    }
}
```

Note that setting `maskId` or `maskNameOriginal` will not only update the mask (if necessary) but it will also *reset all other index fields*. Therefore, if you want to just update a few fields without affecting the others, omit `maskId` and `maskNameOriginal` in the request.

Members

List members of a group

The following requests are equivalent:

```
GET  /api/members/groups/Administrators
GET  /api/members/groups/(E10E1000-E100-E100-E100-E10E10E10E43)
GET  /api/members/groups/9998
```

Response:


```
    {
        "guid": "(E10E1000-E100-E100-E100-E10E10E10E43)",
        "id": 9998,
        "name": "Administrators",
        "parentGroups": [],
        "childGroups": [],
        "users": [
            {
                "guid": "(E10E1000-E100-E100-E100-E10E10E10E40)",
                "id": 0,
                "name": "Administrator"
            },
            ...
        ],
        "rights": null
    }
```


Get user

The following requests are equivalent:

```
GET  /api/members/users/Alice
GET  /api/members/users/(4357EA87-9744-B3A5-6911-4414A5160288)
GET  /api/members/users/5
```

Response:


```
    {
        "guid": "(4357EA87-9744-B3A5-6911-4414A5160288)",
        "id": 5,
        "name": "Alice",
        "password": null,
        "change_password": null,
        "email": "alice@wonder.lands",
```

```
        "nologin": false,
        "online": false,
        "memberOf": [
            {
            "guid": "(E10E1000-E100-E100-E100-E10E10E10E43)",
            "id": 9998,
            "name": "Administrators"
            },
           ...
        ]
    }
```

Add new user

Request:

```
POST  /api/members/users
```

```
    {
        "name": "Alice",
        "password": "secret",
        "change_password": "MUST_CHANGE",
        "email": "alice@wonder.lands",
        "memberOf": [
            {
                "id": 9998,
                "name": "Administrators"
            }
        ]
    }
```

Response:

```
{
    "guid": "(4357EA87-9744-B3A5-6911-4414A5160288)",
    "id": 5,
    "name": "Alice"
}
```

Note that you can directly provide the groups the user should belong to. In order to do this, you must provide the groups `id` to identify the groups in `memberOf`. Both `guid` and `name` are ignored in this case.

You cannot set individual rights/permissions. It is good practice that users inherit rights from the groups they are member of.

Set user's group memberships

The following request for Alice sets "Teamroom Users" as the only group she is a member of.

Request:

```
PATCH  /api/members/users
```


```
    {
        "guid": "(4357EA87-9744-B3A5-6911-4414A5160288)",
        "id": 5,
        "name": "Alice",
        "memberOf": [
            {
                "guid": "(C0553F55-AEF0-0C9C-5166-78F28047D0FD)",
                "id": 2,
                "name": "Teamroom Creators"
            }
        ]
    }
```


Either the `id` or `guid` of the user is required for this request.

Similar to when adding a user, you must provide the groups `id` to identify the groups in `memberOf`. Both `guid` and `name` are ignored in this case.

Misc

Calling a registered function (IX)

This request will call the registered function `RF_sol_function_ChangeColor` to change the color of the ELO folder *Administration* to red.

Request:

```
POST  /api/misc/functions/RF_sol_function_ChangeColor
```

```
{
    "objId": 2,
    "color": "red"
}
```

Please refer to the business solution documentation for more information about the specific remote functions available depending on your system and installed modules.

Masks (technical name for metadata forms)

Get mask information

The following requests are equivalent:

```
GET  /api/system/masks/(E10E1000-E100-E100-E100-E10E10E10E32)
GET  /api/system/masks/2
GET  /api/system/masks/E-mail
```

Response:


```
    {
    "id": 2,
    "guid": "(E10E1000-E100-E100-E100-E10E10E10E32)",
    "originalName": "E-mail",
    "forDocs": true,
    "forFolders": false,
    "forSearch": true,
    "isUsed": true,
    "fields": [
        {
        "id": 0,
        "key": "ELOOUTL1",
        "type": 3000,
        "label": "From",
        "isRequired": false,
        "isHidden": false
        },
        ...
        {
        "id": 6,
        "key": "ELOOUTLREF",
        "type": 3000,
        "label": "Reference",
        "isRequired": false,
        "isHidden": false
        }
      ]
    }
```


The `type` is a internal constant referring to the field's type (text, number, date...).

Note that even if the field is defined as a number, date or something else, it is nevertheless always handled as a string (of max length 255).

Installation procedure for ELO 12

> These instructions are solely for ELO 12!
>
> Starting from ELO 20, the REST API can be installed using the Server Setup.
>
> Since this is manually installed, it should also be manually removed before updating to ELO 20.

Install the "war" web app

1. Stop Tomcat
2. Copy the `rest-...war` to `<elo-path>/prog/webapps/rest.war`
3. Create the directory `<elo-path>/config/rest-<repo-name>/<app-server>`
4. Copy the `example-config/logback.xml` to the created directory and edit its content accordingly
5. Copy the `example-config/config.properties` to the created directory and edit its content accordingly
6. Copy the `example-config/rest.xml` to `<elo-path>/servers/<app-server>/conf/Catalina/localhost` and edit its content accordingly
7. Start Tomcat

Not working? Then something is probably misconfigured. Check all paths and configurations again, then reload the webapp.

Configuring the IX Proxy

This is required in order to share sessions with the Web Client or using web apps behind the IX proxy plugin.

- Edit `<elo-path>/config/ix-<repo-name>/de.elo.ix.plugin.proxy.properties`
- Add a line similar to `/rest/=http\://localhost\:9090/rest-<repo-name>/`
- Restart the IX

Example config.properties

```
# The IX url
ixUrl = http(s)://<server>:<port>/ix-<repo-name>/ix

# The default connection language
lang = de

# Allowed origins for Browser Cross Site Requests
allowedOrigins = http://localhost:8080, http://localhost:9090

# Allowing this allows using the '/api/login' to obtain durable web sessions
# Due to security considerations, this setting will be disabled if 'allowedOrigins' contains
allowSessions = true
```

Example logback.xml

Path:

```
<elo-path>/config/rest-<repo-name>/<app-server>/logback.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>


  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>***elo-path***/logs/***app-server***/rest-***repo-name***.log</file>

    <encoder>
    <charset>UTF-8</charset>
    <pattern>%d{HH:mm:ss.SSS} %-5level %-60(%thread %X{NDC} \(%logger{0}.java:%L\)) - %msg%n<
    </encoder>

  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <fileNamePattern>***elo-path***/logs/***app-server***/rest-***repo-name***.%d{yyyy-MM-dd}
    <!-- keep 30 days' worth of history -->
    <maxHistory>30</maxHistory>
  </rollingPolicy>

  </appender>



  <root level="info">
    <appender-ref ref="FILE" />
  </root>

</configuration>
```

Example rest.xml

Path:

```
<elo-path>/servers/<app-server>/conf/Catalina/localhost/rest.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Context docBase="***elo-path***\prog\webapps\rest.war" path="/rest-***repo-name***" unpackWA
  <Environment name="webappconfigdir" override="false" type="java.lang.String" value="***elo-
  <Environment name="logging.config"  override="true"  type="java.lang.String" value="***elo-
</Context>
```