

enaio[®]

Software Documentation
enaio[®] appconnector

Version 8.50



All software products as well as all related extension programs and additional functions are registered and/or in-use trademarks of OPTIMAL SYSTEMS GmbH, Berlin or its subsidiaries. They may only be used according to a valid licensing agreement. The software as well as related documentation are protected by German and international copyright law. Unauthorized duplication and sales is plagiarism and subject to criminal prosecution. All rights reserved, including reproduction, transmission, translation, and storage with/on all kinds of media. For all preconfigured test scenarios or demo presentations: All company and person names which occur in examples (screenshots) are fictional. Any resemblance to existing companies or persons is purely coincidental and unintentional.

Copyright 1992 – 2017 by OPTIMAL SYSTEMS GmbH
Cicerostraße 26
D-10709 Berlin

07.04.2017
Version 8.50

Contents

Contents	3
Introduction.....	4
enaio® appconnector	5
About enaio® appconnector	5
IT Security	5
System Requirements	5
Installation	6
Installing a Hotfix or Patch.....	6
Core Service Update.....	7
Configuration	7
Configuring the JVM	7
The enaio® appconnector Configuration File	8
enaio® client	12
Capabilities of enaio® appconnector	12
enaio® apps.....	15
Script Language	18
DropTargets	19
Configuration.....	20
Testing DropTargets	42
Push Notification Service for enaio® apps	42
Configuration.....	42
Attachment.....	44
Integrating enaio® appconnector with .NET	44
Integration Possibilities.....	44
Authentication.....	46
Handling JSON Responses Given by enaio® appconnector	47
API Documentation	49
General.....	49
Authentication.....	49
Services.....	49
Results.....	110

Introduction

This manual is available as a PDF file. The PDF file is installed in the documentation directory. Adobe Reader can be used to read the document on-screen, to quickly search for particular terms, or to completely or partially print it.

This manual describes how to install and configure the enaio® appconnector interface, which must be installed in order to use enaio app® and perform user-oriented queries from within Web applications, for example.

How to use enaio® appconnector as the Details Preview in enaio® client can be found in the Administration handbook. The user guide for enaio® appconnector as a Details Preview can be found in the enaio® client handbook.

enaio® appconnector

About enaio® appconnector

enaio® appconnector is an enaio® core service.

The core services are default components of enaio® and are required for operating the enaio® platform and a proper functioning of the individual enaio® components.

As it is a REST interface (Representation State Transfer) enaio® appconnector provides resource-oriented, flexible HTTP access to index and document data in enaio®.

Thus, enaio® appconnector offers several application areas:

- § It serves as an interface, for example, to mobile applications, such as enaio® app for Android-based smartphones, iPhones, and tablets.
- § It can be deployed as the Details Preview for displaying index data and other data in enaio® client and other external applications.

The OSRest API documentation is included in the Attachment.

IT Security

Mobile applications are designed to be used on-the-go and are running on small, portable smartphones or tablets. However, in a moment of carelessness, these devices can easily be stolen by another person. Furthermore, apps that have been installed unconsciously or by a third party can pass unnoticed and read out information, posing a significant risk to the owner.

Thus, mobile applications must ensure the security of sensitive data even in case of theft. Detailed information on IT security concerning the use of enaio® appconnector and enaio® app can be found in the system manual, in the 'Security' chapter.

System Requirements

enaio® appconnector is provided together with an Apache Tomcat version 6.

Due to security reasons, it is recommended to run the application server in a DMZ with an upstream web server (e.g. apache HTTPD Server).

For security reasons, the use of SSL encryption is recommended.

This can either be transferred by an Apache Tomcat Server or an upstream web server. It is additionally recommended to configure compression of the contents with Gzip (e.g. with the Apache module `mod_deflate`).

The following system requirements must be met in order to install and run enaio® appconnector:

- § The 'APP' license has to be registered in enaio® server. The 'MOB' license must be registered for enaio® app.
- § The system role 'DMS: Supervisor' is required for the configuration.

Installation

The enaio® appconnector REST interface is installed as a service through the enaio® setup. To do so, select the enaio® appconnector core service in the setup.

The runtime environment (JDK and application server) is also automatically installed.

The installed runtime environment should be used only for this core service, because when updating the core service the runtime environment is updated as well. If other enaio® or third-party components are run in the runtime environment, update errors may occur or the components may not be run after an update anymore.

The following data must be entered during the installation process:

- § Server name and port
- § Name and password of the technical user

The service is automatically registered with enaio® server during the setup.

The 'Server: Switch job context' system role must be assigned to the technical user whose user account is used to execute enaio® appconnector.

The installation of enaio® appconnector is then complete, and you can start the 'osrest' application in the Web Application Manager.

enaio® appconnector registers its service endpoint with enaio® server during installation, so that it can be read by other components. You can view and change the service endpoint in enaio® enterprise-manager under **Server properties > Category: Services > 'Core Service' > Service endpoint**.

Installing a Hotfix or Patch

When installing a hotfix or patch, only those files that differ from the current version are replaced. Updating your installation to a newer version is not possible using a hotfix or a patch.

A hotfix or patch replaces only a few system files, so the enaio® appconnector installation does not need to be reconfigured.

A hotfix does not back up the existing enaio® appconnector installation.

Before replacing files, it is checked whether adequate file versions are available at hotfix or patch installation. If this is not the case or a newer hotfix or patch was

already installed, no files will be replaced. The hotfix and patch installations will be canceled reporting a message that the version of the installed service is wrong.

Hotfixes (`osappconnector_hotfix.exe`) are located in the `SP` directory of the installation data.

Patches (`osappconnector_patch.exe`) can be downloaded from the OPTIMAL SYSTEMS [partner portal](#), the service portal for partners and customers of the OPTIMAL SYSTEMS group.

Note that, as soon as an installation has been patched, the `SP` directory will not be read out during future updates anymore. Supplement changes can then be installed only by patch. A patch installation cannot be undone.

Core Service Update

A core service update can be performed from the enaio® setup.

A backup of the current configuration file versions is performed automatically. After an update, the backed up configuration files can be found in the subdirectory `backup-(timestamp)` of the core service.

Configuration

A configuration file is available for configuring enaio® appconnector.

If enaio® appconnector was installed manually and not via the enaio® setup, you will need to enter the Home URL of enaio® appconnector in the **Server properties** > **Category: Services** > **AppConnector** area of enaio® enterprise-manager.

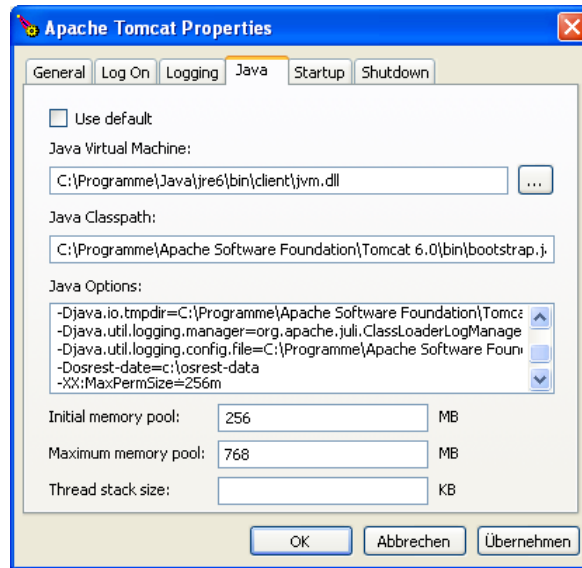
Example: `http://localhost:8060/osrest`

Before editing the configuration file, you also have to configure the JVM (see 'Configuring the JVM').

Configuring the JVM

As enaio® appconnector uses an external directory for its configuration data, you must create a directory outside of the Apache Tomcat directories, and copy the contents of the `...\\services\\os_appconnector\\osappconnector-<Version>.zip\\osappconnector\\configuration` installation data directory to this new location.

Then open the Apache Tomcat properties dialog and enter the directory outside of the Tomcat installation, e.g. `Dosrest-data=c:/osrest-data` on the **Java** tab in the `osrest-data` system property.



enaio® appconnector requires a minimum JVM heap size (**initial memory pool**) of 256 MB and a maximum size (**maximum memory pool**) of 768 MB. These values are considered initial sizes, and it is recommended to adapt them to the specific conditions depending on whether other applications are run on the application server or whether very large documents have to be processed. It is recommended to set the storage for JVM administration (permanent generation) to 256 byte ('-XX:MaxPermSize=256m').

The enaio® appconnector Configuration File

To configure enaio® appconnector, edit the `osrest.properties` configuration file in the application directory ... \services\OS_Appconnector\configuration.

The 'Server: Switch job context' system role must be assigned to the technical user whose user account is used to execute enaio® appconnector.

The following settings can be configured in the configuration file:

Server Connection

`connection.string`

Connection data to the server, following the pattern `[Name|IP]:[Port]:[Weighting]`. Several servers are separated by the # character. The server's weighting indicates the possibility (in percent) that a connection to the server is established. If you use one server only, enter the value '100'.

Example:

`localhost:4000:50#127.0.0.1:4600:50`

`technical.user.name`

Name of the technical user

`technical.user.password`

Password of the technical user

Note that the password entered here is not encrypted. You can also enter the encrypted

password including # characters from the user table of the enaio® data base.

General Configuration

<code>default.pagesize</code>	Maximum number of hits provided by enaio® appconnector. The default value is 500 hits.
<code>favouritesPortfolioName</code>	For versions earlier than 7.10, a portfolio containing the favorites can be specified. Specify the topic.
<code>notifications.skipworkflow</code>	Send notifications about incoming workflows Users will not be notified if the parameter is set to 'true'.
<code>metadata.defaultmappingname</code>	Name of the configuration file that defines which index fields of the various enaio® object types are displayed, e.g. in enaio® app and enaio® sync. The file is in the ...\\configuration\\schema directory. The file can be found in the ...\\configuration\\schema directory.
<code>osecm.default.nameschema</code>	Specify whether internal names in the object definition must be used for index data assignment. Default value is <code>internal_name</code> . You can also use the names and index data fields of DMS objects (name) for assignment, but the names must then follow the same rules as internal names (see 'Script Language').
<code>admin.email</code>	E-mail address to which app users can send error messages. In addition to the error message, the e-mail also contains the stack trace to the error.
<code>welcomepage</code>	URL of a welcome screen suitable for display on mobile devices The welcome screen will be shown when starting enaio® app and when changing the server profile, provided that the user has enabled the app configuration parameter Always show welcome screen . Please note that the colon in the URL must be preceded by an escape character. Example: <code>http\\://www.ecm.mobi</code>

Connection to Core Services

<code>base.url</code>	Basic URL of enaio® appconnector
-----------------------	----------------------------------

```
fileservice.  
contentviewerurl
```

The URL will be read automatically from the server registry and entered here. If you enter a URL yourself, the registry value will be ignored.

Base URL to enaio® contentviewer

It will be read out automatically from the server registry and entered here. If you enter a URL yourself, the registry value will be ignored.

```
fileservice.  
documentviewerurl
```

Base URL to enaio® documentviewer

enaio® documentviewer creates previews of enaio® objects, which can be viewed with enaio® app, for example. The URL will be read automatically from the server registry and entered here. If you enter a URL yourself, the registry value will be ignored.

Example: `https://demo.optimal-systems.org/osdocumentviewer`

```
services.  
renditioncache
```

URL of the rendition service

The URL specified here will overwrite the corresponding value in enaio® enterprise-manager.

```
services.  
detailsvviewer
```

URL of enaio® detailsvviewer

The URL specified here will overwrite the corresponding value in enaio® enterprise-manager.

```
fileservice.  
osweburl
```

Base URL to enaio® webclient

This URL is required to edit the content of DMS objects in enaio® web-client on your mobile device, for example. These can otherwise only be viewed in enaio® app.

Example: `https://demo.optimal-systems.org/osweb`

```
extractionservice.url
```

Base URL to enaio® extraction

If you would like to use these components (extraction of EXIF files from audio, video, and image files, XMP data from Office and PDF documents, and default properties from e-mails in MSG and EML formats), contact the OPTIMAL SYSTEMS Professional Services team.

Authentication

<code>authentication.usebasic</code>	If it should be possible to log in through basic authentication, 'true' must be entered.
<code>authentication.usebasic. windowsauth</code>	If it should be possible to log in through Windows authentication, 'true' must be entered.
<code>authentication. defaultdomain</code>	Default Windows domain for basic authentication
<code>authentication. usentlm</code>	The NTLM authentication can be activated if necessary ('true'). However, make sure that the authentication of all core services is applied by default by enaio® gateway.
<code>authentication. useprofileuser</code>	<p>A profile user can be used ('true') when the service needs to be addressed without authentication.</p> <p>The parameters <code>profile.user.password</code> and <code>profile.user.name</code> are then used to specify the name and password of the profile user.</p>
<code>profile.user.name</code>	<p>Name of the profile user</p> <p>This information is only required if the <code>authentication.useprofileuser</code> parameter was set to 'true.'</p>
<code>profile.user.password</code>	<p>Password of the profile user</p> <p>This information is only required if the <code>authentication.useprofileuser</code> parameter was set to 'true.'</p> <p>Note that the password entered here is not encrypted. You can also enter the encrypted password including # characters from the user table of the enaio® data base.</p>
<code>authentication. restrictaccesstogroups</code>	<p>OS groups that will receive access to the service</p> <p>Use the comma to separate multiple groups. User must always be a member of all groups specified here in order to have access to the service.</p>

Pushnotification

<code>services.pushnotification. enabled</code>	This must be set to 'true' for the system to send push notifications to mobile devices.
<code>services.pushnotification. production</code>	Internal parameter that cannot be changed. Changes will prevent the push notification service from working.

<code>services.pushnotification.version</code>	Internal parameter that cannot be changed. Changes will prevent the push notification service from working.
<code>services.pushnotification.proxy.enabled</code>	If a proxy is used to connect to the Internet, this must be set to 'true.'
<code>services.pushnotification.proxy.address</code>	If a proxy is used to connect to the Internet, its address must be specified here.
<code>services.pushnotification.proxy.port</code>	Port of the proxy server
<code>services.pushnotification.proxy.password</code>	Password of the proxy user
<code>services.pushnotification.proxy.username</code>	Name of the proxy user
<code>res.revision</code>	Internal parameter that cannot be changed.

Settings for the push notification service are described below ('Push Notification Service for enaio® apps').

The core service must be restarted for the changes made to the configuration file to take effect.

enaio® appconnector logs all actions to the `osrest.log` file in the application directory `...\services\OS_Appconnector\configuration\logs`.

enaio® client

Profile pictures for notes in the details preview

The details preview in enaio® client makes it possible to directly input text notes next to which the author's profile picture is shown.

Profile pictures will only be displayed if JPG images for users (of size 64x64 pixels) are present in the application directory

`...\services\OS_Appconnector\configuration\avatar`. The names of the pictures must match the enaio® user names, for example `root.jpg` or `demo.jpg`.

Capabilities of enaio® appconnector

The capabilities of enaio® appconnector can be configured in the `osrest.caps.xml` configuration file, located in the application directory `...\services\OS_Appconnector\configuration`.

The capabilities of an installation depend on several aspects, including the version of enaio® appconnector, the enaio® licenses, administrative restrictions, and the system roles of the enaio® user.

Capabilities can be enabled or disabled (true/false) or have values. Capabilities apply to Android, iOS, and Windows devices, unless values for the respective

platform are specified in the configuration. You can control this by including the `ua` attribute in the `defaultto` element of the corresponding capability.

Example:

```
<oscap name="PlatformNotifications" source="POL">
  <defaultto ua="ios" value="true"/>
  <defaultto ua="android" value="false"/>
  <defaultto ua="other" value="false"/>
</oscap>
```

The following settings can be configured here:

Category	Capability name	Function	Default
Version-dependent capabilities			
VER	DateRanges	Time intervals can be specified via 'from' and 'to' dates.	true
VER	MarkInboxItemsAsRead	All elements will be displayed differently depending on whether or not they have been read.	true
License-dependent capabilities			
LIC	Capture	The camera entry or photo library entry is displayed in the DropTarget list.	true
LIC	Fulltext	The full text search is displayed in the queries list.	true
LIC	Import	The capture area is enabled or disabled.	true
LIC	Inbound	The 'Open in mobileDMS' feature is available for external apps.	true
LIC/POL	Offline	Documents are cached and favorites are available offline. If disabled, all documents are removed from the app storage after closing the document viewer.	true
LIC/POL	Outbound	The 'Open document' feature is available in the action menu of the document viewer.	true

LIC	workflow	Indicates if workflows are displayed on the inbox tab.	true
Policy-dependent capabilities			
POL	PlattformNotifications	Indicates if the server profile option 'Push notifications' can be enabled.	true
POL	AllowDetailsView	The document viewer is available. If disabled, only index data are displayed.	false
POL	AllowInbox	Displays the 'Inbox' section.	true
POL	AllowQueries	Provides the queries tab.	true
POL	AllowUsertray	Provides the entry 'User tray' in the queries list.	true
POL	CacheTime	Indicates how long documents are kept in the app storage and are available offline. An integer value for the number of minutes must be entered here, e.g. 10080 for 7 days.	10080
POL	ForceClientSSL	The client will check whether encryption is used for all data exchanges.	
POL	ForcePinLock	Specifies whether the app must be secured using a PIN lock. If 'true,' users can only connect to the server if a PIN lock has been set up.	false
POL	ForceSSLTrust	Specifies whether users can enable the server profile option 'Trust server certificate,' letting them establish a connection to enaio® appconnector even with invalid SSL server certificates.	false
POL	ForceStartView	With this capability, the start tab can be specified. The values are: inbox, queries, favorites, outbox.	-

POL	ForceWelcome	Specifies if the welcome page is displayed at every start. If 'true', this option cannot be disabled in the settings.	false
POL	MaxHierarchyDepth	Hierarchy depth of cached favorites objects	10
POL	AllowFavorites	Displays the 'Favorites' section	true
POL	AutoReloadInbox	Specifies the interval (in minutes) at which the inbox is reloaded. Permitted values: 0 (= disabled), 1, 5, 10, 30, 60	0
POL	AllowLocation	Specifies whether an object's location can be identified.	true
Other capabilities			
CFG	UrlRenService	URL of the enaio® rendition service	
CFG	WelcomeURL	A server's welcome page is displayed every time a new connection to this server is established.	true

enaio® apps

Showing Index Data

In order to display enaio® objects on mobile devices with enaio® apps, you will need to define which index data fields of the individual object types will be visible in the app. To do so, assign enaio® index data fields to the fields in the app. These can be assigned in the `OSMetadata.xml` configuration file.

Without the assignment, DMS objects are listed in the app but they cannot be identified due to unfilled fields.

These assignments will also be used for enaio® sync.

The `OSMetadata.xml` file is installed by the setup and can be found in the `configuration\schema` directory. As the `OSMetadata.xml` configuration file contains sample data by default, you will need to adjust the file.

The sample configuration file is structured as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<archive>
  <model>
    <property name="title"/>
```

```

        <property name="info"/>
    </model>
    <cabinet name="Customer">
        <object name="Miscellaneous">
            <property name="title" field="Type"/>
            <property name="info" field="Short text"/>
        </object>
        <object name="Supportcall">
            <property name="title" field="Call_Nr + ' ' + Status"/>
            <property name="info" field="Problem"/>
        </object>
        <object name="Sales">
            <property name="title" field="Type"/>
            <property name="info" field="Short text"/>
        </object>
        <object name="Customer">
            <property name="title" field="Company name"/>
            <property name="info" field="Street + ' ' + ZIP + ' ' +
Location"/>
        </object>
        <object name="Email">
            <property name="title" field="'E-mail from: ' +
MAIL_FROM"/>
            <property name="info" field="MAIL_SUBJECT"/>
        </object>
        <object name="Person">
            <property name="title" field="FirstName + ' ' +
LastName"/>
            <property name="info" field="Classification"/>
        </object>
        <object name="Incoming invoice">
            <property name="title" field="'Invoice ' +
InvoiceNumber"/>
            <property name="info" field="'From ' + InvoiceDate + ',
' + Status"/>
        </object>
        <object name="Document">
            <property name="title" field="Type"/>
            <property name="info" field="empty(Description) ?
'empty' : Description"/>
        </object>
        <object name="History">
            <property name="title" field="Type"/>
            <property name="info" field="Topic"/>
        </object>
    </cabinet>
    <typeless>
        <property name="title" field="if (OBJECT_MAIN == 3) { y=
'Image'; } else { y = 'File'; y }"/>
        <property name="info" field="'Created on: ' +
OBJECT_CRDATE"/>
    </typeless>
    <notification>
        <workflow>
            <property name="title" field="ActivityName"/>
            <property name="info" field="ActivitySubject"/>
        </workflow>
    </notification>
</archive>

```

The fields `title` and `info` are defined in the `<model>` area and displayed in enaio® apps, and can be filled with the index data of enaio® object types.

The tag `<cabinet name="name of the cabinet">` is then entered for every cabinet in the configuration file. Within this tag, index data fields are mapped for the individual object types (`<object name="Name of the object">`). For this purpose, the name of the field is defined in the `<property>` tag in the `name` attribute, and the index data from enaio® are referenced in the `field` attribute. It is possible to freely configure the values of different index data fields using a special script language (see 'Script Language').

Typeless documents are defined by the `<typeless>` tag. In this tag, the main type of the object (`OBJECT_MAIN`) and the capture date (`OBJECT_CRDATE`) are assigned to app fields.



Please note that only internal object type names of the object definition can be used (see the 'enaio® editor' handbook) to map enaio® index data.

Filing Objects

The app can be used to create new objects at a location in enaio®. By default, users can select from all object types to which they also have access in enaio® client. Also by default, the index data forms only contain the mandatory fields of the object types.

Using the `OSMetadata.xml` configuration file, you can specify which object types should not be visible in the app, and which index data fields should be shown in addition to the mandatory fields.

To hide object types for the filing tray, insert the tag `<insert active>` and set it to 'false.' The default value of the tag is 'true.'

Example:

```
<object name="Sales">
  <property name="title" field="Type"/>
  <property name="info" field="Short text"/>
  <insert active="false" />
</object>
```

To display additional index data, insert the tag `<insert active>` for each object type and specify inside it all index data fields that should be shown in the app in addition to the mandatory fields when filing an object.

Example:

```
<object name="Miscellaneous">
  <property name="title" field="Type"/>
  <property name="info" field="Short text"/>
  <insert active="true">
    <property field="Type"/>
    <property field="Short text"/>
  </insert>
</object>
```

Script Language

To guarantee processing in the script environment, the internal names of the enaio® object types must be structured as follows: Names begin with the characters from a to z, from A to Z, or &. The characters from 0 to 9, from a to z, from A to Z or the characters _ or \$ can be used afterwards. The names of variables are case-sensitive.

The following names are reserved and cannot be used as variable names: `or`, `and`, `eq`, `ne`, `lt`, `gt`, `le`, `ge`, `div`, `mod`, `not`, `null`, `true`, `false`, and `new`.

To assign index data fields from enaio® to fields in enaio® apps, use the script language described below.

Language Elements

Statements Use a semicolon to terminate statements.

Blocks Blocks are different statements enclosed by braces.

Assignments Values can be assigned to variables using the equals sign:

```
var='value';
```

Literals

Integer One or more digits from 0 to 9.

Floating Point One or more digits from 0 to 9 followed by a decimal point and further digits from 0 to 9.

Boolean `true` or `false`

String Strings enclosed by single quotation marks:

```
'Hello world'
```

Functions

empty Returns `true` if the following expression is `null`.

An empty string, e.g. `empty(var1);`

size Returns the length of a string, e.g. `size('Hello');`

Operators

Logical operators **AND:**

```
cond1 and cond2
```

```
cond1 && cond2
```

OR:

```
cond1 or cond2
```

```
cond1 || cond2
```

NOT:

```
!cond1
```

```
not cond1
```

Conditional operators

The common operator `condition ? if_true : if_false` as well as the short form `value ?: if_false` can be used:

```
val1 ? val1 : val2
val1 ?: val2
```

Relational operators

e.g.

```
val1 == val2
val1 eq val2
val1 != val2
val1 ne val2
val1 < val2
val1 lt val2
val1 <= val2
val1 le val2
val1 >= val2
val1 ge val2
```

Regular expressions

e.g.

```
var1 =~ 'abc.*'
var1 !~ 'abc.*'
```

Calculations

Additions, subtractions, multiplications and divisions can be carried out:

```
val1 + val2
val1 - val2
val1 * val2
val1 / val2
val1 div val2
```

Condition

if e.g.

```
if ((x * 2) == 5) {
  y = 1;
} else {
  y = 2;
}
```

DropTargets

DropTargets can be used to import data and documents into the enaio® system using enaio® appconnector. With drop targets, various actions in the system can be executed, such as starting a workflow or creating or updating DMS objects.

Each droptarget describes exactly one import scenario.

Droptargets are written in Jelly which is a script language used in XML scripts.

Droptargets are, for example, implemented by script. The necessary API call is found in the appendix (see `'/osrest/api/documentfiles/droptargets/[targetname]'`).

Configuration

DropTargets must be saved to the

... \services\OS_Appconnector\configuration\droptargets application directory. The two demo DropTargets demo.xml and demo2.xml can already be found in this directory.

Each droptarget is defined in an XML file. The XML file must not contain umlauts or special characters unless they are UML encoded.

The XML file name serves as the droptarget name.

DropTarget Structure

The following sections describe the basic structure of droptargets and further elements.

General Structure

As usual for Jelly scripts, a droptarget starts with a definition of required namespaces.

You must then select a cabinet that actions are executed in. It is also possible to select multiple cabinets at once, which will then be processed in sequence.

Within a cabinet you can define actions. Creating server jobs that are consecutively executed is also possible. For each server job you must specify a DMS object.

The following example creates a cabinet with the internal name 'customer' with a folder for a customer named Peter Smith.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="customer">
    <os:insert>
      <os:folder>
        <os:field key="name">Mueller</os:field>
        <os:field key="firstname">Franz</os:field>
      </os:folder>
    </os:insert>
  </os:cabinet>
</j:jelly>
```

The key attribute is used to identify the cabinet by its internal name.

Using the keytype attribute, you can define whether the internal name (internal_name) or the name (name) of a cabinet must be specified in the DropTarget. By default, the internal name must be specified, so the keytype attribute is only necessary in the script if you would like to use the name of a DMS object instead of the internal name.

Then define:

```
<os:cabinet key="Customer" keyType="NAME" />
```

You can change the default behavior by setting the osecm.default.nameschema parameter in the osrest.properties enaio® appconnector configuration file from internal_name to name. However, names must then follow the same rules as

internal names (see 'Script Language' and 'The enaio® appconnector Configuration File').

Links

Server job results must often be further processed. To allow this, use the `id` and `ref` attributes. With `id` you can add a DMS object to the Jelly context, and with `ref` you can then refer to this DMS object in other script lines.

With the following example shows how to find and then delete a DMS object.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="customer">
    <os:search id="myfolder">
      <os:folder>
        <os:field key="name">Mueller</os:field>
        <os:field key="firstname">Franz</os:field>
      </os:folder>
    </os:search>
    <os:delete>
      <os:folder ref="myfolder"/>
    </os:delete>
  </os:cabinet>
</j:jelly>
```

Beneath the 'myfolder' identifier the search result is added to the Jelly context. The **<folder>** tag within the **<delete>** tag refers to the search result. As a result, the found folder is deleted. The **<search>** tag by default adds the first result to the Jelly context.

Server Jobs Processing Several DMS Objects

For some server jobs, more than one DMS object must be specified. The purpose of each DMS object must be defined explicitly. This is done via the `purpose` attribute.

The following example shows how to create a document using the **<insert>** tag. First, a DMS object with the `purpose="INSERT"` attribute – here a document with the internal name 'doc' – is defined as the DMS object to be created. The DMS object with the `purpose="LOCATION"` attribute then defines the location, here a folder with the OSID '5566'.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="customer">
    <os:insert file="mypdf">
      <os:document id="mynewdocument" key="doc"
purpose="INSERT"/>
      <os:folder osid="5566" purpose="LOCATION"/>
    </os:insert>
  </os:cabinet>
</j:jelly>
```

Most server jobs have a default purpose. This is assigned to the first DMS object for which the `purpose` attribute was not set. If, in the above example, the `purpose="INSERT"` attribute was not been specified for `os:document`, this attribute would be set automatically.

Access to Variables

Within a Jelly context different variables can be accessed. For example:

- § variables passed to the Jelly script
- § IDs and object type IDs of DMS objects
- § Attributes of DMS objects, i.e. their field values

These attributes are accessed using this notation:

```

${NAME_OF_PASSED_ATTRIBUTE}
${OBJECTNAME.FIELDNAME}
${OBJECTNAME['FIELDNAME']}

```

In line 1, a passed attribute is accessed. The lines 2 and 3 access the field of a DMS object, whereas the notation in line 3 is necessary only if the field name is empty and/or contains special characters.

Besides, there are reserved fields that enable access to specific object information:

```

${OBJECTNAME.key}
${OBJECTNAME.keyType}
${OBJECTNAME.osid}
${OBJECTNAME.objectTypeId}

```

Logging

Object information can be used to create meaningful log entries.

The following example shows how to create a support call and write a log entry with log level INFO.

```

<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="Customer" keyType="name">
    <os:insert>
      <os:register key="Supportcall" id="suppcall">
        <os:field key="Call-Nr">${myNumber}</os:field>
      </os:register>
      <os:folder osid="1494"/>
    </os:insert>
    <os:logger level="info">The support call ${myNumber} with
OSID ${supportcall.osid} was created.</os:logger>
  </os:cabinet>
</j:jelly>

```

It is recommended to put `<logger>` tags outside of server jobs. Otherwise, logs will be written before server jobs are completely processed.

Handling Several Hits

The hit number of a search can be defined using the search mode. By default, the first hit is added to the Jelly context. In addition, you can use the `mode="ALL"` attribute to get all hits.

```

${OBJECTNAME[0].FIELDNAME}
${OBJECTNAME.FIELDNAME}
${OBJECTNAME[n].FIELDNAME}

```

In line 1, the field of the first DMS object is addressed. The first DMS object can be addressed without specifying an index (line 2), as well. That way it is possible to

change the search mode without the need to adapt other script lines. Further DMS objects must always be addressed by the index (line 3).

Multi-Purpose Tags

Often used process flows have been collected in tags.

The `<select>` tag, for example, finds and updates a DMS object, or creates one if none is present.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="qsmanuals">
    <os:select id="folder">
      <os:folder purpose="SEARCH">
        <os:field key="name">User manual</os:field>
        <os:field key="author">John Doe</os:field>
      </os:folder>
      <os:folder purpose="INSERT, UPDATE">
        <os:field key="name">User manual</os:field>
        <os:field key="author">John Doe</os:field>
        <os:field key="visiblefor">Schmidt</os:field>
        <os:field key="editfor">Meier</os:field>
      </os:folder>
    </os:select>
  </os:cabinet>
</j:jelly>
```

It is possible to define multi-purpose objects. The example above shows a folder used for inserting and updating.

By default, the `<update>` and `<insert>` tags have their value set to 'true.' For this reason, even DMS objects must be defined for the `purpose="INSERT, UPDATE"` purpose.

Using Processors

Metadata can be read out and used in the droptarget with the help of file processors.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:email id="Mail" />
  <os:cabinet key="Emailarchive" keyType="name">
    <os:insert id="InsertedObject" >
      <os:register osid="14024315" purpose="LOCATION"/>
      <os:document key="Email" purpose="INSERT">
        <os:field key="MAIL_TO"
keyType="Internal_Name">${Mail.TO}</os:field>
        <os:field key="MAIL_FROM"
keyType="Internal_Name">${Mail.FROM}</os:field>
      </os:document>
    </os:insert>
  </os:cabinet>
</j:jelly>
```

The e-mail processor is defined in the second line. Files in supported file formats can be passed to the DropTarget, and their readable metadata can be referenced using the attribute `${<id>.<attribute>}`.

Processors are independent of cabinets and therefore do not need to be inside `<cabinet>` tags.

Formatting Dates

In order to format dates according to a desired pattern, Jelly provides individual JSP Standard Tag Library oriented means.

```
<j:jelly xmlns:j="jelly:core" xmlns:fmt="jelly:fmt"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <j:new var="currentDate" className="java.util.Date"/>
  <fmt:setLocale/>
  <fmt:formatDate pattern="yyyy" value="{currentDate}"
var="year"/>
</j:jelly>
```

In line 1, the `fmt` namespace is defined. A variable is then created, formatted as `yyyy`, and added to the context as `year`. You can access the current year within the Jelly script using `{year}`.

Converting the Date

enaio® works with the following date types: UNIX timestamp, German date, and German date with time. Applications that use the droptargets should always provide a date as a UNIX timestamp. As enaio® cannot carry out a conversion itself, this must be performed within the droptargets.

```
<j:jelly xmlns:j="jelly:core" xmlns:fmt="jelly:fmt"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <j:new var="currentDate" className="java.util.Date">
    <j:arg value="{(timestamp * 1000)}" type="long"/>
  </j:new>
  <fmt:setLocale/>
  <fmt:formatDate pattern="dd.MM.yyyy" value="{currentDate}"
var="date"/>
</j:jelly>
```

In lines 2-4, the `timestamp` variable that contains a UNIX timestamp is converted to a Java date object. In line 7, this object is converted to the `dd.MM.yyyy` format and saved to the `date` variable.

Evaluating date formats and regular expressions

Droptargets, date formats, and regular expressions can be evaluated and processed in order to assume data from third-party systems.

Example for date formats:

```
<j:jelly xmlns:j="jelly:core"
xmlns:utils="jelly:com.os.dtUtils.UtilsTagLibrary">
  <j:new var="date" className="java.util.Date"/>
  <utils:DateFormat id="formatFromDate" value="{date}"
pattern="dd. MMM yyyy"/>

  <utils:DateFormat id="formatFromText" value="GEB:12.12.2002"
inputPattern="dd.mm.yyyy" pattern="yyyy" parsePosition="4"/>
</j:jelly>
```

To do so, a date object or a date can be passed as a value (value attribute) when provided as a character sequence. Using a specified pattern (inputPattern attribute), a character sequence can be imported in accordance with SimpleDateFormat class rules. The time from which the pattern is to be used can be determined with the parsePosition attribute (counted becomes zero-based).

Example for regular expressions:

```
<j:jelly xmlns:j="jelly:core"
  xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary"
  xmlns:utils="jelly:com.os.dtUtils.UtilsTagLibrary">
  <utils:RegexFormat id="formattedText" value="GEB:12.12.2002"
    pattern="([0-9]{4})"/>

  <os:logger level="INFO">${formattedText[0]}</os:logger>
</j:jelly>
```

The value is written to the value attribute. The regular expression is specified in the pattern attribute.

To be able to access the results of the formatting, groups must be defined in the regular expression. The results can then be accessed using the respective group's number.

Add Several Table Rows

Jelly also allows you to add more than one table row within a DMS object. To do so, the `<forEach>` tag from the Jelly namespace is used.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="customer">
    <os:update>
      <os:folder id="myfolder">
        <os:table key="history">
          <j:forEach items="${hist}" var="entry">
            <os:row>
              <os:field key="date">${entry.date}</os:field>
              <os:field
key="username">${entry.user}</os:field>
              <os:field
key="state">${entry.state}</os:field>
              <os:field
key="priority">${entry.prio}</os:field>
            </os:row>
          </j:forEach>
        </os:table>
      </os:folder>
    </os:update>
  </os:cabinet>
</j:jelly>
```

In line 6, a loop is defined, iterating over a list which exists under the name `hist` in the Jelly context. Individual list entries are available under the name `entry`. Within the loop, the `<row>` tag must be used to define the individual columns of the table row.

The following example shows the variable structure in JSON.

```
{
  "hist": [
    {
      "date": "01.02.2001",
      "user": "Hans",
      "state": "Negotiation"
    },
    {
      "date": "12.12.2000",
      "user": "Petra",
```

```

        "state": "Complete",
        "prio": "high"
    }
]
}

```

The variable is a list containing key-value-pair (map) entries.

Creating a Reference Copy

A reference copy is created with the `<link>` tag.

```

<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="Emailarchive" keyType="name">
    <os:link>
      <os:document ref="email"/>
      <os:register ref="locationForTheReferenceCopy"
purpose="LOCATION"/>
    </os:link>
  </os:cabinet>
</j:jelly>

```

Creating a Reference Document

A reference document is created using the `<reference>` tag.

```

<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:cabinet key="Customer" keyType="name">
    <os:reference>
      <os:document key="Email">
        <os:field key="From:">peter@optimal-
systems.de</os:field>
        <os:field key="To:">schmidt@optimal-
systems.de</os:field>
        <os:field key="Date:">12.12.2013</os:field>
      </os:document>
      <os:document ref="documentToBeReferenced"
purpose="SOURCE"/>
      <os:register ref="locationForTheReferenceDocument"
purpose="LOCATION"/>
    </os:reference>
  </os:cabinet>
</j:jelly>

```

If the document to be referenced comes from another cabinet, another `<cabinet>` tag must also be used when searching.

Start a Workflow Process

Within a droptarget you can also initiate workflow processes.

```

<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <os:start>
    <os:process wfid="F265B0168961493887A269A30530F0ED"
clientType="OS_DESKTOP_CLIENT">
      <os:datafield type="string"
key="s_EMail">my@email.com</os:datafield>
      <os:datafield type="string"
key="d_Inbox">NO</os:datafield>
      <os:object osid="123456" objectTypeId="141222"/>
      <os:object ref="myfolder"/>
    </os:process>
  </os:start>
</j:jelly>

```

```
</os:start>
</j:jelly>
```

A workflow process is started with the `<start>` tag. Similar to other server jobs, the `<start>` tag must contain a `<process>` tag that in turn contains a tag of the `<datafield>` type.

In addition, you can also define DMS objects to be added to the workflow file.

The `<start>` tag can be found both in the root and in a `<cabinet>` tag.

Adding a Description for a Client

If the droptargets are to be used by a Client, information about the droptarget and the required fields can also be entered.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
  <description>
    <title>Filing Incoming Invoices</title>
    <longtitle>
      Filing incoming invoices in a separate register of the
customer record.
    </longtitle>
    <helptext>
      help help help
    </helptext>
    <acceptedmimetypes>
      <mimetype>image/*</mimetype>
      <mimetype>.*/*.pdf</mimetype>
    </acceptedmimetypes>
    <field required="true" type="DATE" size="199"
validate="^4[0-9]{12}([0-9]{3})?$">fieldname</field>
    <cabinet key="Customer">
      <object key="Customer">
        <field key="SUPFEST" validate="^4[0-9]{12}([0-
9]{3})?$">fieldname1</field>
        <field key="Company name">fieldname2</field>
        <field key="State">fieldname3</field>
      </object>
    </cabinet>
    <files>
      <file fileid="DOC_01" filetype="PDF"/>
      <file fileid="DOC_02" filetype="PDF"/>
    </files>
  </description>
</j:jelly>
```

The description must be given inside the `<description>` tag. General information about the DropTarget can be defined using the `<title>`, `<longtitle>` and `<helptext>` tags. The following way to define fields exist:

1. Enter a field with all required information directly under the `<description>` tag.
2. Enter a field in a `<cabinet>`-`<object>` structure. The information will be parsed from the object definition.

The `validate` attribute is optional. It must be evaluated by the Client.

If several files should also be transferred with the DropTarget, IDs can be set under the `<files>` tag. If only one file is transferred, IDs are not necessary and the internal default name (Default_Files) is used.

Tag Overview

DMS Objects And Its Fields

The following tags represent DMS objects in enaio[®].

<cabinet>

Use this tag to specify a cabinet that the action is executed in.

Can be contained in	Can contain
<root>	<search> <update> <insert> <move> <copy> <delete> <link> <select> <reference> <logger> <return>

Attributes:

- § key: The cabinet identifier.
- § keyType (optional): 'name' or 'internal_name' (default)

<folder>

Use this tag to specify a folder.

Can be contained in	Can contain
<search> <update> <insert> <move> <copy> <delete> <select> <process>	<field> <table>

Attributes:

- § osid (optional): The OSID of the folder.
- § objectType (optional): The object type ID of the folder.
- § id (optional): ID for access within the Jelly context.

§ ref (optional): Reference to another folder tag within the same Jelly context. All values of the referenced folder will be applied.

<register>

Use this tag to specify a register.

Can be contained in	Can contain
<search>	<field>
<update>	<table>
<insert>	
<move>	
<copy>	
<delete>	
<select>	
<process>	

Attributes:

- § key: The register identifier.
- § keyType (optional): 'name' or 'internal_name' (default)
- § osid (optional): The OSID of the register.
- § objectType (optional): The object type ID of the register.
- § id (optional): ID for access within the Jelly context.
- § ref (optional): Reference to another register tag within the same Jelly context. All values of the referenced register will be applied.

<document>

Use this tag to specify a document.

Can be contained in	Can contain
<search>	<field>
<update>	<table>
<insert>	
<move>	
<copy>	
<delete>	
<select>	
<process>	

Attributes:

- § key: The document identifier.
- § keyType (optional): 'name' or 'internal_name' (default)

- § osid (optional): The OSID of the document.
- § objectType (optional): The object type ID of the document.
- § id (optional): ID for access within the Jelly context.
- § ref (optional): Reference to another document tag within the same Jelly context. All values of the referenced document will be applied.

<object>

Use this tag to specify a DMS object without defining whether it is a folder, a register or a document.

Can be contained in	Can contain
<search>	<field>
<update>	<table>
<insert>	
<move>	
<copy>	
<delete>	
<select>	
<process>	

Attributes:

- § key: The DMS object identifier.
- § keyType (optional): 'name' or 'internal_name' (default)
- § osid (optional): The OSID of the DMS object.
- § objectType (optional): The object type ID of the DMS object.
- § id (optional): ID for access within the Jelly context.
- § ref (optional): Reference to another object tag within the same Jelly context. All values of the referenced DMS object will be applied.

<process>

This tag represents a workflow.

Can be contained in	Can contain
<start>	<datafield>
	<folder>
	<register>
	<document>
	<object>

Attributes:

- § wfid: ID of the workflow family that the process belongs to.

§ clientType: Client type for which the process will be started.

OS_DESKTOP_CLIENT: Rich client

OS_MOBILE_CLIENT: Mobile client (app)

OS_WEB_CLIENT: Webclient

<field>

Use this tag to specify a text field.

Can be contained in	Can contain
<folder> <register> <document> <object> <row>	Value as clear text or Jelly context variable, e.g. \${myVariable}

Attributes:

§ key: The field identifier.

§ keyType (optional): 'name' or 'internal_name' (default)

§ id (optional): ID for access within the Jelly context.

§ ref (optional): Reference to another field tag within the same Jelly context. All values of the referenced field will be applied.

<datafield>

This tag represents a data field of a workflow process.

Can be contained in	Can contain
<process>	Value as clear text or Jelly context variable, e.g. \${myVariable}

Attributes:

§ type: Data field type (STRING, INTEGER, DATE).

§ key: The data field identifier

§ id (optional): ID for access within the Jelly context.

§ ref (optional): Reference to another data field tag within the same Jelly context. All attributes and the value of the referenced field will be applied.

<table>

Use this tag to specify a table.

Can be contained in	Can contain
<folder> <register> <document> <object>	<row>

Attributes:

- § key: The table identifier.
- § keyType (optional): 'name' or 'internal_name' (default)
- § id (optional): ID for access within the Jelly context.
- § ref (optional): Reference to another table tag within the same Jelly context. All attributes and the value of the referenced table will be applied.

<row>

Use this tag to specify a table row.

Can be contained in	Can contain
<table>	<field>

Processors

<e-mail>

This tag defines an e-mail processor which facilitates the automatic readout of e-mail metadata.

Can be contained in	Can contain
<table>	<field>

Attributes:

- § id: ID for access within the JellyContext.
- § file (optional): Name under which one or several files to be inserted have been transferred to the DropTarget. If not specified, the default name is used.

Metadata:

- § TO
- § FROM
- § CC
- § BCC
- § SUBJECT
- § SENT_DATE

Supported file formats:

- § ima
- § eml
- § msg

<pdf>

This tag defines a PDF processor which facilitates the automatic readout of PDF file metadata.

Attributes:

- § id: ID for access within the Jelly context.
- § file (optional): Name under which the files to be inserted have been transferred to the DropTarget. If not specified, the default name is used.

Metadata:**Supported file formats:**

- § pdf
- § pdfa

<image>

This tag defines an image processor which facilitates the automatic readout of image file metadata.

Attributes:

- § id: ID for access within the Jelly context.
- § file (optional): Name under which the files to be inserted have been transferred to the DropTarget. If not specified, the default name is used.

Metadata:

- § NAME
- § CREATION_DATE
- § GEO_LOCATION

Supported file formats:

- § JPEG (.jpg, .jpeg, .jpe, .jif, .jfif)
- § TIFF (.tif, .tiff)
- § PSD (.psd)
- § PNG(.png)
- § BMP (.bmp, .dib)
- § GIF (.gif)
- § Camera Raw (.raw, .nef, .cr2, .orf)

<autoDetect>

This tag defines an AutoDetection processor which will try to read the meta data of each file.

Attributes:

- § id: ID for access within the Jelly context.
- § file (optional): Name under which the files to be inserted have been transferred to the DropTarget. If not specified, the default name is used.
- § documentTypes (optional): Specifies the file types that the processor should be limited to, separated by commas. If not specified, all Jelly context files are loaded.

The document types are defined in the following example.

```
<os:autoDetect id="Mail" documentTypes="msg, ima, eml"/>
```

Metadata:

- § an overview of all readable metadata can be found at the following address:

<https://tika.apache.org/1.4/api/constant-values.html#org.apache.tika.metadata.ClimateForecast.ACKNOWLEDGEMENT>

A variable is referenced directly with a Jelly context:

```
<os:field key="MAIL_TO"
keyType="Internal_Name">${context.getVariable("Mail").get("Message-
To")}</os:field>
<os:field key="MAIL_FROM"
keyType="Internal_Name">${context.getVariable("Mail").get("Message-
From")}</os:field>
<os:field key="MAIL_CC"
keyType="Internal_Name">${context.getVariable("Mail").get("Message-
Cc")}</os:field>
```

Description

Description tag texts are displayed by default in enaio® apps, provided they are specified.

The tags described here are part of the OS tag library and must therefore be specified without namespace.

<description>

This tag serves as a container for the other description tags.

Can be contained in	Can contain
<root>	<title> <longtitle> <helptext> <cabinet>

	<field>
--	---------

<title>

Use this tag to provide the name of a droptarget.

Can be contained in	Can contain
<description>	Name of the DropTarget

<longtitle>

Use this tag to provide a help text.

Can be contained in	Can contain
<description>	Description of the droptarget

<helptext>

Use this tag to provide a description of the droptarget.

Can be contained in	Can contain
<description>	Help text about the DropTarget

<cabinet>

This tag defines the cabinet to be used when parsing the object definition.

Can be contained in	Can contain
<description>	<object>

Attributes:

§ key: The cabinet identifier.

§ keyType (optional): 'name' or 'internal_name' (default)

<object>

This tag defines the object to be used when parsing the object definition.

Can be contained in	Can contain
<cabinet>	<field>

Attributes:

§ key: The object identifier

§ keyType (optional): 'name' or 'internal_name' (default)

<field>

This tag defines the object to be used when parsing the object definition.

Can be contained in	Can contain
<object> <description>	Name of the variable to be displayed

Attributes:

- § key: The field identifier
 - § keyType (optional): 'name' or 'internal_name' (default)
- Further attributes (if directly under the <description> tag):
- § required: true or false
 - § type: Field type
 - § size: length of the field

Server Jobs and Options

The following tags represent server jobs.

<search>

Use this tag to run a combined search. You can define 1-n DMS objects to be found, where the first DMS object defines the result object.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § SEARCH 1 time (default): DMS object to be searched for.
- § COMBINED_SEARCH n times: Further DMS objects in the same cabinet refining the search.
- § LOCATION 0/1 time: Location of the DMS object to be searched for. It must be a folder or register having the OSID set. The object is not suited to search for folders.

Attributes:

- § id: Identifier used to access the search results within the Jelly context.
- § mode (optional): Search mode, the following options may be chosen:
 - § FIRST (default): Returns only the first hit.
 - § ALL: Returns all hits.
 - § EXACTLY_ONE: Returns an error if there is more than one hit, or returns the hit if there is only one.

- § AT_LEAST_ONE: Returns an error if there are no hits. Otherwise returns all hits.

<update>

This tag updates a DMS object. The OSID attribute must be set.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § UPDATE 1 time (default): DMS object to be updated.

Attributes:

- § file (optional): Name under which a file to be inserted has been transferred to the DropTarget. For several files, several names must be specified. If not specified, the default name is used.

<insert>

Use this tag to insert a DMS object at a selected location. The first DMS object specified is inserted. If it is not a folder, you must define the location, i.e. a folder or register, through a second DMS object.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § When inserting a folder

- § UPDATE 1 time (default): Folder to be inserted.

- § When inserting a register or document

- § UPDATE 1 time (default): Register or document to be inserted.

- § LOCATION 1 time: Target location (folder or register)

Attributes:

- § file (optional): Name under which a file to be inserted has been transferred to the DropTarget. For several files, several names must be specified. If not specified, the default name is used.

<move>

Use this tag to move a register or document to another location. The first DMS object specified is moved, the second is the target location, i.e. a folder or register.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § MOVE 1 time (default): Register or document to be moved.
- § LOCATION 1 time: Target location (folder or register)

<copy>

Use this tag to copy a register or document to another location. The first DMS object specified is copied, the second is the target location, i.e. a folder or register.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § MOVE 1 time (default): Register or document to be copied.
- § LOCATION 1 time: Target location (folder or register)

<link>

This tag creates a reference copy. Internally, the <copy> tag is invoked with LINKDOCUMENT=1.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § MOVE 1 time (default): Object to be copied.
- § LOCATION 1 time: Target location (folder or register)

<delete>

Use this tag to delete a specified DMS object. The `osid` attribute must be set.

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

§ DELETE 1 time (default): DMS object to be deleted.

<start>

Use this tag to start a new workflow process.

Can be contained in	Can contain
<root> <cabinet>	<process>

<options>

Use this tag to pass further options to a server job.

Can be contained in	Can contain
<search> <update> <insert> <move> <copy> <delete> <select>	<option>

<option>

Use this tag to define an option value (true or false).

Can be contained in	Can contain
<options>	Option value (true or false)

Attributes:

§ key: The option identifier (see also the enaio® server-api manual).

<logger>

Use this tag to write to the log from within a Jelly context.

Can be contained in	Can contain
All tags in the OS namespace	Text to be written to the log

Attributes:

§ **level**: Log level of entries to be written to the log (see also [Commons Logging – Message Priority Levels](#)).

<return>

Use this tag to export values from a Jelly context.

Can be contained in	Can contain
All tags in the OS namespace	Value to be exported

Attributes:

§ **id (optional)**: Identifier of the value to be exported. If not specified, the default identifier (DEFAULT_RETURN_VALUE) is used.

Combined Server Jobs

<select>

This tag executes three functions:

1. search for a DMS object
2. creation of a DMS object if 1. does not return a result
3. updating of the DMS object if 1. did return a result

Can be contained in	Can contain
<cabinet>	<folder> <register> <document> <object>

Required objects:

- § **SEARCH** 1 time: DMS object to be found.
- § **COMBINED_SEARCH** n times: Further DMS objects in the same cabinet to narrow the search.
- § **INSERT** 1 time: DMS object to be inserted.
- § **UPDATE** 1 time: DMS object to be updated.
- § **LOCATION** 1 time: Location (folder or register) for search and insertion.

Attributes:

§ **id**: Identifier used to access the found or created DMS object within the Jelly context.

- § file (optional): Name under which a file to be inserted has been transferred to the DropTarget. For several files, several names must be specified. If not specified, the default name is used.
- § create (true (default) or false): Specifies whether a DMS object is created if 1. returned no result.
- § update (true (default) or false): Specifies whether the result of 1. is updated.

DropTarget tools

<utils:DateFormat>

This tag can be used to format dates.

The tag cannot include any other tags and is generally not included in a special tag. Ideally, it is used within the top level (<jelly>) or in a <cabinet> tag.

Attributes:

- § id: Identifier used to access the result of the formatting within the Jelly context.
- § inputPattern: Pattern, in accordance with java.text.SimpleDateFormat class rules, to be used when reading a date in text format.
- § parsePosition: Zero-based position from which an entry date in text format should begin being read.
- § pattern: Freely definable pattern in accordance with java.text.SimpleDateFormat class rules, to be used during formatting.
- § value: Date value to be formatted. Can be set as a character sequence or java.util.Date.

<utils:RegexFormat>

This tag can be used to format regular expressions.

The tag cannot include any other tags and is generally not included in a special tag. Ideally, it is used within the top level (<jelly>) or in a <cabinet> tag.

Attributes:

- § id: Identifier used to access the result of the formatting within the Jelly context.
- § pattern: Regular expression
- § value: Text to be formatted.

To be able to access the results of the formatting, groups must be defined in the regular expression. The results can then be accessed using the respective group's number.

Testing DropTargets

Before providing a new droptarget, it is recommended to test its functionality in a test system.

These tools may be useful:

- § `curl.exe` – The command line tool can be used as a test client to run a droptarget per URL call. It is installed by the enaio® setup and can be found in the `...\server` directory.
- § JSONView browser add-on – The Firefox add-on displays JSON documents like XML documents. Formatting and highlighting features are available and fields and objects can be collapsed or expanded.
- § enaio® enterprise-manager – This enaio® component is used to manage enaio® server, archive media, licenses, etc. All jobs sent to enaio® server can be monitored under **Extended administration > Monitoring > Job calls**.

Push Notification Service for enaio® apps

The push notification service (PnS) is now an integral part of enaio® appconnector, and will be deployed to notify the user of new messages.

The PnS checks regularly whether new messages of subscriptions, follow-ups or workflow process steps are available. If this is the case, the app notifies you of every new message even when it is not active. Users can customize the display of a message in the app.

The PnS is only available for enaio® app running on iOS.

Configuration

To configure the push notification service, edit the `osrest.properties` configuration file in the application directory

`...\services\OS_AppConnector\configuration.`

enaio® appconnector must be restarted after changing the configuration.

Key name	Description
<code>services.pushnotification.enabled</code>	To enable the PnS, set the value to 'true'.
<code>services.pushnotification.proxy.enabled</code>	UrbanAirship is integrated via the Internet. If there is a proxy server between enaio® appconnector and the Internet, set this parameter to 'true.'
<code>services.pushnotification.proxy.address</code>	If the proxy server is enabled, the proxy server address or IP will be entered here.

	Example: proxy.my-company.com
services.pushnotification.proxy.port	If the proxy server is enabled, the port from which the proxy server with the specified address can be accessed will be entered here. Example: 3128
services.pushnotification.proxy.username	If the proxy server is enabled and a login user name is required, it will be entered here.
services.pushnotification.proxy.password	If the proxy server is enabled and a login password is required, it will be entered here.
services.pushnotification.production	The default setting 'true' must be left unchanged for notifications. In development environments, 'false' switches notifications off and development notifications on.

To test your configuration, trigger, for example, a subscription or a follow-up in the app or enaio® client. Then check whether you have received an app notification about the change even if the app has not been active.

Information about possible errors in your PnS configuration can be found in the enaio® appconnector log. The `osrest.log` file is located in the `logs` application directory.

Attachment

Integrating enaio® appconnector with .NET

The following sections describe technologies and strategies for integrating enaio® appconnector with .NET using the REST interface.

Note that OS_AppConnector integration requires software developers to have experiences in .NET programming.

Integration Possibilities

enaio® appconnector is implemented as a REST Web service with the name `OSRest`. Features of REST Web services are provided by calling specific URLs. The features' results are made available as Web server response.

The following strategies are available for integrating enaio® appconnector with .NET. You can use either the `WebClient` or the `WebRequest` class. Both classes are found in the `System.Net` namespace of the framework.

Further information on the classes is available in the MSDN (Microsoft Developer Network) documentation.

The 'WebClient' Class

The `WebClient` class offers a highly convenient way to call enaio® appconnector. The class' advantage is that its features are already encapsulated in methods. However, some settings, such as custom HTTP headers, cannot be made in detail.

The following example illustrates a call with the `WebClient` class:

```
WebClient client = new WebClient();
string result =
client.DownloadString("http://localhost:8080/osrest/api/documents");
Console.WriteLine(result);
```

This way very few lines of code are necessary to list all document types.

There are also more functions you can use, for example:

- \$ UploadString
- \$ OpenRead
- \$ DownloadData
- \$ DownloadFile

Further information on these features is available in the MSDN documentation.

The 'WebRequest' Class

Calling with the `WebRequest` class requires you to handle the responses as a separate stream. The class' advantage is that it allows to fully control communication. However, the class results in higher programming efforts.

The following example illustrates a call with the `WebRequest` class:

```
HttpWebRequest request =
    (HttpWebRequest)WebRequest.Create(@"http://localhost:8080/osrest/api
/documents");
request.KeepAlive = true;
request.Method = "GET";
WebResponse result = request.GetResponse();
Stream resultStream = result.GetResponseStream();
StreamReader readStream = new StreamReader(resultStream,
Encoding.Default);
Console.WriteLine(readStream.ReadToEnd());
```

The result of this code example is also a list of all document types.

HTTP Multipart Call

Use a HTTP Multipart call to send fields containing a file to a `DropTarget`.

The call must consist of two parts: the JSON content, called `data`, and the file, called `file`.

As there is no suitable method in the .NET framework to execute a HTTP Multipart call, the source text example below describes such a call:

```
public static void WriteMultiParts(this WebRequest request,
IDictionary<string, object> parts)
{
    string boundary = "-----" +
DateTime.Now.Ticks.ToString("x");
    byte[] boundarybytes = Encoding.ASCII.GetBytes("\r\n--"
+ boundary + "\r\n");
    request.ContentType = "multipart/form-data; boundary=" +
boundary;
    request.Method = "POST";
    var buffer = new MemoryStream();
    foreach (var part in parts)
    {
        if (part.Value == null)
        {
            continue;
        }

        if (part.Value is FileInfo)
        {
            var file = (FileInfo)part.Value;
            if (!file.Exists)
            {
                throw new
FileNotFoundException(string.Format("The file='{0}' in part='{1}' is
not existing.", file.FullName, part.Key), file.FullName);
            }

            buffer.Write(boundarybytes, 0,
boundarybytes.Length);
            var contenttype =
Helper.GetMediaType(file.Extension);
            string header = string.Format("Content-
Disposition: form-data; name=\"{0}\"; filename=\"{1}\" \r\nContent-
Type: {2} \r\n\r\n", part.Key, file.FullName, contenttype);
            byte[] headerbytes =
Encoding.UTF8.GetBytes(header);
            buffer.Write(headerbytes, 0,
headerbytes.Length);
```

```

        var fileStream = new FileStream(file.FullName,
        FileMode.Open, FileAccess.Read);
        var fileBuffer = new byte[4096];
        int bytesRead;
        while ((bytesRead = fileStream.Read(fileBuffer,
        0, fileBuffer.Length)) != 0)
        {
            buffer.Write(fileBuffer, 0, bytesRead);
        }

        fileStream.Close();
    }
    else
    {
        buffer.Write(boundarybytes, 0,
        boundarybytes.Length);
        string formitem = string.Format("Content-
        Disposition: form-data; name=\"{0}\"\\r\\n\\r\\n{1}", part.Key,
        part.Value);
        byte[] formitembytes =
        Encoding.UTF8.GetBytes(formitem);
        buffer.Write(formitembytes, 0,
        formitembytes.Length);
    }

    }

    byte[] trailer = Encoding.ASCII.GetBytes("\\r\\n--" +
    boundary + "--\\r\\n");
    buffer.Write(trailer, 0, trailer.Length);
    buffer.Close();
    var requestStream = request.GetRequestStream();
    var data = buffer.ToArray();
    requestStream.Write(data, 0, data.Length);
}

```

Authentication

Authentication is required if you wish to connect to enaio® appconnector.

The following authentication procedures are available in enaio® appconnector:

§ Negotiate

§ NTLM

§ Basic Authentication

To enable authentication, you must explicitly set the `Credentials` property for the `WebClient` or the `WebRequest` object.

NTLM Authentication

To enable NTLM authentication, you must set the `Credentials` property to the value `CredentialCache.DefaultCredentials`. The account of the currently logged in user is then used as login information.

If you perform the call from within an ASP.NET application you must set the value to `CredentialCache.DefaultNetworkCredentials`.

Further information is available in the MSDN documentation.

Basic Authentication

To enable basic authentication, you must set the value of the `Credentials` property to the value `NetworkCredential("user", "password")`.

If your system allows NTLM or Negotiate authentication, primarily this authentication procedure will be used. Basic authentication may need to be enforced (see 'Enforcing Specific Authentication Modes').

Enforcing Specific Authentication Modes

Various authentication methods are provided by the .NET Framework. You can use the following example to list them all:

```
IEnumerator moduleEnumerator =
AuthenticationManager.RegisteredModules;
while (moduleEnumerator.MoveNext())
{
    Console.WriteLine(moduleEnumerator.Current.ToString());
}
```

All mechanisms apply, in descending order, according to their availability.

If you want to force a specific authentication mechanism for enaio® appconnector, such as basic authentication, you must explicitly disable or remove all other authentication mechanisms.

The following example demonstrates how to force basic authentication:

```
AuthenticationManager.Unregister("Negotiate");
AuthenticationManager.Unregister("NTLM");
```

Please note that this is a static call. If you want to change the authentication mechanism to one that has been disabled before, you must activate it. This is done via the `Register` method.

Handling JSON Responses Given by enaio® appconnector

Results given by enaio® appconnector are available as JSON-formatted HTTP contents. If you want to convert the JSON format to files readable by .NET, use one of the following strategies:

- § Serialization with the 'DataContract' Attribute
- § Conversion to XML and Parsing with XPath

Serialization with the 'DataContract' Attribute

You can mark a .NET object with the `DataContract` attribute and set corresponding `DataMember` properties in the class. These attributes are found in the `System.Runtime.Serialization` namespace. In order to use this namespace, you must add a reference to the `System.Runtime.Serialization` assembly in your project from the .NET Framework. By using the `DataContractJsonSerializer` class, you can then easily convert JSON-formatted results to .NET objects.

Further information is available in the MSDN documentation.

Conversion to XML and Parsing with XPath

An alternative way to make JSON formatted result data readable for your application is to convert the data into XML format and then parse it or evaluate it using XPath expressions.

The following example presents this procedure:

```
XmlDictionaryReader reader =  
JsonReaderWriterFactory.CreateJsonReader(Encoding.Default.GetBytes(j  
sonDocuments), XmlDictionaryReaderQuotas.Max);  
XmlDocument xml = new XmlDocument();  
xml.Load(reader);
```

Please note that this example is valid only if your project includes a reference to the `System.Runtime.Serialization` assembly of the .NET Framework.

API Documentation

General

OSRest provides a HTTP-based programming interface (API); all calls are stateless and resource-oriented (REST). The API is used to bundle and simplify complex function calls in enaio®. This includes mapping of all index data to a schema which is useful for the specific application.

As HTTP is used for logging, correct URL encoding has to be ensured for all calls.

Authentication

OSRest supports default HTTP authentication with the following procedures:

- § NTLM
- § HTTP SPNEGO (Kerberos)
- § HTTP Basic Authentication (RFC2617)

With the Windows-based authentication procedures NTLM and SPNEGO, authentication is delegated to the Windows operating system of the OSRest server. The alternative HTTP basic authentication can either use Windows or internal enaio® accounts. If required, OSRest can also be configured for a technical user to run without authentication. The URL to the Admin interface can only be opened with supervisor rights.

Services

OSRest provides different services:

- § DocumentService – Methods for searching index data
- § DocumentFileService – Methods for handling document files
- § NotificationService – Access to enaio® notifications
- § SessionService – Information on user sessions
- § ServiceInfoService – General information on the OSRest service
- § OSFileService – Anonymous service to create enaio®-internal link files
- § WorkflowService – Methods for starting and editing workflows
- § ObjDevService – Methods for working with object definitions in JSON format
- § OrganizationService – Methods for users and groups, and also for e-mail dispatch
- § IconService – Methods for retrieving catalog icons

All methods by default return [JSON](#)-encoded results.

Error Messages

In case of an error, an HTTP status code not equal to 200 is returned. Application errors basically have the status code 500 and additionally the JSON-encoded error message.

```
{
  "errormessage": "EcmException => The saved search with ID
564546754 was not found...",
  "id": 500
}
```

DocumentService

Next to the methods for searching and displaying index data, DocumentService also offers features for marking favorites. These features are part of an especially configurable user portfolio where the favorite enaio® objects are administered.

/osrest/api/documents

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a shorter version of the user object definition.

/osrest/api/documents/cabinets

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all cabinets visible to the user.

/osrest/api/documents/cabinets/[Cabinet]

§ Supported query methods: GET

§ Supported result formats: JSON, XML

The method returns all folders of a cabinet with the indicated name.

Optional parameters are:

§ metadata (string): File name of an alternative metadata mapping.

§ format (string): output format (json, xml). The default setting is json.

§ pagesize (integer): Maximum number of hits.

§ offset (integer): Hit offset for browsing the hits.

§ page (integer): The requested page for browsing the hits.

/osrest/api/documents/storedqueries

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all saved queries.

[/osrest/api/documents/storedqueries/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the result of the saved query with the specified ID.

Optional parameters are:

§ metadata (string): File name of an alternative metadata mapping.

§ pagesize (integer): The maximum number of hits for the call.

§ offset (integer): Hit offset for browsing the hits.

§ page (integer): The requested page for browsing the hits.

§ HTTP query parameter [object].[field] (e.g. email.sender=Peter)

```
.../osrest/api/documents/storedqueries/1234?Person.firstname=Gustav&
Person.lastname=Gans&metadata=MyMapping
```

[/osrest/api/documents/tray](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns objects from the user tray.

Optional parameters are:

§ metadata (string): File name of an alternative metadata mapping.

[/osrest/api/documents/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON, XML

The method returns index data of the object with the specified ID.

Optional parameters are:

§ metadata (string): File name of an alternative metadata mapping.

§ insertInfo (Boolean): Outputs all object types that can be added.

§ insertInfo_verbose (Boolean): Outputs all object types that can be added, in greater detail.

§ locale (DE, EN, SP, FR, IT, NL, SV, HU, PL): Use one of these capital-letter abbreviations to specify the output language.

§ lockinfo (Boolean): Information on the user who locked the file, as well as the date and time that it was locked. Alternative values are UNLOCKED, SELF, and EXTERNAL.

§ timestamps (Boolean): If set to 'true,' date, time, and date/time basic parameters will be returned as Java timestamps.

§ format (string): output format (json, xml). The default setting is json.

- § refresh (Boolean): Ignores cached data and retrieves the requested data again.
- § DEPRECATED: verbose (Boolean): Outputs all field data (please use documents/search/{Id})

Note on insertInfo parameters:

Normally only the mandatory fields of an object type would be returned. For additional fields, an insertFields tag must be added to the schema file.

Optional fields in insertInfo:

```
<cabinet name="Customer">
  <object name="Document">
    <property name="title" field="Type"/>
    <property name="info" field="Description"/>
    <insertFields active="true"> <!-- true is default -->
      <property field="Description"/> <!-- Description is
not a mandatory field -->
    </insertFields>
  </object>
</cabinet/>
```

[/osrest/api/documents/parents/\[id\]](#)

- § Supported query methods: GET
- § Supported result formats: JSON, XML

The method returns a sorted, linear list of parent objects for an object with the given ID. In addition to this list, it is also possible to display the search tree of a location, which is particularly important when displaying multiple locations.

Optional parameters are:

- § metadata (string): File name of an alternative metadata mapping.
- § format (string): output format (json, xml). The default setting is json.
- § tree (Boolean): If 'true,' the location hierarchy of all of an object's locations will be displayed as a tree. This parameter should always be included and set to "true" – otherwise problems may be encountered when dealing with objects that have multiple locations. The "simple" linear output is still used for compatibility reasons.

Example output (the object ID here was 14007051):

```
{
  "pagesize": 500,
  "totalHits": 3,
  "more": false,
  "documents": [
    {
      "id": "4475",
      "type": "FOLDER",
      "fields": {
        "title": "Stoz Pumpenfabrik GmbH",
        "info": "User 0234-OS"
```

```

    },
    "fav": false,
    "typeless": false
  },
  {
    "id": "14003866",
    "type": "REGISTER",
    "fields": {
      "title": "Previous stock",
      "info": "enaio introduction, 50 user, jukebox"
    },
    "fav": false,
    "typeless": false
  },
  {
    "id": "14007051",
    "type": "DOCUMENT",
    "fields": {
      "title": "Offer: STP-AN-1/13",
      "info": "ECM introduction..."
    },
    "fav": false,
    "typeless": false
  }
]
}

```

[/osrest/api/documents/objectHierarchy/\[id\]](/osrest/api/documents/objectHierarchy/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

The method provides the directory structure of a folder or register up to a variable depth.

Optional parameters are:

§ depth (int): Depth of the directory search tree.

§ hash (string): Hash value to be matched with the new, cumulatively calculated hash value. If the same, no tree is returned.

Example output (id=4779;depth=10):

```

http://localhost:8080/osrest/api/documents/objectHierarchy/4779?depth=10
{
  "pagesize": -1,
  "totalHits": 1,
  "more": false,
  "documents": [
    {
      "id": "4779",
      "type": "FOLDER",
      "fields": {
        "title": "null",
        "info": "0"
      },
      "fav": false,
      "typeless": false,
      "hasPages": false,
      "lastmodified": "1295711263",
      "children": {
        "hash": "-1010811263",

```

```

    "documents": [
      {
        "id": "14043114",
        "type": "REGISTER",
        "fields": {},
        "fav": false,
        "typeless": false,
        "hasPages": false,
        "lastmodified": "1263306900",
        "children": {
          "hash": "-363596267",
          "documents": [
            {
              "id": "4780",
              "type": "DOCUMENT",
              "fields": {},
              "fav": false,
              "typeless": false,
              "hasPages": false,
              "lastmodified": "1231420126"
            },
            {
              "id": "5770",
              "type": "DOCUMENT",
              "fields": {},
              "fav": false,
              "typeless": false,
              "hasPages": false,
              "lastmodified": "1231420126"
            }
          ]
        }
      }
    ],
    "hash": "840007432"
  }

```

[/osrest/api/documents/variants/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns all variants for the object with the given ID. The variant tree is displayed as follows:

```

{
  "active": false,
  "id": 14007049,
  "parentId": 0,
  "version": "Original",
  "children": [
    {
      "active": true,
      "id": 14007051,
      "parentId": 14007049,
      "version": "1.0.0",
      "children": [
        {
          "active": false,

```

```
        "id": 14041838,  
        "parentId": 14007051,  
        "version": "1.1.0",  
        "children": []  
      }  
    ],  
    },  
    {  
      "active": false,  
      "id": 14119107,  
      "parentId": 14007051,  
      "version": "2.0.0",  
      "children": []  
    }  
  ]  
}
```

[/osrest/api/documents/explore/\[id\]](/osrest/api/documents/explore/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the contents of a folder/register with the specified ID.

Optional parameters are:

§ metadata (string): File name of an alternative metadata mapping.

§ format (string): output format (json, xml). The default setting is json.

§ pagesize (integer): Maximum number of hits.

§ offset (integer): Hit offset for browsing the hits.

§ insertInfo (Boolean): Outputs all object types that can be added (see note on insertInfo parameters)

§ page (integer): The requested page for browsing the hits.

[/osrest/api/documents/insert/\[locationId\]](/osrest/api/documents/insert/[locationId])

§ Supported query methods: POST

§ Supported output formats: JSON

The query has to have the content type multipart/form-data (RFC 1876).

Creates an object in the folder or register with the specified location ID. If you are creating a folder, simply leave the location ID unspecified.

The json schema is the same as that returned with explore + the 'insertInfo' parameter. The json schema is the same as that returned with explore + 'insertInfo' parameters.

The 'type' attribute must be included in the field for radio buttons ("type": "RADIO").

Variants: If no index data are transferred in the request, the index data will be taken from the parent document.

Optional parameters are:

- § `setvariantactive=true`: Sets the new variant as the active variant, e.g.
`http://demo.optimal-systems.org/osrest/api/documents/insert/14133432?setvariantactive=true`
- § `archivable=true`: Sets the document as archivable.
- § `islink=true`: Creates another location for the document.

```
{
  "objectTypeId": "262159",
  "mainTypeId": "1",
  "fields": {
    "Project type": {
      "internalName": "i_projecttype",
      "value": "PUA - General"
    },
    "Type": {
      "value": "Technology"
    },
    "3": {
      "internalName": "i_visiblefor",
      "value": "DEMO(g)"
    }
  }
}
```

[/osrest/api/documents/update/\[id\]](#)

- § Supported query methods: POST
- § Supported output formats: JSON

The method modifies the object with the specified ID. It is used similarly to the `documents/insert` method – the `ObjectTypeId` can optionally be included to increase the speed of the call. Furthermore, only explicitly specified fields can be modified in enaio®. The call must have the content type *multipart/form-data* ([RFC 1867](#)).

Optional parameters are:

- § `replacefiles` (Boolean): Replaces the file of a document if set to 'true.'

[/osrest/api/documents/search](#)

- § Supported query methods: POST
- § Supported output formats: JSON

Optional parameters are:

- § `pagesize` (integer): Size of the result set. This can be used for paging.
- § `offset` (integer): Detailed offset, independent from the `pagesize`.

- § **page (integer):** The page to be displayed during paging. Overwrites the `offset` parameter. Offset is calculated as `page * pagesize`.
- § **maxhits (integer):** Maximum number of hits to be loaded. This can be more than the number specified in `pagesize`. This allows caching to be regulated so that subsequent pages can be returned more quickly.
- § **rights (Boolean):** The object rights for every object should also be output in the hit list.

Use this method to search for an object using field values and/or basic parameters. The search can be narrowed (combined search) by specifying more object types. Object types are specified with either the internal name, the cabinet name, the OSID, or the object type ID. Fields are specified as field objects in "fields." They can be given the "internalName" attribute, otherwise the object identifier will be taken as the simple name. The basic parameters can also be negated by setting the `negate` key to `true`. The basic parameters can be found in the table further below. Areas can be queried with all numerical, date, time, and date/time values.

The following table lists all operators:

Operator	Description
>Value	All values greater than Value.
>=Value	All values greater than or equal to Value.
!=Value	All values not equal to Value.
<Value	All values smaller than Value.
<=Value	All values smaller than or equal to Value.
Value1-Value2	All values within the range from Value1 to Value2.

The hit list output includes all index data and metadata, ensuring more overhead than other search methods available in this API.

Note: If simple names are used, the cabinet name must also be specified, as the simple name is not unique across the system.

```
{
  "query": {
    "objectTypeId": "262220",
    "osid": "14131857",
    "cabinet": "i_SeeleGersthofen",
    "name": "i_Documents",
    "fields": {
      "Contract basis": {
        "value": "1"
      },
      "Create archived receipt": {
        "value": "1"
      }
    },
    "baseparams": {
```

```

    "Creator": {
      "value": "DEMO"
    },
    "Created": {
      "value": "1028114820000"
    },
    "Modifier": {
      "value": "DEMO",
      "negate": "true"
    }
  },
  "additionalQueries": [
    {
      "objectTypeId": "54",
      "fields": {
        "Project type": {
          "value": "PUA - General",
          "internalName": "i_projecttype"
        },
        "Project no.": {
          "value": "123",
          "internalName": "i_projectno."
        },
        "Project name": {
          "value": "test value",
          "internalName": "i_projectname"
        },
        "Visible for": {
          "value": "DEMO(g)",
          "internalName": "i_visiblefor"
        }
      },
      "baseparams": {
        "Creator": {
          "value": "DEMO"
        },
        "Created": {
          "value": "1028114820000"
        },
        "Modifier": {
          "value": "DEMO",
          "negate": "true"
        }
      }
    }
  ]
}

```

The results list can be configured with the "result_config" parameter. The following values are available:

Parameters	Type	Description	Default
pagesize	Integer	Number of hits – see the URI parameter of the same name. This	null

		parameter is prioritized over the URI parameter.	
offset	Integer	Paging offset – see the URI parameter of the same name. This parameter is prioritized over the URI parameter.	0
maxhits	Integer	Maximum number of hits – see the URI parameter of the same name. This parameter is prioritized over the URI parameter.	500
deny_empty	Boolean	If 'true,' null field values will NOT be displayed in the results list.	false
normalize_values	Boolean	If 'true,' Date, Time, and DateTime fields are converted to timestamps (with milliseconds), otherwise the output will use the respective server format.	false
fieldsschema	Object	Specifies the order of hits.	null

Field schema:

Parameters	Type	Description	Default
internalName	String	Identifier of the field	null
displayName	String	Identifier of the field	null
objectTypeId	String	Identifier of the field	null
dbName	String	Identifier of the field	null
sort_order	\[ASC, DESC\]	Ascending or descending order	ASC
sort_pos	Integer	Position in the sort order	Position within the field schema

Example result configuration

```
{
  "query": {
    "cabinet": "Customer",
    "name": "Customer",
    "result_config": {
      "fieldsschema": [
        {
          "dbName": "id",
          "sort_position": 0,
          "sort_order": "DESC"
        }
      ]
    }
  }
}
```

```

    "pagesize": 100,
    "offset": 100,
    "maxhits": 500,
    "deny_empty": true,
    "normalize_values": false
  }
}

```

A full text search can also be integrated. To do this, simply fill out the "vtx_query" attribute for any object.

Combined Search and Full Text Search

```

{
  "query": {
    "objectTypeId": "3",
    "vtx_query": "Olaf",
    "fields": {
      "Sector": {
        "value": "Industry"
      },
      "Status": {
        "value": "2"
      }
    }
  },
  "additionalExamples": [
    {
      "objectTypeId": "131079",
      "vtx_query": "Björn"
    }
  ]
}

```

The following basic parameters from the server API manual are currently supported. Unset keys are irrelevant in terms of value:

Basic parameters	Values	Explanation
Creator	User name ¹	A list of user names of ECM users. The user names are queried using OR logic. If negated, however, the user names will be linked with AND.
Created	Java timestamp ¹	
Modifier	User name ¹	A list of user names of ECM users. The user names are queried using OR logic. If negated, however, the user names will be linked with AND.
Modified	Java timestamp ¹	
Owner	User name ¹	A list of user names of ECM users. The user names are queried using OR

		logic. If negated, however, the user names will be linked with AND.
ArchiveState	ARCHIVED ARCHIVABLE NOT_ARCHIVABLE NO_PAGES REFERENCE	A list of these values. They will then be linked with OR.
Archivist	User name ¹	A list of user names of ECM users. The user names are queried using OR logic. If negated, however, the user names will be linked with AND.
ArchiveDate	Java timestamp ¹	
Locked	User name ¹	A list of user names of ECM users. The user names are queried using OR logic. A list entry can also contain an empty string for 'UNLOCKED' or a * for 'blocked by somebody.' If negated, however, the user names will be linked with AND.
HasVariants	true	
Version	Integer ¹	
RetentionDate	Java timestamp ¹	
RetentionPlanned Date	Java timestamp ¹	
Left	Integer ¹	
SystemID	Integer ¹	
ForeignID	Integer ¹	
DocumentPageCount	Integer ¹	
Registers	true false	

¹ Value can be negated by setting a "negate" key.

Note: Only use keys that begin with a lowercase letter! Keys that begin with uppercase letters are deprecated.

The information on rights, which can be additionally requested via the GET parameter rights=true, indicates the following:

objModify	The user is allowed to edit the document files.
objDelete	The user is allowed to delete the object.
objExport	The user is allowed to open and/or export the object.

indexModify	The user is allowed to change the object's index data.
-------------	--

[/osrest/api/documents/search/\[id\]](/osrest/api/documents/search/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

Use this method to search for an object with the specified OSID. The output, with all index data and metadata, will have the same structure here as the output of the POST search method.

Optional parameters are:

- § refresh (Boolean): The cache will be emptied and the results will be requested from the server again.
- § locale (DE, EN, SP, FR, IT, NL, SV, HU, PL): Use one of these capital-letter abbreviations to specify the output language.
- § lockinfo (Boolean): Information on the user who locked the file, as well as the date and time that it was locked. Alternative values are UNLOCKED, SELF, and EXTERNAL.
- § timestamps (Boolean): If set to 'true,' date, time, and date/time basic parameters will be returned as Java timestamps.

[/osrest/api/documents/vtx?query=\[search term\]](/osrest/api/documents/vtx?query=[search term])

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the result of a full text search with the specified search terms.

Optional parameters are:

- § metadata (string): File name of an alternative metadata mapping.

.../osrest/api/documents/vtx?query=My%20searchterms

</osrest/api/documents/vtx>

§ Supported query methods: POST

§ Supported result formats: JSON

§ Supported input formats: JSON

The method provides the results of a full text search with the given search term.

Required parameters are:

- § query

Optional parameters are:

§ facets (Map<String, List<String>>): multiple values are permissible per facet type

§ maxHits (integer): Maximum number of hits

§ objectIds (integer[]): List of object IDs

§ objectTypeIds (Integer[]): List of object type IDs

Input JSON:

```
{
  "query": "demo*",
  "maxHits": 50,
  "facets": {
    "creator": [
      "demo"
    ],
    "objtype": [
      "262144",
      "131114",
      "393225"
    ]
  }
}
```

Output JSON:

```
{
  "facets": {
    "creator": [
      {
        "value": "demo",
        "hits": 26
      }
    ],
    "objtype": [
      {
        "value": "262144",
        "hits": 15
      },
      {
        "value": "131114",
        "hits": 5
      },
      {
        "value": "393225",
        "hits": 2
      }
    ]
  },
  "facetCount": 8,
  "documentResult": {
    "totalHits": 20,
    "more": false,
    "documents": [
      {
        "id": "14121247",
        "type": "DOCUMENT",

```

```

    "fields": {
      "title": "Documents",
      "info": "PM New Version enaio (draft SJ)"
    },
    "fav": false,
    "typeless": false,
    "pages": 1,
    "lastmodified": "1393589826000",
    "preview": "Documents 20050321 PM New Version enaio (draft SJ)
<em>DEMO</em> <em>DEMO</em> 20140228 1.0.0 <em>DEMO</em>
<em>DEMO</em>"
  }, {...}
]
}
}

```

[/osrest/api/documents/vtx/autocomplete/\[term\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method gives up to five suggestions for auto-completing the provided term.

[/osrest/api/documents/baseparams/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the enaio® properties for the given object ID.

[/osrest/api/documents/history/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the object history for the given object ID.

Example of output:

```

[
  {
    "time": 1392211328000,
    "action": 7,
    "actionname": "Document output",
    "description": "The document was read by the user, printed,
or otherwise output. No changes were made.",
    "username": "ROOT",
    "userdisplayname": "",
    "info": "The document was opened in read-only mode."
  },
  {
    "time": 1392211327000,
    "action": 31,
    "actionname": "Object info",
    "description": "Log entry from the business model.",
    "username": "DEMO",
    "userdisplayname": "John Doe",
    "info": "Document preview was retrieved."
  },
]

```



```
{
  "time": 1203583524000,
  "action": 7,
  "actionname": "Document output",
  "description": "The document was read by the user, printed,
or otherwise output. No changes were made.",
  "username": "CONTRACT",
  "userdisplayname": "",
  "info": "Document downloaded for editing"
},
{
  "time": 1203583489000,
  "action": 7,
  "actionname": "Document output",
  "description": "The document was read by the user, printed,
or otherwise output. No changes were made.",
  "username": "CONTRACT",
  "userdisplayname": "",
  "info": "Document downloaded in read-only mode"
}
]
```

[/osrest/api/documents/notes/\[id\]](/osrest/api/documents/notes/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

Parameter:

§ id (int): Object ID

The method returns notes for the given object ID.

Example of output:

```
[
  {
    "id": 14121121,
    "creator": {
      "id": 22,
      "name": "DEMO",
      ...
    },
    "modifier": {
      "id": 77,
      "name": "MEIER",
      ...
    },
    "color": "2",
    "colorName": "YELLOW",
    "text": "This is an ideal test document for notes.",
    "date": 1465909598000,
    "creationDate": 1392290415000
  },
  ...
]
```

[/osrest/api/documents/notes/\[id\]](/osrest/api/documents/notes/[id])

§ Supported query methods: POST

§ Supported input formats: JSON

§ Supported result formats: JSON

Parameter:

§ id (int): Object ID

The method creates a new note for the given object ID. As a result, the complete list of all notes is returned (like with GET).

Input data example:

```
{
  "color": "3",
  "text": "Note via REST API"
}
```

[/osrest/api/documents/notes/\[id\]/\[noteld\]](#)

§ Supported query methods: DELETE

§ Supported result formats: JSON

The method deletes a note with the provided object ID and note ID. As a result, the complete list of all notes is returned (like with GET).

[/osrest/api/documents/notes/remove/\[id\]/\[noteld\]](#)

§ Supported query methods: DELETE

§ Supported result formats: JSON

Parameter:

§ id (int): Object ID

§ noteld (int): Note ID

The method deletes a note with the provided object ID and note ID. As a result, the complete list of all notes is returned (like with GET).

[/osrest/api/documents/notes/update/\[id\]](#)

§ Supported query methods: POST

§ Supported input formats: JSON

§ Supported result formats: JSON

The method applies changes to the passed note on the object with the passed object ID. As a result, the complete list of all notes is returned (like with GET).

Parameter:

§ id (int): Object ID

Input data example

```
{
  "id": "17",
  "color": "3",
}
```

```
}      "text": "Note via REST API"
```

[/osrest/api/documents/portfolios](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns all portfolios visible to the user.

Optional parameters are:

§ creator (string): Narrows the search to portfolios created by a specific user.

§ recipient (string): Narrows the search to portfolios created for a specific user.

§ subject (string): Narrows the search to portfolios with a specified subject.

[/osrest/api/documents/portfolio/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns the content of a portfolio.

Optional parameters are:

§ pagesize (integer): Maximum number of hits.

§ offset (integer): Hit offset for browsing the hits.

§ page (integer): The requested page for browsing the hits.

[/osrest/api/documents/portfolio/{id}/add/{docid}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method adds a document to a portfolio.

[/osrest/api/documents/portfolio/{id}/remove/{docid}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method removes a document from a portfolio.

[/osrest/api/documents/favorites](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all objects of the favorites portfolio.

[/osrest/api/documents/favorites/count](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the number of all objects of the favorites portfolio.

[/osrest/api/documents/favorites/add/\[id\]](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Inserts the object with the specified enaio® ID of the favorites portfolio.

[/osrest/api/documents/favorites/delete/\[id\]](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Deletes the object with the specified enaio® ID from the favorite portfolio.

[/osrest/api/documents/raw/\[id\]](#)

Deprecated! Use [/osrest/api/documents/search/\[id\]](#) instead.

§ Supported query methods: GET

§ Supported result formats: XML, HTML, JSON

The method returns object information in the internal enaio® XML format (see server API documentation). The output can also be in HTML format. The output serves as the basis for enaio® detailsviewer.

Optional parameters are:

§ format (string): Output format (XML, JSON).

§ locale (string): The output language can be set by specifying an ISO language code (de, en, etc.).

[/osrest/api/documents/annotations/{objectId}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns all annotations in the document associated with an ECM object.

[/osrest/api/documents/annotations](#)

§ Supported query methods: POST

§ Supported result formats: JSON

§ Supported input formats: JSON

The method adds an annotation to the document associated with the ECM object, or updates an existing annotation. The number of the page containing the annotation is zero-based, i.e. the first page is 0.

The pageXStart, pageYStart, and corresponding End values are per mil values (percentage values times 10). Currently, the only supported type is ANNOT, which stands for text annotations. More annotation types will be added soon.

The backgroundColor is RGBA. Data and text are currently identical. In a further development stage, data will also be able to contain binary Base64 data, while text must contain searchable text.

Input JSON

```
{
  "id": 0,
  "objectId": 123,
  "page": 0,
  "pageXStart": 780.0,
  "pageYStart": 542.1,
  "pageXEnd": 0,
  "pageYEnd": 0,
  "type": "ANNOT",
  "backgroundColor": "FF000000",
  "data": "I am a text annotation",
  "text": "I am a text annotation"
}
```

[/osrest/api/documents/annotations/delete/{objectId}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method removes one or all annotations from the document associated with an ECM object.

Optional parameters are:

§ id (integer): The ID of the annotation to be removed from the document. If no ID is specified, all annotations in the document will be deleted.

[/osrest/api/documents/annotations/pdf/{objectId}](#)

§ Supported query methods: POST

§ Supported result formats: JSON

This method returns the PDF version of the document associated with the ECM object, including annotations. The annotations will be embedded in the PDF.

DocumentFileService

[/osrest/api/documentfiles/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns detailed information on the object with the specified ID.

The result contains the following information:

§ fav: The document has been marked as favorite by the user.

§ typeless: The document does not have a type.

§ contentviewerUrl: URL to open the object in the combined view of OS_contentviewer.

§ contentviewerDocumentUrl: URL to open the object in the document view of OS_contentviewer.

§ contentviewerIndexdataUrl: URL to open the object in the index data view of OS_contentviewer.

§ oswebLinkUrl: URL to open the object in enaio® webclient.

§ files: List of files associated with the document.

```
{
  fav: false
  typeless: false
  contentviewerUrl: "https://..."
  contentviewerDocumentUrl: "https://..."
  contentviewerIndexdataUrl: "https://..."
  oswebLinkUrl: "https://..."
  pageCount: 1
  -files: [
    "1.doc"
  ]
}
```

[/osrest/api/documentfiles/\[id\]/\[page\]](#)

§ Supported query methods: GET

§ Supported result formats: application/octet-stream

The method returns a certain page for the object with the indicated page name or page number (see [/osrest/api/documentfiles/\[id\]](#)). The value 1 can also be passed for one-page documents.

[/osrest/api/documentfiles/\[id\]/pdf](#)

§ Supported query methods: GET

§ Supported result formats: application/octet-stream

The method converts the documents of the object with the specified ID to PDF.

[/osrest/api/documentfiles/\[id\]/zip](#)

§ Supported query methods: GET

§ Supported result formats: application/octet-stream

The method summarizes documents of the object with the specified ID in a ZIP file

[/osrest/api/documentfiles/\[id\]/osfile](/osrest/api/documentfiles/[id]/osfile)

§ Supported query methods: GET

§ Supported result formats: application/octet-stream

The method creates an internal enaio® link file for the object with the specified ID.

</osrest/api/documentfiles/pdf>

§ Supported query methods: POST

§ Supported result formats: application/pdf

Optional parameters are:

§ **whitepage** (Boolean): If 'true,' a blank page will be inserted as a separator between each document in the merged document.

§ **forceprint** (Boolean): If 'true,' application code will be inserted into the PDF that will cause the PDF reader to open the print dialog.

The method merges the PDF versions of multiple original documents, specified via ID in the POST body, into a single PDF document, which is then returned. The GET parameter *forceprint* can be used to add application code to the target PDF document that will cause the PDF reader to open the print dialog directly after opening the document. A maximum of 200 documents can be merged into one PDF. If one or more documents could not be processed, a PDF will not be returned as a result. Instead, a JSON object will be returned containing the failed object IDs.

```
{
  "pdfname" : "merge.pdf",
  "ids" : [
    14024030,
    14024032,
  ]
}
```

</osrest/api/documentfiles/zip>

§ Supported query methods: POST

§ Supported result formats: application/zip

The method packs the original documents, specified via ID in the POST body, into a ZIP archive, which is then returned. Optionally, the name of each listed document in the ZIP archive can be specified in addition to the ID. As well as the documents to be compressed, the *archivename* tag must be used to specify the file name of the returned ZIP archive. A ZIP archive can contain up to 200 files. If one or more documents could not be processed, a ZIP will not be returned as a result. Instead, a JSON object will be returned containing the failed object IDs.

```
{
  "archivename" : "myArchive.zip",
}
```

```
"ids" : [
{
  "id": 14024030,
  "name": "file one"
},
{
  "id": 14024032,
  "name": "file two"
}]
}
```

[/osrest/api/documentfiles/addtotray](#)

§ Supported query methods: POST

§ Supported result formats: JSON

A file can be uploaded to the user tray with a multipart POST query (RFC 1867).

The file must contain the content-disposition `form-data` and additionally the `filename` attribute with the name of the file.

```
content-disposition: form-data; name="imagefile";
filename="image.jpg"
```

[/osrest/api/documentfiles/move/\[id\]?targetLocation=\[id of folder or register\]&sourceLocation=\[id of current folder/register\]](#)

§ Supported query methods: POST

The method moves the object to the location specified in "targetLocation" and therefore must be provided. If successful, "OK" will be returned, otherwise an exception will occur. If the object to be moved is a folder or register, "recursive" can be used to specify whether its contents should also be moved.

Optional parameters are:

- § recursive (Boolean): If the object is a folder or register, setting this parameter to 'true' will specify that its contents should also be moved.
- § sourceLocation (integer): Specifies the current location (direct parent object) of the object. Only required if the document has multiple locations.

[/osrest/api/documentfiles/droptargets](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The returned list contains the names of available droptargets.

Variables and data that are specified in a description area are also entered (see '<description>').

If between a variable and a field in enaio® there is a 1:1 mapping, the maximum field length for the enaio® field, the mandatory field status, and all possible catalog data will be returned. If no unambiguous mapping is possible (e.g. if variables are used several times), only the variable name will be returned.

```
{
  demo: {
    regtype: {
      catalog: [
        "Invoices",
        "Previous stock",
        "Miscellaneous",
        "Support info",
        "Project",
        "Milestone",
        "Correspondence",
        "Marketing",
      ],
      required: true,
      size: "15"
    },
    dkreditor: { },
    dstatus: { },
    fname: {
      required: true,
      size: "80"
    },
    gposition: {
      columns: [
        "Item number",
        "Name",
        "Price per unit",
        "Quantity",
        "Item subtotal"
      ],
      type: "GRID",
    }
  }
}
```

DropTarget scripts for specific users can be filed in enaio® appconnector. The scripts must be filed under the `droptargets` directory in the enaio® appconnector configuration, in a directory called `user/<username>`. For example, the directory structure for the user `demo` would look like this:

```
droptargets/users/demo
```

[/osrest/api/documentfiles/droptargets/\[targetname\]](/osrest/api/documentfiles/droptargets/[targetname])

§ Supported query methods: POST

§ Supported result formats: JSON

The method passes a file to a droptarget. The query must have the content type `multipart/form-data` (RFC 1867). The multipart query has to consist of two parts:

§ Variable data: This part must contain the content-disposition `form-data` and additionally the `name` attribute with the name `data`.

```
content-disposition: form-data; name="data"
```

§ **File upload:** This part must have the content disposition `form-data` and additionally the `filename` attribute with the file name. The `name` attribute is not important here.

```
content-disposition: form-data; name="imagefile";
filename="image.jpg"
```

Filling tables (type: GRID) requires you to pass the single lines with their column headers as a parameter. Empty columns can be ignored:

```
{
  fname: "0008-OS",
  regtype: "Project",
  dkreditor: "indat",
  gposition: {
    row_1: {
      ItemNumber: 123,
      Name: "Cat",
      PricePerUnit: 1000
    },
    row_2: {
      ItemNumber: 666,
      Name: "Mouse",
      PricePerUnit: 12,
      Quantity: 55
    }
  }
}
```

Notation 2.0:

```
"position": [ { "Type": "Item", "Number": "1" }, { .. } ]
```

The method returns the enaio® object ID of the created document. However, the exact return value depends on the setting in the DropTarget script itself.

</osrest/api/documentfiles/delete>

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to delete multiple objects from the archive. If successful, the method returns the string "true." The POST body of the query consists of a JSON array.

```
{
  "ids": [14118033, 14117993, 14117994, ....]
}
```

The IDs entered will then be returned in two arrays:

```
{
  "success": [
    14118033,
    14117993
  ],
  "failed": [
    14117994
  ]
}
```

[/osrest/api/documentfiles/delete/\[id\]](#)

§ Supported query methods: GET, DELETE

§ Supported result formats: JSON

Use this method to delete a document from the archive. If successful, the method returns the string 'true'.

[/osrest/api/documentfiles/update/\[id\]](#)

§ Supported query methods: POST

§ Supported result formats: -

Use this method to edit a document file. The query must have the content type `multipart/form-data` (RFC 1867). Multipart query structure:

§ File upload: This part must have the content disposition `form-data` and additionally the `filename` attribute with the file name. The `name` attribute is not important here.

```
content-disposition: form-data; name="imagefile";  
filename="image.jpg"
```

Optional parameters are:

§ `maintype (int)`: You can force a specific main type (see server API documentation).

§ `page (int)`: For multi-page document types (images), the transferred page can be specified here to avoid transferring all pages.

[/osrest/api/documentfiles/processmetadata/\[id\]\[?override=\[true|false\]&preview=\[true|false\]\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method can be used to supplement a document (ID) with data from the `ExtractionService`. The mapping defines which data are mapped.

Optional parameters are:

§ `override (bool)`: Specifies whether index fields should be filled with data even if they are not empty (optional). This option can also be set as a dedicated setting in the mapping.

§ `preview (bool)`: If true, instead of updating the document, the mapping will be returned as a "suggestion" formatted as JSON.

[/osrest/api/documentfiles/processmetadata/\[objectTypeId\]](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method can be used to supplement a document (ID) with data from the ExtractionService, and the document does not have to exist in enaio®. The file will be sent as a multipart POST request. The object type that the document is saved with is specified in the URL. An ExtractionMapping of the file is returned as JSON along with the mapping of the object type.

[/osrest/api/documentfiles/checkout/{id}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

Required parameters are:

§ objectType (int): ObjectId of the object

The method sets a document (ID) to the status 'checked out' for the user who is logged in via enaio® appconnector. If successful, HTTP 204 (No Content) will be returned. If the document has already been checked out by another user, that user's data will be returned in JSON format, along with an HTTP 409 (Conflict) status.

[/osrest/api/documentfiles/checkout/undo/{id}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

Required parameters are:

§ objectType (int): ObjectId of the object

The method sets a document (ID) that was checked out by the user who is logged in via enaio® appconnector to the status 'not checked out.' If successful, HTTP 204 (No Content) will be returned. If the document was checked out by another user, that user's data will be returned in JSON format, along with an HTTP 409 (Conflict) status.

NotificationService

[/osrest/api/notifications](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all notifications (subscriptions and follow-ups).

Optional parameters are:

§ shown (Boolean): Specifies whether self-addressed notifications are displayed (default: false).

§ starttime (int - Unix timestamp): Time from which notifications are to be displayed (default: no restriction).

- § metadata (string): File name of an alternative metadata mapping.
- § reload (Boolean): If true, caching will be ignored and the notifications will be reloaded from the server. The cache will be updated.
- § user (string): Name of a user for whom the notifications should be displayed. Assumes that the IP of the querying user has been entered in `osrest.properties`.
- § verbose (Boolean): An extended generic metadata model will be used additionally.
- § clienttype (string): Specifies the client type workflow for which notifications are received ("web," "mobile," "desktop," optionally with a specific language, e.g. "web_de," "web_en," "web_fr").

To use this feature for workflow notifications as well, workflow use has to be enabled in the configuration.

[/osrest/api/notifications/revisits](#)

- § Supported query methods: GET
- § Supported result formats: JSON

The method returns a list of all follow-up notifications.

Optional parameters are:

- § showown (Boolean): Specifies whether self-addressed notifications are displayed (default: false).
- § starttime (int - Unix timestamp): Point in time from which notifications are to be displayed (default: no restriction).
- § metadata (string): File name of an alternative metadata mapping.
- § verbose (Boolean): An extended generic metadata model will be used additionally.

[/osrest/api/notifications/subscriptions](#)

- § Supported query methods: GET
- § Supported result formats: JSON

The method returns a list of all notifications for subscribed objects.

Optional parameters are:

- § showown (Boolean): Specifies whether self-addressed notifications are displayed (default: false).
- § starttime (int - Unix timestamp): Point in time from which notifications are to be displayed (default: no restriction).
- § metadata (string): File name of an alternative metadata mapping.

§ verbose (Boolean): An extended generic metadata model will be used additionally.

[/osrest/api/notifications/revisits/remove/{id}/{eventDate}](#)

§ Supported query methods: GET

This method deletes the notification associated with a follow-up. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If successful, the function returns the value 0.

Required parameters:

§ id: ID of the object for which the notification will be removed.

§ eventDate: Date of the notification

[/osrest/api/notifications/remove](#)

§ Supported query methods: GET

Required parameters:

§ id: ID of the object for which the notification will be removed.

§ eventDate: Date of the notification

§ (optional – speeds up processing) notificationType: Type of the notification (REVISIT/SUBSCRIPTION)

[/osrest/api/notifications/revisits/markread/{id}/{eventDate}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method sets the status of a follow-up to `Read`. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If successful, the function returns the value 0.

[/osrest/api/notifications/revisits/markread](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method sets the status of follow-up notifications to `Read` or `Unread`. To do so, both the object ID {id} and the date {eventDate} must be specified in each case (both fields of a notification). If {reset} is used as a parameter, the notifications will be marked as unread.

If an error occurs during the "Mark as read" process, the method will return a list of notifications that could not be marked as read or unread.

Required parameters:

- § id: ID of the object
- § eventDate: Date of the notification
- § senderId: The trigger for the follow-up (optional)

Optional parameters:

- § reset: Mark notification as unread

POST example

```
[
  {
    "id": 4999,
    "eventDate": 1459760812000,
    "senderId": 8991
  },
  ...
]
```

[/osrest/api/notifications/revisits/process/{id}/{eventDate}](#)

§ Supported query methods: GET

The method sets the status of a notification follow-up to Processed or Unprocessed. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If {open} is used as a parameter, the notification will be marked as Unprocessed.

If successful, the function returns the value 0.

Required parameters:

- § id: ID of the object
- § eventDate: Date of the notification

Optional parameters:

- § reset: Mark notification as unprocessed
- § password: Base64-encoded password for your own account, if the follow-up requires password validation. If a password is required but not provided, the action will fail.

[/osrest/api/notifications/revisits/process](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method sets the status of follow-up notifications to Processed or Unprocessed. To do so, both the object ID {id} and the date {eventDate} must be specified in each case (both fields of a notification). If {reset} is used as a parameter, the notifications will be marked as Unprocessed. If an error occurs during the "Mark as processed"

process, the method will return a list of notifications that could not be marked as processed or unprocessed.

Required parameters:

- § id: ID of the object
- § eventDate: Date of the notification
- § senderId: The trigger for the follow-up (optional)

Optional parameters:

- § reset: Mark notification as unprocessed

</osrest/api/notifications/subscriptions/remove/{id}/{eventDate}>

§ Supported query methods: GET

This method deletes the notification associated with a subscription. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If successful, the function returns the value 0.

Required parameters:

- § id: ID of the object for which the notification will be removed.
- § eventDate: Date of the notification

Optional parameters:

- § senderId: The trigger for the follow-up. If this is not specified or 0, all notifications for the user of the object will be deleted.

</osrest/api/notifications/subscriptions/remove>

§ Supported query methods: POST

§ Supported result formats: JSON

This method deletes multiple notifications regarding subscriptions. To do so, both the object ID {id} and the date {eventDate} must be specified in each case.

If an error occurs during deletion, the method will return a list of notifications that could not be deleted.

Required parameters:

- § id: ID of the object for which the notification will be removed.
- § eventDate: Date of the notification
- § senderId: The trigger for the follow-up. If this is not specified or 0, all notifications for the user of the object will be deleted.

POST example

```
[
  {
    "id": 4999,
    "eventDate": 1459760812000,
```



```
        "senderId": 8991
      },
      ...
    ]
```

[/osrest/api/notifications/subscriptions/markread/{id}/{eventDate}](#)

§ Supported query methods: GET

The method sets the status of a subscription notification to Read or Unread. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If {reset} is used as a parameter, the notification will be marked as unread.

If successful, the function returns the value 0.

Required parameters:

§ id: ID of the object

§ eventDate: Date of the notification

Optional parameters:

§ reset: Mark notification as unread

[/osrest/api/notifications/subscriptions/markread](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method sets the status of subscription notifications to Read or Unread. To do so, both the object ID {id} and the date {eventDate} must be specified in each case (both fields of a notification). If {reset} is used as a parameter, the notification will be marked as unread. If an error occurs during the "Mark as read" process, the method will return a list of notifications that could not be marked as read or unread.

Required parameters:

§ id: ID of the object

§ eventDate: Date of the notification

§ senderId: The trigger for the follow-up (optional)

Optional parameters:

§ reset: Mark notification as unread

POST example

```
[
  {
    "id": 4999,
    "eventDate": 1459760812000,
    "senderId": 8991
  },
  ...
]
```

```
] ...
```

[/osrest/api/notifications/subscriptions/process/{id}/{eventDate}](#)

§ Supported query methods: GET

The method sets the status of a subscription notification to Processed. To do so, both the object ID {id} and the date {eventDate} must be specified (both fields of a notification). If successful, the function returns the value 0.

Required parameters:

§ id: ID of the object

§ eventDate: Date of the notification

§ password: Base64-encoded password for your own account, if the subscription requires password validation. If a password is required but not provided, the action will fail.

§ confirmed: true|false. If the subscription requires confirmation, true must be used here. If false is used in such a case, the action will fail.

[/osrest/api/notifications/subscriptions/process](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method sets the status of follow-up notifications to Processed or Unprocessed. To do so, both the object ID {id} and the date {eventDate} must be specified in each case (both fields of a notification). If an error occurs during the "Mark as processed" process, the method will return a list of notifications that could not be marked as processed or unprocessed.

Required parameters:

§ id: ID of the object

§ eventDate: Date of the notification

§ senderId: The trigger for the follow-up (optional)

POST example

```
[
  {
    "id": 4999,
    "eventDate": 1459760812000,
    "senderId": 8991
  },
  ...
]
```

</osrest/api/notifications/subscriptionQueries>

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns all of the user's subscribed queries See below for an explanation of the constants for "confirm," "notifyType," and "actions."

```
[
  {
    infoText: "For admin group",
    confirm: "NO_CONFIRMATION",
    aboGroup: "1263CA058C1841A5B5BD0F07C77BA901",
    notifyType: "INTERNAL",
    actions: [
      "INDEXDATA_MODIFIED"
    ]
  }
  ...
]
```

</osrest/api/notifications/subscriptionObjects>

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all of the user's subscribed objects.

Actions can have the following values:

Key	Description
INDEXDATA_MODIFIED	The index data were changed.
DOCUMENT_MODIFIED	A document was added to the object or it was modified.
OBJECT_CREATED	An object was created in the folder/register.
OBJECT_DELETED	An object was deleted from the folder/register.
OBJECT_ADDED	An object was added to the folder/register from another cabinet.
OBJECT_MOVED_FROM_TRAY	An object was added to the folder/register from the user tray.
LOCATION_ADDED	A object in the folder/register was copied to another location.

Confirm can have the following values:

Key	Description
NO_CONFIRMATION	No confirmation required.

CONFIRMATION	Confirmation is required before the notification can be deleted.
CONFIRMATION_PASSWORD	Confirmation with password input is required before the notification can be deleted.
CONFIRMED	Confirmation has been received.

NotifyType can have the following values:

Key	Description
INTERNAL	enaio® client notification.
E-MAIL	E-mail notification.

```
[{
  "objectId": 1235,
  "infoText": "Free text when creating a subscription",
  "confirm": "NO_CONFIRMATION",
  "aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",
  "notifyType": "INTERNAL",
  "actions": [
    "OBJECT_DELETED"
  ],
  "groupsToBeNotified": [
    {
      "id": 8522,
      "name": "PERSONAL"
    },
    ...
  ],
  "usersToBeNotified": [
    {
      "id": 77,
      "name": "DEMO",
      "fullname": "John Doe"
    },
    ...
  ]
},
...
]
```

</osrest/api/notifications/subscriptionObjects/{id}>

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all notifications set up for the user that relate to a given object. See above for an explanation of the constants for "confirm," "notifyType," and "actions."

```
{
  "objectId": 1235,
  "infoText": "Free text when creating a subscription",
```

```

"confirm": "NO_CONFIRMATION",
"aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",
"notifyType": "EMAIL",
"actions": [
    "OBJECT_DELETED"
],
"groupsToBeNotified": [
    {
        "id": 8522,
        "name": "PERSONAL"
    },
    ...
],
"usersToBeNotified": [
    {
        "id": 77,
        "name": "DEMO",
        "fullname": "John Doe"
    },
    ...
]
}

```

[/osrest/api/notifications/subscriptionObjects](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to add or modify a subscription to the object with the specified ID. The subscription to be added/modified must be submitted as POST in JSON format, exactly as it was obtained via the GET method of the same name. If a new subscription should be created, the aboGroup entry must be left empty. If an existing subscription is to be modified, its aboGroup must be included here. If successful, the method will return 0, otherwise it will return an error. See above for an explanation of the constants for "confirm," "notifyType," and "actions."

```

{
    "objectId": 1235,
    "infoText": "",
    "confirm": "NO_CONFIRMATION",
    "aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",
    "notifyType": "EMAIL",
    "actions": [
        "OBJECT_DELETED"
    ],
    "mailAddresses": [
        "demo_deleted222@demo.optimal-systems.de"
    ],
    "groupsToBeNotified": [
        {
            "id": 8522,
            "name": "PERSONAL"
        },
        ...
    ],
    "usersToBeNotified": [
        {
            "id": 77,
            "name": "DEMO",
            "fullname": "John Doe"
        }
    ]
}

```

```
    },  
    ...  
  ]  
}
```

[/osrest/api/notifications/subscriptionMultiObjects](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to add or modify multiple subscriptions to objects. The subscriptions to be added/modified must be submitted as POST in JSON format, exactly as they were obtained via the GET method of the same name. As this method can modify multiple subscriptions at once, the top JSON level is an array of the JSON subscription objects. *aboGroup* is left empty for new subscriptions, and filled in for subscriptions to be modified. New subscriptions and subscriptions to be modified can both be specified in a single POST JSON array. The method will return an array of all failed objects. See above for an explanation of the constants for "confirm," "notifyType," and "actions."

```
[ {  
  "objectId": 1235,  
  "infoText": "",  
  "confirm": "NO_CONFIRMATION",  
  "aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",  
  "notifyType": "EMAIL",  
  "actions": [  
    "OBJECT_DELETED"  
  ],  
  "mailAddresses": [  
    "demo_deleted222@demo.optimal-systems.de"  
  ],  
  "groupsToBeNotified": [  
    {  
      "id": 8522,  
      "name": "PERSONAL"  
    },  
    ...  
  ],  
  "usersToBeNotified": [  
    {  
      "id": 77,  
      "name": "DEMO",  
      "fullname": "John Doe"  
    },  
    ...  
  ]  
},  
...  
]
```

[/osrest/api/notifications/subscriptionObjects/delete/{id}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

Use this method to delete a subscription. The *aboGroup* of the subscription must be provided as the ID.

If successful, the method will return 0, otherwise it will return an error.

</osrest/api/notifications/subscriptionObjects/delete>

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to delete multiple subscriptions to objects. The JSON in the POST body is a string array containing the *aboGroup* elements to be deleted. The method returns all *aboGroup* elements for which the associated subscription could not be deleted.

```
[
  7cf2ebaf2f7a451f8014a35d397996fd,
  7cf2ebaf2f7b451f8014a35d397996fe,
  7cf2ebaf2f7c451f8014a35d397996ff,
  ...
]
```

</osrest/api/notifications/revisitObjects>

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all of the user's follow-ups.

```
{
  "osid": 14008672,
  "duedate": 1433847600000,
  "subject": "",
  "creationdate": 1433847417000,
  "notified": false,
  "confirm": false,
  "notifiedbyemail": false,
  "sender":
  {
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  "recipients": [
    {
      "id": 77,
      "name": "DEMO",
      "fullname": "John Doe"
    }
  ]
}
```

</osrest/api/notifications/revisitObjects/{id}>

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all follow-ups set up for the user that relate to a given object.

```
{
  "osid": 14008672,
  "duedate": 1433847600000,
  "subject": "",
  "creationdate": 1433847417000,
  "notified": false,
  "confirm": false,
  "notifiedbymail": false,
  "sender": {
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  "recipients": [
    {
      "id": 77,
      "name": "DEMO",
      "fullname": "John Doe"
    },
    ...
  ]
}
```

[/osrest/api/notifications/revisitObjects](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to add or modify a follow-up. The follow-up to be added/modified must be submitted as POST in JSON format, exactly as it was obtained via the GET method of the same name.

The recipient structure in the JSON must be duplicated and added as *newRecipients*. Only those users listed in *newRecipients* will be included in the new or modified follow-up. If a new follow-up is to be created, no *creationdate* can be provided. However, if an existing follow-up is to be modified, its *creationDate* must be specified. If successful, the method will return 0, otherwise it will return an error.

```
{
  "osid": 14008672,
  "duedate": 1433847600000,
  "subject": "",
  "creationdate": 1433847417000,
  "notified": false,
  "confirm": false,
  "notifiedbymail": false,
  "notificationmail": "E-mail address if notifiedbymail is true",
  "sender": {
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  "recipients": [{
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  {
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  }
]
```



```
        "fullname": "John Doe"
      },
      ...
    ],
    "newRecipients": [{
      "id": 77,
      "name": "DEMO",
      "fullname": "John Doe"
    },
    ...
  ]
}
```

[osrest/api/notifications/revisitMultiObjects](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to add or modify multiple follow-ups for objects. The follow-ups to be added/modified must be submitted as POST in JSON format, exactly as they were obtained via the GET method of the same name. As this method can modify multiple follow-ups at once, the top JSON level is an array of the JSON follow-up objects. *creationDate* is left empty for new follow-ups, and specified for follow-ups to be modified. New follow-ups and follow-ups to be modified can both be specified in a single POST JSON array. The method will return an array of all failed objects.

```
[{
  "osid": 14008672,
  "duedate": 1433847600000,
  "subject": "",
  "creationdate": 1433847417000,
  "notified": false,
  "confirm": false,
  "notifiedbyemail": false,
  "notificationmail": "E-mail address if notifiedbyemail is true",
  "sender": {
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  "recipients": [{
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  ...
],
  "newRecipients": [{
    "id": 77,
    "name": "DEMO",
    "fullname": "John Doe"
  },
  ...
]
},
...
]
```

[/osrest/api/notifications/revisitObjects/delete/{id}?userId=X&dueDate=Y](#)

§ Supported query methods: GET

§ Supported result formats: JSON

Use this method to delete a follow-up. The *aboGroup* of the subscription must be provided as the ID.

If successful, the method will return 0, otherwise it will return an error.

Required parameters are:

§ *userId* (int): Defines the user (recipient) whose follow-up will be deleted.

§ *dueDate* (long – Java timestamp): *dueDate* value of the follow-up.

[/osrest/api/notifications/revisitObjects/delete](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to delete multiple follow-ups. At the top level, the POST body is a JSON array containing the follow-ups to be deleted. The respective *userId* is the ID of the user that the follow-up is set up for.

```
[ {
  "osid": 14008672,
  "duedate": 1433847600000,
  "userId": 12345
},
...
]
```

[/osrest/api/notifications/workflows](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all notifications on workflow events.

Optional parameters are:

§ *starttime* (int - Unix timestamp): Point in time from which notifications are to be displayed (default: no restriction).

§ *metadata* (string): File name of an alternative metadata mapping.

§ *clienttype* (string): Specifies the client type workflow for which notifications are received ("web," "mobile," "desktop," optionally with a specific language, e.g. "web_de," "web_en," "web_fr").

§ *verbose* (Boolean): More detailed data on the workflows will be returned.

To use this feature, workflow use must be enabled in the configuration.

SessionService

/osrest/api/session

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns information on the enaio® user session.

§ server: enaio® server address

§ port: enaio® server port

§ username: Name of the user who is logged in.

§ osguid: enaio®-internal GUID of the user who is logged in.

§ sessionGuid: GUID of the current user session on enaio® server.

```
{
  server: "192.168.0.1"
  port: 4000
  username: "demo"
  osguid: "648F3878205E4FF8BD08B9A4C96EDDF1"
  sessionGuid: "B982870084354AA99768C7617B0135B9"
}
```

/osrest/api/session/login

§ Supported query methods: POST

§ Supported result formats: JSON

A form-based login is possible with this method.

To this end, the following parameters must be passed in a POST request with the content type application/x-www-form-urlencoded:

§ osrest_username

§ osrest_password

The method returns information on the enaio® user session (see /osrest/api/session).

/osrest/api/session/logout

§ Supported query methods: GET

§ Supported result formats: JSON

It is possible to end the current user session (log out) with this method.

/osrest/api/session/changePassword

§ Supported query methods: POST

§ Supported result formats: HTTP 204 / No Content

Users may change their own password with this method.

The following JSON must be included in the POST body:

```
{
  "oldPassword": "Old password encoded as Base64",
  "newPassword": "New password encoded as Base64"
}
```

[/osrest/api/session/checklicense/\[license module\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

Use this method to check whether enaio® license modules are available on the server. The method returns a simple JSON string with *true* or *false*. If case of doubt required license modules must be assigned to the OSRest workstation.

[/osrest/api/session/runscript](#)

§ Supported query methods: POST

§ Supported result formats: JSON

Use this method to run a VB script on the server.

Parameters and the name of a script file in the configuration subdirectory *scripts* are provided here in a POST request. If only parameters in a server-side event need to be evaluated, the *script* parameter can be omitted.

```
{
  "parameters": {
    "RecordNumber": "4711",
    "GetLastVersion": true
  },
  "script": "example.vbs"
}
```

The call returns the individual return parameters of the job:

```
{
  RecordNumber: "4711",
  MaxVersion: 4,
  DocumentCount: 6
}
```

[/osrest/api/session/user/groups](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all groups in the system.

```
[
```

```
"EDEKA",  
"STANDARD",  
"DEMO",  
"PERSONAL",  
"NAVISON",  
"CRM",  
"FSA",  
"QMS-READ",  
"PRESSE",  
"GEVER"  
]
```

[/osrest/api/session/user/users](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns a list of all users in the system.

```
{  
  "id": 14024926,  
  "name": "SCHMIDT",  
  "fullname": "Sebastian Schmidt"  
}
```

[/osrest/api/session/userdesktops](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the private and public desktop in tree form.

Optional parameters are:

§ refresh (Boolean): If this parameter is set to 'true,' the cache will be ignored and the queries will be requested from the server again.

[/osrest/api/session/userdesktops/add](#)

§ Supported query methods: POST

§ Supported query format: JSON

Use this method to add folders and references to DMS objects to the current user's private desktop. If no "parentId" is specified, the object will be added at the top level of the private desktop.

The "objectTypeId" for DMS objects is optional. However, it is recommended to include it for performance reasons.

Adding folders

```
{  
  "name": "Customer folder",  
  "parentId": "26279936",  
  "type": "FOLDER"
```

```
}
```

Adding a DMS object

```
{
  "name": "Contract record 12",
  "id": "4570",
  "objectTypeId" : "131082",
  "parentId": "26279936",
  "type" : "OBJECT"
}
```

[/osrest/api/session/userdesktops/remove](#)

§ Supported query methods: POST

§ Supported query format: JSON

Use this method to remove folders, search requests, and references to DMS objects from the current user's private desktop.

Removing a folder

```
{
  "folderId": "26279941",
  "parentId": "26279936",
  "type": "FOLDER"
}
```

Removing a saved search

```
{
  "name": "1",
  "type": "QUERY"
}
```

Removing a reference to a DMS Object

```
{
  "name": "1",
  "type": "OBJECT"
}
```

ServiceInfoService

[/osrest/api/serviceinfo](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns information about the API version and build revision of the OSREST service. This information will be requested for support cases. This service will also display the capabilities of the installation (see 'Capabilities of enaio® appconnector').

```
{
  apiVersion: "1.1.0"
  buildRevision: "5168"
}
```

[/osrest/api/serviceinfo/ping](#)

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns the time on the OSREST server as a timestamp. The feature is mainly used to test the accessibility of the service and the authentication.

[/osrest/api/serviceinfo/errorTypes/](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns all error types known by the API.

[/osrest/api/serviceinfo/errorTypes/{errorCode}](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns a single error type.

```
{
  "ErrorType": "UNSPECIFIC_ERROR",
  "HttpStatusCode": "Internal Server Error - [500]",
  "DefaultMessage": "It looks like an internal failure occurred -
this should not have happened! :",
  "ErrorCode": 0
}
```

OSFileService

[/osrest/api/anon/osfile/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: application/octet-stream

The method creates an internal enaio® link file for the object with the specified ID. This service can be used without authentication.

The optional query parameter `followactivevariant=true` redirects the link to an active variant of the object, if it refers to an inactive variant.

WorkflowService

[/osrest/api/workflows](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns a list of workflows that can be started by the user.

Optional parameters are:

- § **clienttype** (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

Result:

```
{
  id: "4A3D7DE0352B496F9C4CD8565479B1B3",
  title: "Inbox (ad-hoc)",
  info: "Version 3.0",
  workflowParameters: [
    {
      readonly: false,
      type: "TEXT",
      id: "783241C856A04D17AEE520C9630CFDA1",
      name: "sApplicant",
      value: ""
    },
    ...
  ],
  ...
}
```

</osrest/api/workflows/start>

§ Supported query methods: POST/JSON

This method starts a workflow.

Input: An entry from the results of (</osrest/api/workflows>), with adapted values (value) if necessary.

Optional parameters are:

- § **clienttype** (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

```
{
  "id": "16B30DD524614774A64C26346B7A3B01",
  "workflowParameters": [
    {
      "type": "TEXT",
      "id": "55C31B02940B4FC98772C583DD627AC5",
      "name": "sOutputParameter",
      "value": "output",
      "readonly": false
    },
    {
      "type": "LIST",
      "id": "45319353B64E47F0B18CC4F0581B24BE",
      "name": "sInputParameter",
      "value": "Inbox",
      "readonly": false
    }
  ],
}
```



```
{
  {
    "type": "TEXT",
    "id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
    "name": "sRelease",
    "value": "false",
    "readonly": false
  }
},
"files": [
  "4133"
]
}
```

[/osrest/api/workflows/startWithData](#)

§ Supported query methods: POST/JSON

This method starts a workflow, thus putting the files in the workflow filing tray. The workflow is started and files uploaded in the workflow filing tray using a multipart POST request (RFC 1867). These must contain the content disposition 'form-data' and the `filename` attribute with the file name. The workflow is passed as text to a 'data' content disposition in JSON format (see `/osrest/api/workflows/start`).

Optional parameters are:

§ `clienttype` (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

[/osrest/api/workflows/running](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns a list of all work items that the user can access. The parameters are those that were configured for display in the inbox.

Optional parameters are:

§ `clienttype` (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

§ `verbose` (Boolean): More detailed data on the work items will be returned.

Short result:

```
[
  {
    id: "ABE6D05571CF40968F8447C1556D33CA",
    processID: "0B938F1389DD4877BD178DD1B83A2EBC",
    title: "Initialization - Task: Take note",
    info: "Comment: This task still needs to be completed.",
    creationTime: 1317038062000,
  }
]
```

```
        personalized: "MEIER",
        substitute: false,
        read: false
    },
    ...
]
```

Detailed result (verbose):

```
[
  {
    id: "ABE6D05571CF40968F8447C1556D33CA",
    processID: "0B938F1389DD4877BD178DD1B83A2EBC",
    iconId: "1073743021",
    activityName: "Initialization",
    processName: "Default ad hoc workflow 27",
    processSubject: "Task: Take note",
    creationTime: 1465635668000,
    personalized: "MEIER",
    read: false,
    substitute: false,
    overTime: true,
    warningTime: 1465635668000,
    workflowParameters: [
      {
        type: "TEXT",
        name: "Comment",
        value: "This task still needs to be completed.",
        position: 0
      }
    ]
  },
  ...
]
```

[/osrest/api/workflows/running/full/\[id\]](/osrest/api/workflows/running/full/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

The method returns all the XML files of a WorkItem in JSON format. These are: ExtendedAttributes, Parameters, Version, WorkflowType, File, Masks, and ProcessResponsible.

Parameter:

§ id: WorkItem ID

Optional parameters are:

§ refresh (Boolean): Ignores the cache and retrieves WorkItem data again.

§ personalize (Boolean): The WorkItem will also be personalized when opened.

§ clienttype (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

Result:

```

{
  "WorkItem": {
    "ExtendedAttributes": {
      "ExtendedAttribute": [
        {
          "Value": "0",
          "Name": "CHECK_PASSWORD"
        },
        ...
      ]
    },
    "Parameters": {
      "Parameter": [
        {
          "WFVar": {
            "Types": { },
            "IntegerSafe": "0"
          },
          "DataField": "49623478004E4419BF976B182871D674",
          "Selection": "",
          "ListType": "",
          "Mode": "3",
          "FormField": "0F8D00DCF1F14649BE9E407632F415E7",
          "InfoText": "",
          "Name": "intNumber"
        },
        ...
      ]
    },
    "Version": "31",
    "WorkflowType": "1",
    "File": {
      "Lists": {},
      "Docs": {}
    },
    "Masks": {
      "Mask": {
        "MaskFields": {
          "MaskField": [
            {
              "ToolTip": "This is a mandatory
field!|",
              "FieldTop": "100",
              "RegularExpression": {},
              "InpRight": "80",
              "InpBottom": "12",
              "Flags": "1",
              "InternalName": "",
              "FieldBottom": "12",
              "Name": "MandatoryField",
              "ValuesId": "",
              "TabOrder": "0",
              "Init": "",
              "FieldRight": "30",
              "FieldLeft": "45",
              "Flags1": "0",
              "InpLeft": "80",
              "DataType": "X",
              "InpLen": "50",
              "Flags2": "0",
              "Id":
"833FD8E7FD614083A31DB9BBEB37A94D",
              "InpTop": "100"
            }
          ]
        }
      }
    }
  }
}

```

```
        },
        ...
    ]
    }
    },
    "FrameWidth": "428",
    "FrameHeight": "250",
    "Id": "00F51E9EB65141BD92601F625E61E42C",
    "Flags": "0",
    "Name": "m"
  }
},
"ProcessResponsible": "1"
}
```

[/osrest/api/workflows/running/\[id\]](/osrest/api/workflows/running/[id])

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns all parameters and files that belong to a workflow activity.

Parameter:

§ id (string): ID of the workflow activity

§ personalize (Boolean): The WorkItem will also be personalized when opened.

§ clienttype (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

§ verbose (Boolean): Supplements the returned data with more detailed information about record elements.

Result:

```
{
  id: "ABE6D05571CF40968F8447C1556D33CA",
  workflowParameters: [
    {
      "readonly": false,
      "type": "CHECKBOX",
      "id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
      "name": "A check box",
      "value": "0",
      "required": false
    },
    {
      "readonly": false,
      "type": "RADIO",
      "id": "0F665AAAA1C34569B6863C3F73498204",
      "name": "completed",
      "value": "3",
      listData: [
        "RadioButton1",
        "RadioButton2"
      ],
      "required": false
    }
  ]
}
```

```

    },
    {
      "readonly": false,
      "type": "TEXT",
      "id": "EFF64844A6A241169EE8AB19DC05886A",
      "name": "RegExp field",
      "value": "",
      "regularExpression":
"Value\\{(.*)\\}\\|Message\\{(.*)\\}",
      "required": false
    },
    {
      "readonly": true,
      "type": "DATE",
      "id": "88CA29EB04E54377AC8D3C788CE90906",
      "name": "Date readonly",
      "value": "",
      "required": true
    }
  ],
  files: [
    "1452"
  ]
}

```

Result (verbose):

```

{
  id: "ABE6D05571CF40968F8447C1556D33CA",
  workflowParameters: [
    {
      "readonly": false,
      "type": "CHECKBOX",
      "id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
      "name": "A check box",
      "value": "0",
      "required": false
    },
    ...
  ],
  files: [
    "1452"
  ],
  verboseFiles: [
    {
      id: "1452",
      objectType: "131081",
      location: "1",
      workspace: "1",
      deletable: true,
      useActiveVariant: false,
      moveable: true,
      sig: "0",
      rights: "15",
      originalId: "1452",
      display: "1"
    }
  ]
}

```

[/osrest/api/workflows/personalize](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method personalizes WorkItems for the user who is logged in. To this end, the ID of each WorkItem must be specified. If an error occurs during personalization, the method will return a list of WorkItems that could not be personalized.

Required parameters:

§ id (string): WorkItem ID

Optional parameters:

§ clienttype (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

```
[
  {
    "id": "1F93E705A4FB46C3B237CC582FD9BFE1"
  },
  ...
]
```

[/osrest/api/workflows/depersonalize](#)

§ Supported query methods: POST

§ Supported result formats: JSON

The method removes personalization from the specified WorkItems for the user who is logged in. To this end, the ID of each WorkItem must be specified. If an error occurs when removing personalization, the method will return a list of WorkItems from which the personalization could not be removed.

Required parameters:

§ id (string): WorkItem ID

Optional parameters:

§ clienttype (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

```
[
  {
    "id": "1F93E705A4FB46C3B237CC582FD9BFE1"
  },
  ...
]
```

[/osrest/api/workflows/forward](#)

§ Supported query methods: POST/JSON

Optional parameters:

- § **clienttype** (string): Specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

This method forwards a workflow activity. It requires the JSON result of `/osrest/api/workflows/running/[id]` to be passed (with adapted parameter values if required).

POST example

```
{
  id: "A568441085B74CBDB088C7B43838AAEA",
  workflowParameters: [{
    "type": "TEXT",
    "id": "2A1B85B652A5423BAE78CA0C2FBDAA6D",
    "name": "MandatoryField",
    "value": "modified text"
  }, ...
],
files: ["1452"],
(optional) verboseFiles: [
  {
    id: "1452",
    workspace: 0
  }
]
}
```

Making changes to the workflow file§ **Adding a DMS object to the record****POST example, adding a new file element**

```
{
  id: "A568441085B74CBDB088C7B43838AAEA",
  workflowParameters: [],
  files: [],
  verboseFiles: [
    {
      id: "1527",
      objectType: "262144",
      location: "1",          (1 - DMS object exists in file
system, 2 - DMS object only exists in the system tray)
      workspace: "0",        (0 - Info area of the record, 1
- Workspace of the record)

      (optional)
      deletable: false,
      useActiveVariant: false,
      moveable: false
    }
  ]
}
```

§ **Editing a record element****POST example, editing a record element**

```
{
  id: "A568441085B74CBDB088C7B43838AAEA",
  workflowParameters: [],
```

```

    files: ["1527"],
    verboseFiles: [
      {
        id: "1527",

        (only the property that was modified is required in
each case)
        workspace: "0",      (0 - Info area of the record, 1
- Workspace of the record)
        deletable: false,
        useActiveVariant: false,
        moveable: false
      }
    ]
  }
}

```

§ Deleting a DMS object from the record

To delete a DMS object from the record, the ID of the relevant object must be deleted from the "files" property.

[/osrest/api/workflows/cancel](#)

§ Supported query methods: POST/JSON

Optional parameters:

- § save: true|false. If 'true,' the data included will be saved.
- § clienttype (string): If save:true, then specifies the client type for which the workflow form will be internally loaded for saving ("web," "mobile," "desktop," or forms in a specific language e.g. "web_de," "web_en," or "web_fr").

Use this method to save a workflow activity with new values without forwarding it. It requires the JSON result of [/osrest/api/workflows/running/\[id\]](#) to be passed (with adapted parameter values if required) (see forward call).

[/osrest/api/workflows/processes/\[Id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method provides all running workflow processes of an object.

Parameter:

- § id (int): OSID of the object

Result:

```

[
  {
    "id": "D44ACCBDCD14481ABA425E383501CD83",
    "name": "Test Mobile 0.8 188",
    "subject": "Test workflow for enaio app",
    "state": "RUNNING",
    "creatorId": "217F716436F04D85AFCDE61F7192DD7D",
  }
]

```



```
"creationTime": 1392730091000,
"processResponsible": false,
"activities": []
},
{
  "id": "CF51FCD484F246A6A02898F1522FC3E3",
  "name": "Test Mobile 0.8 185",
  "subject": "Test workflow for enaio app",
  "state": "COMPLETED",
  "creatorId": "217F716436F04D85AFCDE61F7192DD7D",
  "creationTime": 1392281546000,
  "processResponsible": false,
  "activities": [
    "Activity Client",
    "Activity mobileDMS",
    "Activity mobileDMS 2"
  ]
}
]
```

</osrest/api/workflows/abort>

§ Supported query methods: POST/JSON

This method aborts one or more workflow processes.

Input: Either the process ID of the workflow instance or an OSID of a file document. In the latter case, all workflow instances that contain this file document will be aborted. This means that more than one workflow instance may be aborted! First, the processID is evaluated. If it is present, the osID will be ignored. If no processID is present, the osID will be evaluated.

```
{
  "processID": "07D8A5CC07D94BDC9604EA6377085891",
  "osID": "1532"
}
```

[/osrest/api/workflows/absence/\[true|false\]](/osrest/api/workflows/absence/[true|false])

§ Supported query methods: GET

§ Supported result formats: JSON

This method subscribes or unsubscribes the current user to/from workflows.

Input: If *true* is passed as the final path element, the current user will be unsubscribed from workflows and will no longer receive workflows via the notification call. Conversely, *false* as the final path element will subscribe the user to workflows again. The following JSON result, for example, will be returned for the path element *true*. The current status is returned in the result key *wfAbsence*.

```
{
  "null": {
    "id": "AA772557EC874AF48B229B8D4832D61B",
    "wfAbsence": "true"
  }
}
```

ObjDevService

[/osrest/api/objdef/full](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns the full object definitions in JSON format. Depending on size, this call can take a while to complete and produce quite extensive results. The structure is equivalent to XML, only transformed to JSON.

[/osrest/api/objdef/languages](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns the languages defined in the object definition in JSON format. Depending on size, this call can take a while to complete and produce quite extensive results. The structure is equivalent to XML, only transformed to JSON.

[/osrest/api/objdef/search/\[id\]](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method expects an object type ID from a folder, register, or document, and returns this part of the object definition in JSON format. The structure of the JSON is equivalent to XML.

Parameter:

§ id: ObjectTypeId of the object

Optional parameters:

§ refresh: Retrieves the object definitions again

JSON content

```
{
  "object": {
    "IconID": "1073742179",
    "compressionflags": "0",
    "cotype": "55",
    "extablename": "",
    "fields": { "field": [
      {
        "classstring": "",
        "field_pos": {
          "bottom": "398",
          "left": "286",
          "right": "229",
          "top": "26"
        }
      }
    ]
  }
}
```

```
}
```

[/osrest/api/objdef/search](#)

§ Supported query methods: POST

§ Supported result formats: JSON

This method expects one or more object type IDs from a folder, register, or document, and returns these parts of the object definition in JSON format. The structure of the JSON is equivalent to XML.

Optional parameters:

§ refresh: Retrieves the object definitions again

POST example:

```
[
  { "id": 22 }, { "id": "413" }, ...
]
```

Result:

```
{
  "objectTypes": [
    {
      "internal": "addresses",
      "cotype": "2",
      ...
    },
    ...
  ]
}
```

OrganizationService

[/osrest/api/organization/users](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns a list of all ECM users.

[/osrest/api/organization/groups](#)

§ Supported query methods: GET

§ Supported result formats: JSON

This method returns a list of all ECM groups that the current user is a member of or, if the 'all' parameter is passed, all ECM groups in the system.

Optional parameters are:

§ all (Boolean): All groups in the system will be returned (default: false).

§ **loadUsers (Boolean):** The members (users) of each group will be returned.

[/osrest/api/organization/sendmail](#)

§ Supported query methods: POST

§ Supported result formats: JSON

This method can be used to send an email. A JSON mail object must be passed as the POST body.

```
{
  "receiver": "recipient@optimal-systems.de",
  "sender": "sender@optimal-systems.de",
  "subject": "E-mail title",
  "text": "E-mail content"
}
```

[/osrest/api/organization/avatar/{username}](#)

§ Supported query methods: GET

§ Supported result formats: Image/*.*

This method returns the avatar picture of the user with the specified user name for download. The avatar will be returned in size 80px.

[/osrest/api/organization/avatar/{username}/{size}](#)

§ Supported query methods: GET

§ Supported result formats: Image/*.*

This method returns the avatar picture of the user with the specified user name for download. The avatar will be returned in the size specified with *size*.

[/osrest/api/organization/user/updateCreate](#)

§ Supported query methods: POST

§ Supported result formats: JSON

This method can be used to create an ECM user or modify the data of an existing ECM user.

```
{
  "id": 0,
  "name": "USER NAME",
  "fullname": "John Doe",
  "password": "password",
  "description": "This user has this free text description",
  "locked": 0,
  "email": "max@mustermann.de"
}
```

```
}
```

Supervisor rights are required to run this method. If 0 is passed as the ID, the user will be created. If ID > 0, the relevant user will be updated. It is not possible to update the user name.

[/osrest/api/organization/user/delete/{id}](#)

§ Supported query methods: GET

This method deletes the ECM user with the specified ID. Supervisor rights are required to run this method. Supervisor users cannot be deleted.

Optional parameters are:

- § transferPortfolios (Boolean): The portfolios of the user to be deleted can be transferred to another user.
- § transferNotifications (Boolean): The subscriptions and follow-ups of the user to be deleted can be transferred to another user.
- § transferUserId(Integer): The user ID of the ECM user to whom the portfolios and/or subscriptions and follow-ups should be transferred.

IconService

Catalog icons for hit list objects can be retrieved via the IconService. The "iconId" JSON attribute exists within hit lists for this purpose, and specifies the ID of an object's catalog icon.

[/osrest/api/icon/preload](#)

§ Supported query methods: POST

§ Supported result formats: NONE

The method loads a list of catalog icons from enaio® server. This means that enaio® appconnector does not need to retrieve each icon from enaio® server individually when using icon/{id} later on. If successful, only an HTTP 204 status code (success, no content) will be returned, without content. In the event of an error, the HTTP status code will vary accordingly.

```
{
  "iconIds": [
    1073741986,
    1073741987,
    1073741988,
    1073742158,
    1073742242
  ]
}
```

[/osrest/api/icon/{id}](#)

§ Supported query methods: GET

§ Supported result formats: Image/gif

This method returns the catalog icon with the specified ID for embedding in an `<img...>` tag. For multiple icons in hit lists, it is recommended to load them in advance using the preload method.

Results

OSRest returns three types of results:

§ Hit Lists

§ Notification Lists

§ Saved queries

Hit Lists

```
{
  -
  documentResult: {
    pagesize: 500
    startPosition: 0
    totalHits: 18

    -
    documents: [
      -
      {
        id: "1452"
        type: "FOLDER"

        -
        fields: {
          title: "Stadtwerke Jena GmbH"
          info: "Göschwitzer Str. 22 7745 Jena"
        }
        fav: false
      }, ...
    ]
  }
}
```

`documentResult` is the root element of a hit list. It contains information such as `pagesize`, `startPosition` (offset), and `totalHits`, as well as a list of `documents`.

`documents` represents the hits from enaio®. A document has the attributes `id`, `type`, `fields`, and `fav`.

id: OSID of the hit

type: The document type of a hit. Possible values: FOLDER, REGISTER, DOCUMENT.

fields: List of the hit's mapped index data (see metadata mapping)

fav: Flag indicating whether a hit is in the favorites folder.

Notification Lists

```
notifications: [
  -
  {
    id: "9015"
    type: "DOCUMENT"
    notificationType: "REVISIT"
    eventDate: 1282654839000
    -
    fields: {
      title: "Hugo Distler"
      info: ""
    }
  }, ...
]
```

notifications is the root element of the notification list. It contains a list of notifications.

A notification contains the information **id**, **type**, **notificationType**, **eventDate**, and **fields**.

id: OSID of the hit

type: The document type of a hit. Possible values: FOLDER, REGISTER, DOCUMENT.

notificationType: Notification type. Possible values: REVISIT, SUBSCRIPTION, WORKFLOW.

eventDate: Date of the notification

fields: List of the hit's mapped index data (see metadata mapping)

Saved queries

```
{
  -
  storedqueries: [
    -
    {
      id: 14004695
      name: "Open support calls"
      queryParams: [ ]
    }
    -
    {
      id: 14041962
      name: "E-mails"
      -
      queryParams: [
        -
        {
          object: "Email"
          field: "Date:"
          dataType: "DATE"
        }
      ]
    }
  ], ...
}
```

storedqueries is the root element. It contains a list of saved queries.

A saved search contains the information **id**, **name**, and **queryParams**.

id: OSID of the saved search

name: Name of the saved search

queryParams: List of parameters of the saved search (if available)