OPTIMAL SYSTEMS

Softwaredokumentation
enaio® appconnector

Version 8.50

Sämtliche Softwareprodukte sowie alle Zusatzprogramme und Funktionen sind eingetragene und/oder in Gebrauch befindliche Marken der OPTIMAL SYSTEMS GmbH, Berlin oder einer ihrer Gesellschaften. Sie dürfen nur mit gültigem Lizenzvertrag benutzt werden. Die Software sowie die jeweils zugehörige Dokumentation sind nach deutschem und internationalem Recht urheberrechtlich geschützt. Das illegale Kopieren und Vertreiben der Software stellt Diebstahl geistigen Eigentums dar und wird strafrechtlich verfolgt. Alle Rechte vorbehalten, einschließlich der Wiedergabe, Übermittlung, Übersetzung sowie Speicherung mit/auf Medien aller Art. Für vorkonfigurierte Testszenarien oder Demo-Präsentationen gilt: Alle Firmennamen und Personen, die in Beispielen (Screenshots) erscheinen, sind frei erfunden. Eventuelle Ähnlichkeiten mit tatsächlich existierenden Firmen und Personen sind zufällig und unbeabsichtigt.
In Dokumentationen enthaltene Codebeispiele dienen der Verdeutlichung der Funktionalität und können ohne Überprüfung und Anpassung nicht übernommen werden. Zur übersichtlicheren Darstellung sind Codebeispiele beispielsweise mit Zeilenumbrüchen abgebildet, die in der jeweiligen Programmiersprache nicht erlaubt sind. Aus diesem Grund kann OPTIMAL SYSTEMS GmbH keine Haftung für Codebeispiele übernehmen. Gern beraten wir Sie bei der Konfiguration und Einrichtung Ihres Systems.

Copyright 1992 – 2018 by OPTIMAL SYSTEMS GmbH
Cicerostraße 26
D-10709 Berlin

28.06.2018
Version 8.50

Inhalt

Zur Einführung

Das Handbuch liegt Ihnen als PDF-Datei vor. Die PDF-Datei wird in das Dokumentationsverzeichnis installiert. Sie kann mit Adobe Reader am Bildschirm gelesen, ganz oder in Teilen ausgedruckt und schnell nach Begriffen durchsucht werden.

Das Handbuch beschreibt die Installation und Konfiguration der Schnittstelle enaio® appconnector, die beispielsweise zur Verwendung von enaio® app und für benutzerorientierte Anfragen aus Webanwendungen eingerichtet werden muss.

Für den Einsatz von enaio® appconnector als Detailvorschau in enaio® client lesen Sie bitte das Administrationshandbuch. Die Benutzeranleitung finden Sie im enaio® client-Handbuch.

enaio® appconnector

Über enaio® appconnector

enaio® appconnector ist ein Kerndienst von enaio®.

Die Kerndienste sind Standardkomponenten von enaio®, die für den Betrieb der enaio®-Plattform und dem reibungslosen Zusammenspiel der einzelnen enaio®-Komponenten erforderlich sind.

Als REST-Schnittstelle (Representational State Transfer) ermöglicht enaio® appconnector den ressourcenorientierten, flexiblen HTTP-Zugriff auf Index- und Dokumentendaten in enaio®.

enaio® appconnector dient somit beispielsweise:

- als Schnittstelle zu mobilen Anwendungen, wie enaio® app für Android-basierte Smartphones, dem iPhone und Tablet-PCs
- als Detailvorschau zur Anzeige von Indexdaten und weiterer Daten in enaio® client und anderen externen Anwendungen

Im Anhang finden Sie u. a. die OSRest-API-Dokumentation.

IT-Sicherheit

Mobile Anwendungen werden überwiegend unterwegs auf kleinen, portablen Smartphones oder Tablet-Computern verwendet. Daher können sie jedoch leicht durch Dritte in einem unaufmerksamen Moment entwendet werden. Außerdem stellen Apps, die aus Unkenntnis oder durch Dritte installiert werden und unbemerkt Informationen auslesen, eine Gefahr dar.

Mobile Anwendungen müssen also sicherstellen, dass sensible Daten nicht in die Hände Dritter fallen. Ausführliche Informationen zur IT-Sicherheit für den Einsatz von enaio® appconnector und enaio® app finden Sie im Systemhandbuch im Kapitel 'Sicherheit'.

Systemvoraussetzungen

enaio® appconnector wird zusammen mit einem Apache Tomcat Version 6 ausgeliefert.

Aus Sicherheitsgründen empfiehlt es sich, den Anwendungsserver für die Anwendung in einer DMZ mit vorgelagertem Webserver (z. B. einem Apache HTTPD Server) zu betreiben.

> Aus Sicherheitsgründen wird die Verwendung von SSL-Verschlüsselung empfohlen.

Diese kann entweder durch einen Apache Tomcat Server oder durch einen vorgelagerten Webserver übernommen werden. Dort sollte zudem eine Kompression der Inhalte über Gzip konfiguriert werden (z. B. über das Apache Modul `mod_deflate`).

Es ergeben sich folgende Systemvoraussetzungen für die Installation und den Betrieb von enaio® appconnector:

- Die Lizenz 'APP' muss bei enaio® server registriert sein. Für enaio® app muss die Lizenz 'MOB' registriert sein.
- Für die Konfiguration benötigen Sie die Systemrolle 'DMS: Supervisor'.

Installation

Die REST-Schnittstelle enaio® appconnector wird als Dienst über das enaio®-Setup installiert. Wählen Sie dazu beim Setup den Kerndienst enaio® appconnector.

Die Laufzeitumgebung (JDK und Anwendungsserver) wird automatisch mitinstalliert.

> Die installierte Laufzeitumgebung sollte nur von diesem Kerndienst verwendet werden, da bei der Aktualisierung des Kerndiensts über das Setup auch die Laufzeitumgebung aktualisiert wird. Wenn mit der Laufzeitumgebung andere enaio®- oder Fremdkomponenten betrieben werden, kann möglicherweise die Aktualisierung nicht korrekt durchgeführt werden oder die anderen Komponenten funktionieren dann nicht mehr.

Bei der Installation geben Sie folgende Informationen an:

- Servername und Port
- Name und Passwort des technischen Benutzers

Der Dienst wird durch das Setup automatisch an enaio® server registriert.

> Dem technischen Benutzer, unter dessen Benutzerkonto enaio® appconnector ausgeführt wird, muss die Systemrolle 'Server: Jobkontext wechseln' zugewiesen werden.

Damit ist die Installation von enaio® appconnector abgeschlossen und Sie können die Anwendung 'osrest' im Webanwendungs-Manager starten.

Bei der Installation registriert enaio® appconnector seinen Service-Endpoint bei enaio® server, sodass er von anderen Komponenten ausgelesen werden kann. Einsehen und ändern können Sie den Service-Endpoint in enaio® enterprise-manager unter **Servereigenschaften > Kategorie: Services > 'Kerndienst' > Service-Endpoint.**

Installation eines Hotfixes oder Patches

Bei der Installation eines Hotfixes oder eines Patches werden nur die Dateien ersetzt, die sich gegenüber der bestehenden Version verändert haben. Das

Aktualisieren auf eine neue Version ist mit der Installation eines Hotfixes oder eines Patches nicht möglich.

Da beim Hotfix oder Patch nur wenige Systemdateien ausgetauscht werden, entfällt ein erneutes Konfigurieren der enaio® appconnector-Installation.

Eine Sicherung der bestehenden enaio® appconnector-Installation wird nicht durchgeführt.

Bei der Installation eines Hotfix oder eines Patches wird vor dem Ersetzen von Dateien geprüft, ob die Version der bestehenden Installation zu der des Hotfix oder Patch passt. Ist das nicht der Fall oder ist bereits ein neuerer Hotfix oder Patch installiert, werden die Dateien nicht ersetzt. Der Hotfix-/Patch-Installer bricht dann mit der Meldung ab, dass der installierte Service die falsche Version trägt.

Hotfixes (`osappconnector_hotfix.exe`) befinden sich im `SP`-Verzeichnis der Installationsdaten.

Patches (`osappconnector_patch.exe`) erhalten Sie von OPTIMAL SYSTEMS und sind als Download im Partnerportal, dem Serviceportal für Partner und Kunden der OPTIMAL SYSTEMS Gruppe, verfügbar.

Sobald ein Patch in eine Installation eingespielt wurde, wird das SP-Verzeichnis bei weiteren Aktualisierungen über das enaio®-Setup nicht mehr ausgelesen. Nachträgliche Änderungen können dann ausschließlich per Patch eingespielt werden. Das Einspielen eines Patches kann nicht wieder rückgängig gemacht werden.

Aktualisierung des Kerndiensts

Die Aktualisierung des Kerndiensts wird über das enaio®-Setup vorgenommen.

Dabei wird der aktuelle Stand der Konfigurationsdateien automatisch gesichert. Die gesicherten Konfigurationsdateien befinden sich nach einer Aktualisierung im Programmunterverzeichnis `backup-(Zeitstempel)`des Kerndiensts.

Konfiguration

Für die Konfiguration von enaio® appconnector steht eine Konfigurationsdatei zur Verfügung.

Wenn enaio® appconnector manuell und nicht über das enaio®-Setup installiert wurde, müssen Sie in enaio® enterprise-manager die Home-URL von enaio® appconnector im Bereich **Servereigenschaften > Kategorie: Services > AppConnector** eintragen.

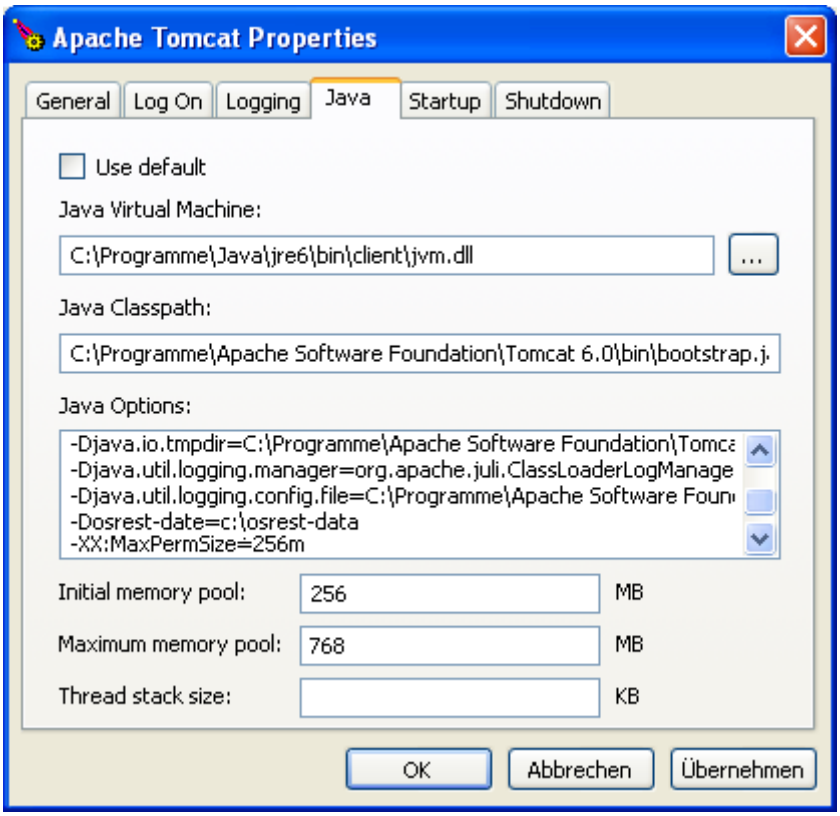
Beispiel: `http://localhost:8060/osrest`

Außerdem müssen Sie vor dem Bearbeiten der Konfigurationsdatei zusätzlich die JVM konfigurieren (siehe 'JVM konfigurieren').

JVM konfigurieren

Da enaio® appconnector ein externes Verzeichnis für die Konfigurationsdaten verwendet, müssen Sie ein Verzeichnis außerhalb der Apache Tomcat-Verzeichnisse anlegen und den Inhalt des Installationsdatenverzeichnisses `…\services\os_appconnector\osappconnector-<Version>.zip\osappconnector\configuration` an den neuen Standort kopieren.

Öffnen Sie dann den Apache Tomcat-Eigenschaftendialog und geben Sie auf dem Reiter **Java** über die Systemeigenschaft `osrest-data` das Verzeichnis außerhalb der Tomcat-Installation an, z. B. `-Dosrest-data=c:\osrest-data`.



enaio® appconnector erfordert eine minimale Heapgröße der JVM (**Initial memory pool**) von 256 MB und eine maximale Größe (**Maximum memory pool**) von 768 MB. Diese Werte gelten als Ausgangsgrößen und sollten, je nachdem, ob auch andere Anwendungen im Anwendungsserver betrieben werden, oder sehr große Dokumente verarbeitet werden sollen, den spezifischen Produktionsbedingungen angepasst werden. Es empfiehlt sich außerdem, den Speicher, in dem sich die JVM selbst verwaltet (Permanent Generation), auf 256 Byte zu setzen ('-XX:MaxPermSize=256m').

Die Konfigurationsdatei von enaio® appconnector

Um enaio® appconnector zu konfigurieren, bearbeiten Sie die Konfigurationsdatei `osrest.properties` im Programmverzeichnis `…\services\OS_Appconnector\configuration`.

Dem technischen Benutzer, unter dessen Benutzerkonto enaio® appconnector ausgeführt wird, muss die Systemrolle 'Server: Jobkontext wechseln' zugewiesen werden.

In der Konfigurationsdatei können Sie folgende Einstellungen vornehmen:

Serververbindung

`connection.string`	Verbindungsdaten zum enaio®-Server nach dem Muster `[Name|IP]:[Port]:[Wichtung]`. Mehrere Server werden mit einem #-Zeichen getrennt. Die Wichtung eines Servers legt fest, mit welcher Wahrscheinlichkeit in Prozent eine Verbindung zu diesem Server aufgebaut wird. Wenn Sie nur einen Server verwenden, gilt der Wert '100'. Bsp.: `localhost:4000:50#127.0.0.1:4600:50`
`technical.user.name`	Name des technischen Benutzers
`technical.user.password`	Passwort des technischen Benutzers Beachten Sie, dass das hier eingetragene Passwort nicht verschlüsselt ist. Sie können auch das verschlüsselte Passwort inklusive #-Zeichen aus der Benutzertabelle der enaio®-Datenbank eingeben.

Allgemeine Konfiguration

`default.pagesize`	Maximale Trefferanzahl, die über enaio® appconnector geliefert wird. Standardwert sind 500 Treffer.
`favouritesPortfolioName`	Für Versionen vor 7.10 kann eine Mappe angegeben werden, die die Favoriten enthält. Sie geben das Thema an.
`notifications.skipworkflow`	Workflow deaktivieren Wenn Sie den Parameter auf 'true' setzen, werden Workflowfunktionen deaktiviert.
`metadata.defaultmappingname`	Name der Konfigurationsdatei, in der definiert ist, welche Indexdatenfelder der einzelnen enaio®-Objekttypen bspw. in enaio® app und in enaio® sync angezeigt werden. Der Dateiname muss ohne die Dateiendung angegeben werden. Die Datei befindet sich in dem Verzeichnis `…\configuration\schema`.
`osecm.default.nameschema`	Sie geben an, dass die internen Namen der Objektdefinition für das Zuweisen der

	Indexdaten verwendet werden. Standardwert ist internal_name. Alternativ können auch die Namen der DMS-Objekte und deren Indexdatenfelder verwendet werden (name), jedoch müssen die Namen wie die internen Namen bestimmten Konventionen folgen (siehe 'Skriptsprache').
`admin.email`	E-Mail-Adresse, an die Benutzer der App Fehlermeldungen senden können. Zusätzlich zur Fehlermeldung enthält die E-Mail auch den Stacktrace zum Fehler.
`welcomepage`	URL zu einer mobiltauglichen Willkommensseite Die Willkommensseite wird in enaio® app beim Start und beim Serverprofilwechsel angezeigt, sofern der Benutzer die App-Einstellung **Willkommensseite immer anzeigen** aktiviert hat. Beachten Sie, dass der Doppelpunkt in der URL maskiert werden muss. Bsp.: `http\://www.ecm.mobi`

Verbindung zu den Kerndiensten

`base.url`	Basis-URL von enaio® appconnector Die URL wird automatisch aus der Server-Registry ausgelesen und hier eingetragen. Wenn Sie hier eine andere URL eingeben, wird der Wert in der Registry ignoriert.
`fileservice.contentviewerurl`	Basis-URL zu enaio® contentviewer Sie wird automatisch aus der Server-Registry ausgelesen und hier eingetragen. Wenn Sie hier eine andere URL eingeben, wird der Wert in der Registry ignoriert.
`fileservice.documentviewerurl`	Basis-URL zu enaio® documentviewer enaio® documentviewer erstellt die Vorschauen von enaio®-Objekten, die Sie bspw. über enaio® app einsehen können. Die URL wird automatisch aus der Server-Registry ausgelesen und hier eingetragen. Wenn Sie hier eine andere URL eingeben, wird der Wert in der Registry ignoriert. Bsp. `https://demo.optimal-systems.org/osdocumentviewer`

`services.renditioncache`	URL des Rendition-Service Die hier angegebene URL überschreibt den entsprechenden Wert in enaio® enterprise-manager.
`services.detailsviewer`	URL von enaio® detailsviewer Die hier angegebene URL überschreibt den entsprechenden Wert in enaio® enterprise-manager.
`fileservice.osweburl`	Basis-URL zu enaio® webclient Die URL wird benötigt, um beispielsweise den Inhalt von DMS-Objekten, den Sie über enaio® app nur ansehen können, auf Ihrem mobilen Endgerät mit enaio® webclient auch zu bearbeiten. Bsp. `https://demo.optimal-systems.org/osweb`
`extractionservice.url`	Basis-URL zu enaio® extraction Wenn Sie diese Komponente (Extraktion von EXIF-Daten aus Audio-, Video- und Bilddateien, XMP-Daten aus Office- und PDF-Dokumenten sowie Standardeigenschaften aus E-Mails im MSG- oder EML-Format) einsetzen möchten, wenden Sie sich an das Professional Services-Team von OPTIMAL SYSTEMS.

Authentifizierung

`authentication.usebasic`	Wenn eine Anmeldung über Basic Authentication möglich sein soll, muss 'true' angegeben werden.
`authentication.usebasic.windowsauth`	Wenn eine Anmeldung über die Windows-Authentisierung möglich sein soll, muss 'true' angegeben werden.
`authentication.defaultdomain`	Standard-Windows-Domäne für die Basic Authentication
`authentication.usentlm`	Bei Bedarf kann die NTLM-Authentifizierung aktiviert werden ('true'). Beachten Sie jedoch, dass die Authentifizierung aller Kerndienste standardmäßig von enaio® gateway übernommen wird.
`authentication.useprofileuser`	Ein Profilbenutzer kann verwendet werden ('true'), wenn alle Aufrufe ohne Authentifizierung realisiert werden sollen.

	Mit den Parametern `profile.user.password` und `profile.user.name` geben Sie Name und Passwort des Profilbenutzers an.
`profile.user.name`	Name des Profilbenutzers Die Angabe ist nur erforderlich, wenn Sie den Parameter `authentication.useprofileuser` auf 'true' gesetzt haben.
`profile.user.password`	Passwort des Profilbenutzers Die Angabe ist nur erforderlich, wenn Sie den Parameter `authentication.useprofileuser` auf 'true' gesetzt haben. Beachten Sie, dass das hier eingetragene Passwort nicht verschlüsselt ist. Sie können auch das verschlüsselte Passwort inklusive #-Zeichen aus der Benutzertabelle der enaio®-Datenbank eingeben.
`authentication.restrictaccesstogroups`	OS-Gruppen, die Zugriff auf den Dienst erhalten sollen Mehrere Gruppen werden mit Komma getrennt angegeben. Benutzer müssen stets in allen Gruppen Mitglied sein, damit sie Zugriff auf den Dienst erhalten.

Pushnotification

`services.pushnotification.enabled`	Wenn das System Push-Benachrichtigungen an mobile Geräte verschicken soll, muss 'true' angegeben werden.
`services.pushnotification.production`	Interner Parameter, der nicht geändert werden darf. Anderenfalls funktioniert der Dienst zur Pushbenachrichtigung nicht mehr.
`services.pushnotification.version`	Interner Parameter, der nicht geändert werden darf. Anderenfalls funktioniert der Dienst zur Pushbenachrichtigung nicht mehr.
`services.pushnotification.proxy.enabled`	Wird eine Verbindung mit dem Internet über einen Proxy hergestellt, muss 'true' angegeben werden.
`services.pushnotification.proxy.address`	Wird eine Verbindung mit dem Internet über einen Proxy hergestellt, so ist dessen Adresse anzugeben.
`services.pushnotification.proxy.port`	Port des Proxy-Servers
`services.pushnotification.proxy.password`	Passwort des Proxy-Benutzers

`services.pushnotification.proxy.username`	Name des Proxy-Benutzers
`res.revision`	Interner Parameter, der nicht geändert werden darf.

Einstellungen für den Pushnotificationservice sind unten beschrieben ('Pushnotificationservice für enaio® apps').

Damit Änderungen an der Konfigurationsdatei wirksam werden, muss der Kerndienst neu gestartet werden.

enaio® appconnector protokolliert alle Aktionen in die Datei `osrest.log` im Programmverzeichnis `…\services\OS_Appconnector\configuration\logs`.

Als Standard wird der Stacktrace protokolliert und kann über den Browser eingesehen werden. Über einen Eintrag des Parameters `hide.stacktrace` in die Konfigurationsdatei `osrest.properties` kann die Stacktrace-Übertragung zum Browser ausgeschaltet werden.

enaio® client

Profilbilder für Notizen in der Detailvorschau

Die Detailvorschau in enaio® client bietet die direkte Eingabe von Textnotizen an, neben denen das Profilbild des Schreibers angezeigt wird.

Profilbilder werden nur angezeigt, wenn im Programmverzeichnis `…\services\OS_Appconnector\configuration\avatar` JPG-Bilder der Benutzer mit den Abmessungen 64x64 Pixel liegen. Die Namen der Bilder müssen dem enaio®-Benutzernamen entsprechen, beispielsweise `root.jpg` oder `demo.jpg`.

Capabilities von enaio® appconnector

Die Capabilities von enaio® appconnector konfigurieren Sie über die Konfigurationsdatei `osrest.caps.xml` im Programmverzeichnis `…\services\OS_Appconnector\configuration`.

Die Capabilities einer Installation werden durch verschiedene Aspekte gesteuert, bspw. durch die Version von enaio® appconnector, der enaio®-Lizenzen, administrative Restriktion und Systemrollen des enaio®-Benutzers.

Capabilities können ein- oder ausgeschaltet werden (true/false) oder Werte annehmen. Capabilities gelten für Android-, iOS- und Windows-Geräte, falls in der Konfiguration nicht Werte für die jeweilige Plattform angegeben sind. Sie können dies durch Einfügen des Attributs `ua` im Element `defaultto` der entsprechenden Capability steuern.

Beispiel:

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

Hier können Sie folgende Einstellungen vornehmen:

Kategorie	Capabilityname	Funktion	Standard
Versionsabhängige Capabilites			
VER	DateRanges	Zeitintervalle können als von-bis-Datumsangabe erfolgen.	true
VER	MarkInboxItemsAsRead	Alle Elemente werden optisch unterschiedlich dargestellt, je nachdem ob sie ungelesen oder bereits gelesen wurden.	true
Lizenzabhängige Capabilites			
LIC	Capture	Der Kameraeintrag bzw. der Fotobibliothekeintrag wird in der DropTargetliste angezeigt.	true
LIC	Fulltext	Die Volltextsuche wird in der Liste der Anfragen angezeigt.	true
LIC	Import	Der Erfassungsbereich ist aktiv oder inaktiv.	true
LIC	Inbound	Funktion 'in mobileDMS öffnen' steht externen Apps zur Verfügung	true
LIC/POL	Offline	Dokumente werden zwischengespeichert und Favoriten sind offline verfügbar. Ist dies inaktiv, werden alle Dokumente nach Verlassen des Dokumentenbetrachters aus dem App-Speicher entfernt.	true
LIC/POL	Outbound	Funktion 'Dokument öffnen' steht im Action-Menü des Dokumenten-betrachters zur Verfügung.	true
LIC	Workflow	Werden im Eingangs-Tab Workflows angezeigt oder nicht.	true

Policyabhängige Capabilites			
POL	PlattformNotifications	Kann die Serverprofiloption 'Pushbenachrichtigungen' aktiviert werden oder nicht.	true
POL	AllowDetailsView	Der Dokumentenbetrachter steht zur Verfügung. Falls nicht, werden nur Indexdaten angezeigt.	false
POL	AllowInbox	Anzeige des Bereichs 'Eingang'	true
POL	AllowQueries	Stellt den Anfragen-Tab zur Verfügung.	true
POL	AllowUsertray	Stellt in der Liste der Anfragen den Eintrag 'Benutzerablage' zur Verfügung.	true
POL	CacheTime	Wie lange werden Dokumente im App-Speicher aufbewahrt und stehen auch Offline zur Verfügung. Hier muss ein ganzzahliger Wert für die Anzahl an Minuten eingetragen werden. z. B. 10080 für 7 Tage.	10080
POL	ForceClientSSL	Es wird durch den Client geprüft, ob für jeden Datenaustausch eine Verschlüsselung genutzt wird.	
POL	ForcePinLock	Muss die App über eine PIN-Abfrage gesichert sein oder nicht. Bei 'true' kann man sich nur zum Server verbinden, wenn man eine PIN-Abfrage eingerichtet hat.	false

POL	ForceSSLTrust	Darf der Anwender die Serverprofiloption 'Serverzertifikat vertrauen' aktivieren und somit auch eine Verbindung zu enaio® appconnector herstellen bei nicht gültigen SSL-Serverzertifikaten.	false
POL	ForceStartView	Start-Tab kann hier vorgegeben werden. Werte: inbox, queries, favorites, outbox	-
POL	ForceWelcome	Wird die Willkommensseite bei jedem Start angezeigt? Bei 'true' kann diese Option in den Settings nicht abgewählt werden.	false
POL	MaxHierarchyDepth	Hierarchietiefe von zwischengespeicherten Favoriten-Objekten	10
POL	AllowFavourites	Anzeige des Bereichs 'Favoriten'	true
POL	AutoReloadInbox	Festlegen, in welchen Abständen (in Minuten) der Eingangskorb neu geladen wird. Erlaubte Werte: 0 (= deaktiviert), 1, 5, 10, 30, 60	0
POL	AllowLocation	Darf der Standort zu einem Objekt ermittelt werden.	true
Sonstige Capabilites			
CFG	UrlRenService	URL zum enaio® rendition-Dienst	
CFG	WelcomeURL	Bei jeder neuen Verbindung zu einem Server wird dessen Willkommensseite anzeigt.	true

enaio® apps

Indexdaten anzeigen

Für die Anzeige der enaio®-Objekte über enaio® apps auf dem mobilen Endgerät müssen Sie festlegen, welche Indexdatenfelder der einzelnen Objekttypen in der

App sichtbar sein sollen. Dazu weisen Sie die enaio®-Indexdatenfelder den Feldern in der App zu. Diese Zuweisung nehmen Sie in der Konfigurationsdatei `OSMetadata.xml` vor.

Ohne die Zuweisung werden die DMS-Objekte zwar in der App aufgelistet, jedoch sind sie aufgrund der nicht ausgefüllten Felder nicht identifizierbar.

Diese Zuweisung wird ebenso für enaio® sync verwendet.

Die Datei `OSMetadata.xml` wird vom Setup installiert und befindet sich im Programmverzeichnis `configuration\schema`. Da die Konfigurationsdatei `OSMetadata.xml` standardmäßig Beispieldaten enthält, müssen Sie die Datei anpassen.

Die Beispielkonfigurationsdatei ist wie folgt aufgebaut:

```
<?xml version="1.0" encoding="UTF-8"?>
<archive>
    <model>
        <property name="title"/>
        <property name="info"/>
    </model>
    <cabinet name="Kunde">
        <object name="Sonstiges">
            <property name="title" field="Typ"/>
            <property name="info" field="Kurztext"/>
        </object>
        <object name="Supportcall">
            <property name="title" field="Call_Nr + ' ' + Status"/>
            <property name="info" field="Problem"/>
        </object>
        <object name="Vertrieb">
            <property name="title" field="Typ"/>
            <property name="info" field="Kurztext"/>
        </object>
        <object name="Kunde">
            <property name="title" field="Firmenname"/>
            <property name="info" field="Strasse + ' ' + PLZ + ' ' +
Ort"/>
        </object>
        <object name="Email">
            <property name="title" field="'E-Mail von: ' +
MAIL_FROM"/>
            <property name="info" field="MAIL_SUBJECT"/>
        </object>
        <object name="Person">
            <property name="title" field="Vorname + ' ' + Name"/>
            <property name="info" field="Klassifizierung"/>
        </object>
        <object name="Eingangsrechnung">
            <property name="title" field="'Rechnung ' +
Rechnungsnummer"/>
            <property name="info" field="'Vom ' + Rechnungsdatum +
', ' + Status"/>
        </object>
        <object name="Dokument">
            <property name="title" field="Typ"/>
            <property name="info" field="empty(Beschreibung) ?
'leer' : Beschreibung"/>
        </object>
        <object name="Historie">
            <property name="title" field="Art"/>
            <property name="info" field="Thema"/>
```

```
        </object>
    </cabinet>
    <typeless>
        <property name="title" field="if (OBJECT_MAIN == 3) { y=
'Bild'; } else { y = 'Datei'; y }"/>
        <property name="info" field="'Erfasst am: ' +
OBJECT_CRDATE"/>
    </typeless>
    <notification>
        <workflow>
            <property name="title" field="ActivityName"/>
            <property name="info" field="ActivitySubject"/>
        </workflow>
    </notification>
</archive>
```

Im Bereich `<model>` sind die Felder `title` und `info` definiert, die in enaio® apps angezeigt werden und die Sie mit Indexdaten der enaio®-Objekttypen füllen können.

Dann wird in der Konfigurationsdatei für jeden Schrank der Tag `<cabinet name="Name des Schranks">` angegeben, in dem für die einzelnen Objekttypen (`<object name="Name des Objekts">`) die Indexdatenfelder zugeordnet werden. Im Tag `<property>` wird dazu im Attribut `name` der Name des Felds in der App definiert und im Attribut `field` werden die Indexdaten aus enaio® referenziert. Über eine spezielle Skriptsprache ist es möglich, hier die Werte verschiedener Indexdatenfelder frei zu konfigurieren (siehe 'Skriptsprache').

Typenlose Dokumente werden über den Tag `<typeless>` definiert, in dem Sie den Feldern der App den Haupttyp des Objekts (`OBJECT_MAIN`) und das Erfassungsdatum (`OBJECT_CRDATE`) zuordnen.

Beachten Sie, dass für die Zuordnung der enaio®-Indexdaten nur die internen Objekttypnamen der Objektdefinition verwendet werden können (siehe Handbuch 'enaio® editor').

Objekte ablegen

Über die App können neue Objekte an einem Standort in enaio® abgelegt werden. Standardmäßig können Benutzer dabei aus allen Objekttypen wählen, auf die sie in enaio® client auch zugreifen können. Ebenfalls standardmäßig enthalten die Indexdatenmasken nur die Pflichtfelder der Objekttypen.

Über die Konfigurationsdatei `OSMetadata.xml` können Sie festlegen, welche Objekttypen nicht und welche Indexdatenfelder zusätzlich zu den Pflichtfeldern in der App sichtbar sind.

Um Objekttypen für die Ablage auszublenden, fügen Sie den Tag `<insert active>` ein und setzen ihn auf 'false'. Der Standardwert des Tags ist 'true'.

Beispiel:

```
        <object name="Vertrieb">
            <property name="title" field="Typ"/>
            <property name="info" field="Kurztext"/>
           <insert active="false" />
        </object>
```

Für die Anzeige zusätzlicher Indexdaten fügen Sie pro Objekttyp den Tag `<insert active>` ein und führen darin alle Indexdatenfelder auf, die neben den Pflichtfeldern in der App bei der Neuablage angezeigt werden sollen.

Beispiel:

```
<object name="Sonstiges">
        <property name="title" field="Typ"/>
        <property name="info" field="Kurztext"/>
        <insert active="true">
                        <property field="Typ"/>
                        <property field="Kurztext"/>
        </insert>
</object>
```

Skriptsprache

Um eine Verarbeitung im Skriptumfeld zu garantieren, müssen die internen Namen der enaio®-Objekttypen nach folgender Konvention aufgebaut sein: Namen beginnen mit den Zeichen a-z, A-Z, _ oder &. Danach können die Zeichen 0-9, a-z, A-Z, _ oder $ verwendet werden. Bei den Namen von Variablen müssen Sie die Groß-/Kleinschreibung beachten.

Die folgenden reservierten Namen dürfen nicht als Variablennamen verwendet werden: `or`, `and`, `eq`, `ne`, `lt`, `gt`, `le`, `ge`, `div`, `mod`, `not`, `null`, `true`, `false` und `new`.

Für die Zuweisung der Indexdatenfelder aus enaio® zu den Feldern der enaio® apps können Sie die im Folgenden beschriebene Skriptsprache verwenden.

Sprachelemente

Statements	Statements werden mit einem Semikolon abgeschlossen.
Blöcke	Blöcke sind verschiedene Statements, die in geschweifte Klammern eingeschlossen sind.

Zuweisungen	Variablen können Werte über ein Gleichheitszeichen zugewiesen werden: `var='Wert';`

Literale

Integer	Eine oder mehrere Ziffern von 0 bis 9.
Floating Point	Eine oder mehrere Ziffern von 0 bis 9 gefolgt von einem Dezimalpunkt und weiteren Ziffern von 0 bis 9.
Boolean	`true` oder `false`
String	Zeichenketten, die in einfache Anführungszeichen eingeschlossen sind: `'Hallo Welt'`

Funktionen

empty	Gibt `true` zurück, wenn der folgende Ausdruck `null` ist. Ein leerer String, z. B. `empty(var1);`
size	Gibt die Länge eine Strings zurück, z. B. `size('Hello');`

Operatoren

Logische Operatoren	AND:
	`cond1 and cond2`
	`cond1 && cond2`
	OR:
	`cond1 or cond2`
	`cond1 || cond2`
	NOT:
	`!cond1`
	`not cond1`

Bedingte Operatoren	Es kann sowohl der übliche `condition ? if_true : if_false` Operator als auch die Abkürzung `value ?: if_false` verwendet werden:
	`val1 ? val1 : val2`
	`val1 ?: val2`

Vergleichsoperatoren	z. B.
	`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
```

Reguläre Ausdrücke	z. B.
	`var1 =~ 'abc.*'`
	`var1 !~ 'abc.*'`

Berechnungen	Es können Additionen, Subtraktionen, Multiplikationen und Divisionen durchgeführt werden:
	`val1 + val2`
	`val1 - val2`
	`val1 * val2`
	`val1 / val2`
	`val1 div val2`

Bedingung

if z. B.

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

DropTargets

DropTargets sind eine Möglichkeit, um Daten und Dokumente über enaio® appconnector in das enaio®-System zu importieren. Gleichzeitig ist es möglich, mit einem DropTarget verschiedene Aktionen im System auszulösen, wie etwa das Starten von Workflows und das Erstellen oder Aktualisieren von DMS-Objekten.

Pro DropTarget wird genau ein Importszenario definiert.

DropTargets werden in Jelly geschrieben, der Skriptsprache zum Schreiben von XML-Skripten.

DropTargets werden beispielsweise per Skript implementiert. Den hierfür benötigten API-Aufruf finden Sie im Anhang (siehe '/osrest/api/documentfiles/droptargets/[targetname]').

Konfiguration

DropTargets müssen im Programmverzeichnis `...\services\OS_Appconnector\configuration\droptargets` gespeichert werden. In diesem Verzeichnis finden Sie bereits die zwei Demo-DropTargets `demo.xml` und `demo2.xml`.

Jedes DropTarget ist in einer eigenen XML-Datei definiert. Innerhalb der XML-Datei dürfen keine Umlaute oder Sonderzeichen verwendet werden, außer sie sind UML-encoded.

Der Dateiname der XML ist gleichzeitig Name des DropTargets.

Aufbau eines DropTargets

In den folgenden Abschnitten werden der grundsätzliche Aufbau eines DropTargets sowie weitere optionale Elemente beschrieben.

Allgemeiner Aufbau

Ein DropTarget beginnt, wie bei Jelly-Skripten üblich, mit der Definition der benötigten Namensräume.

Dann muss der Schrank ausgewählt werden, in dem die Aktionen ausgeführt werden sollen. Es ist auch möglich, mehrere Schränke gleichzeitig auszuwählen. Sie werden dann nacheinander abgearbeitet.

Innerhalb des Schranks können Aktionen definiert werden. Vor allem ist es möglich Serverjobs zu erstellen, die nacheinander abgearbeitet werden. Jeder Serverjob muss ein DMS-Objekt enthalten, auf das er angewendet werden soll.

Im folgenden Beispiel wird ein Schrank mit dem internen Namen 'customer' und darin ein Ordner für den Kunden Franz Müller erstellt.

```
<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>
```

Über das Attribute `key` definieren Sie den Schrank anhand seines internen Namens.

Über das Attribut `keytype` definieren Sie, ob der interne Name (`internal_name`) oder der Name (`name`) des Schrank angegeben werden muss. Standardmäßig muss der interne Name angegeben werden, sodass das Attribut `keytype` im Skript nur erforderlich ist, wenn Sie statt des internen Namens den Namen eines DMS-Objekts angeben möchten.

Definieren Sie dann:

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

Das Standardverhalten können Sie ändern, indem Sie den Parameter `osecm.default.nameschema` in der Konfigurationsdatei `osrest.properties` von enaio® appconnector von `internal_name` auf `name` setzen. Voraussetzung dafür ist jedoch, dass die Namen wie die internen Namen bestimmten Konventionen folgen (siehe 'Skriptsprache' und 'Die Konfigurationsdatei von enaio® appconnector').

Verweise

In vielen Fällen ist es notwendig, die Ergebnisse von Serverjobs weiterzuverarbeiten. Um dies zu ermöglichen, gibt es die Attribute `id` und `ref`. Mit `id` wird ein DMS-Objekt in den Jelly-Kontext geschrieben, über `ref` kann das DMS-Objekt dann an einer anderen Stelle weiterverwendet werden.

Im folgenden Beispiel wird ein DMS-Objekt gesucht und dann gelöscht.

```
<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>
```

Das Ergebnis der Suche wird unter dem Bezeichner 'myfolder' in den Jelly-Kontext geschrieben. Der Tag <folder> innerhalb des Tags <delete> verweist nun auf das Ergebnis der Suche. Der gefundene Ordner wird also gelöscht. Standardmäßig schreibt der Tag <search> das erste Ergebnis in den Jelly-Kontext.

Serverjobs mit mehreren DMS-Objekten

Bei einigen Serverjobs müssen mehrere DMS-Objekte angegeben werden. Dabei muss die Aufgabe jedes DMS-Objekts explizipt definiert sein. Dies geschieht über das Attribut `purpose`.

Im folgenden Beispiel wird mit dem Tag <insert> ein Dokument erstellt. Dazu wird das DMS-Objekt mit dem Attribut `purpose="INSERT"` als zu erstellendes DMS-Objekt – hier ein Dokument mit dem internen Namen 'doc' – definiert. Das DMS-Objekt mit dem Attribut `purpose="LOCATION"` definiert den Standort, im Beispiel ein Ordner mit der 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>
```

Die meisten Serverjobs haben einen Standardzweck. Dem ersten DMS-Objekt, welches das Attribut `purpose` nicht explizit gesetzt hat, wird dieser Standardzweck dann zugeordnet. Wenn man im Beispiel auf `purpose="INSERT"` für `os:document` verzichtet, würde dieses Attribut automatisch gesetzt werden.

Zugriff auf Variablen

Innerhalb eines Jelly-Kontexts hat man Zugriff auf unterschiedliche Variablen. Hierzu zählen:

- Variablen, die von außerhalb an das Jelly-Skript übergeben wurden
- IDs und Objekttyp-IDs von DMS-Objekten
- Attribute von DMS-Objekten, d. h. deren Feldwerte

Der Zugriff auf diese Attribute erfolgt über folgende Notation:

```
${NAME_DES_UEBERGEBEN_ATTRIBUTS}
${OBJEKTNAME.FELDNAME}
${OBJEKTNAME['FELDNAME']}
```

In Zeile 1 wird auf ein übergebenes Attribut zugegriffen. Zeile 2 und 3 greifen auf das Feld eines DMS-Objekts zu, wobei die Notation in Zeile 3 nur notwendig ist, wenn der Name des Felds Leer- und/oder Sonderzeichen enthält.

Außerdem gibt es reservierte Felder, die den Zugriff auf bestimmte Objektinformationen erlauben:

```
${OBJEKTNAME.key}
${OBJEKTNAME.keyType}
${OBJEKTNAME.osid}
${OBJEKTNAME.objectTypeId}
```

Protokollierung

Der Zugriff auf Objektinformationen ist vor allem für das Erstellen aussagekräftiger Protokolleinträge sinnvoll.

Im folgenden Beispiel wird ein Supportcall erstellt und ein entsprechender Eintrag mit dem Log-Level INFO in das Protokoll geschrieben.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
     <os:cabinet key="Kunde" 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">Der Supportcall ${myNumber} mit der
OSID ${supportcall.osid} wurde erstellt.</os:logger>
     </os:cabinet>
</j:jelly>
```

Es wird empfohlen, Tags vom Typ `<logger>` außerhalb der Serverjobs zu positionieren. Andernfalls werden die Protokolle geschrieben, noch bevor die Serverjobs abgearbeitet sind.

Umgang mit mehreren Treffern

Über den Suchmodus kann die Anzahl der Treffer, die eine Suche liefert, definiert werden. Standardmäßig wird der erste Treffer in den Jelly-Kontext geschrieben. Mit dem Attribut `mode="ALL"` ist es auch möglich, alle Treffer zu erhalten.

```
${OBJEKTNAME[0].FELDNAME}
${OBJEKTNAME.FELDNAME}
${OBJEKTNAME[n].FELDNAME}
```

In Zeile 1 wird auf das Feld des ersten DMS-Objekts zugegriffen. Das erste DMS-Objekt lässt sich auch ohne explizite Angabe der Indexe ansprechen (Zeile 2). So ist es möglich, Änderungen am Suchmodus durchzuführen, ohne andere Stellen im Skript anpassen zu müssen. Auf alle weiteren DMS-Objekte wird mit dem entsprechenden Index zugegriffen (Zeile 3).

Mehrzweck-Tags

Einige Arbeitsabläufe, die öfter benötigt werden, wurden zu Tags zusammengefasst.

Der Tag `<select>` sucht beispielsweise nach einem DMS-Objekt und kann es erstellen, wenn es nicht vorhanden ist, oder aktualisieren.

```
<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">Benutzerhandbuch</os:field>
              <os:field key="author">Max Muster/os:field>
           </os:folder>
           <os:folder purpose="INSERT, UPDATE">
              <os:field key="name">Benutzerhandbuch</os:field>
              <os:field key="author">Max Muster/os:field>
              <os:field key="visiblefor">Schmidt</os:field>
              <os:field key="editfor">Meier</os:field>
           </os:folder>
        </os:select>
     </os:cabinet>
</j:jelly>
```

Es ist möglich, Mehrzweckobjekte zu definieren. Im Fall oben wird derselbe Ordner für das Einfügen wie auch für das Aktualisieren verwendet.

Standardmäßig sind die Tags `<update>` und `<insert>` mit den Werten 'true' belegt. Deshalb müssen auch DMS-Objekte für den Zweck `purpose="INSERT, UPDATE"` definiert werden.

Prozessoren verwenden

Mit Hilfe von Datei-Prozessoren ist es möglich, Metadaten auszulesen und im DropTarget zu verwenden.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
     <os:email id="Mail" />
     <os:cabinet key="Emailarchiv" 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>
```

Der E-Mail-Prozessor wird in der zweiten Zeile definiert. Dateien in unterstützten Dateiformaten können dem DropTarget übergeben und deren auslesbaren Metadaten über `${<id>.<attribut>}` referenziert werden.

Prozessoren sind unabhängig von Schränken und können daher auch außerhalb von `<cabinet>`-Tags stehen.

Datum formatieren

Um ein Datum nach einem entsprechenden Muster zu formatieren, bietet Jelly eigene Mittel an, die sich an der JSP Standard Tag Library orientieren.

```
<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 Zeile 1 wurde der Namensbereich `fmt` definiert. Danach wird eine Variable erstellt und dann nach dem Muster `yyyy` formatiert als `year` in den Kontext geschrieben. Über `${year}` lässt sich nun innerhalb des Jelly-Skripts auf das aktuelle Jahr zugreifen.

Datum konvertieren

enaio® arbeitet mit folgenden Datumstypen: UNIX-Timestamp, deutsches Datum und deutsches Datum mit Uhrzeit. Anwendungen, die die DropTargets benutzen, sollen ein Datum immer als UNIX-Timestamp bereitstellen. Da enaio® selbstständig keine Konvertierung durchführen kann, muss diese innerhalb der DropTargets durchgeführt werden.

```
<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 den Zeilen 2-4 wird die Variable `timestamp`, die einen UNIX-Timestamp enthält in ein Java-Date-Objekt umgewandelt. In Zeile 7 wird dieses Objekt in das Format `dd.MM.yyyy` konvertiert und in der Variable `date` gespeichert.

Datumsformate und reguläre Ausdrücke auswerten

Um Daten aus Fremdsystemen übernehmen zu können, können Droptargets Datumsformate und reguläre Ausdrücke auswerten und verarbeiten.

Beispiel für Datumsformate:

```
<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>
```

Als Wert (value-Attribut) kann dabei ein Date-Objekt oder ein Datum als Zeichenkette übergeben werden. Mit Hilfe eines angegeben Patterns (inputPattern-Attribut) kann nach den Regeln der SimpleDateFormat-Klasse eine Zeichenkette

eingelesen werden. Die Stelle, ab wann das Pattern angewendet wird, kann mit dem parsePosition-Attribute (gezählt wird nullbasiert) festgelegt werden.

Beispiel für reguläre Ausdrücke:

```
<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>
```

Der Wert wird in das value-Attribut geschrieben. Der reguläre Ausdruck wird im pattern-Attribut angegeben.

Um auf die Ergebnisse der Formatierung zugreifen zu können, müssen im regulären Ausdruck Gruppen definiert werden. Die Ergebnisse sind dann über Nummer der jeweiligen Gruppe erreichbar.

Mehrere Tabellenzeilen hinzufügen

Mit den Mitteln von Jelly ist es auch möglich, mehrere Tabellenzeilen innerhalb eines DMS-Objekts hinzuzufügen. Hierbei wird der Tag <forEach> aus dem Jelly-Namensbereich eingesetzt.

```
<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 Zeile 6 wird eine Schleife definiert. Es wird über eine Liste iteriert, die unter dem Namen `hist` im Jelly-Kontext vorhanden ist. Die einzelnen Einträge der Liste sind dann unter dem Namen `entry` erreichbar. Innerhalb der Schleife muss der Tag <row> zur Definition der einzelnen Spalten der Tabellenzeile angegeben werden.

Das folgende Beispiel zeigt den Aufbau der Variable in JSON.

```
{
     "hist":[
        {
           "date":"01.02.2001",
           "user":"Hans",
           "state":"Verhandlung"
```

```
        },
        {
           "date":"12.12.2000",
           "user":"Petra",
           "state":"abgeschlossen",
           "prio":"hoch"
        }
     ]
}
```

Bei der Variable handelt es sich um eine Liste, die als Einträge Key-Value-Paare (Map) enthält.

Eine Verweiskopie erstellen

Eine Verweiskopie wird mit dem Tag `<link>` erstellt.

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

Ein Verweisdokument erstellen

Ein Verweisdokument wird mit dem Tag `<reference>` erstellt.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
     <os:cabinet key="Kunde" keyType="name">
        <os:reference>
           <os:document key="Email">
              <os:field key="Von:">peter@optimal-
systems.de</os:field>
              <os:field key="An:">schmidt@optimal-
systems.de</os:field>
              <os:field key="Datum:">12.12.2013</os:field>
           </os:document>
           <os:document ref="dokumentAufDasVerwiesenWerdenSoll"
purpose="SOURCE"/>
           <os:register ref="standortFuerDasVerweisdokument"
purpose="LOCATION"/>
        </os:reference>
     </os:cabinet>
</j:jelly>
```

Wenn das Dokument, auf das verwiesen werden soll, aus einem anderen Schrank kommt, so muss bei einer Suche auch einen weiteren Tag `<cabinet>` genutzt werden.

Einen Workflowprozess starten

Innerhalb eines DropTargets lassen sich auch Workflowprozesse starten.

```
<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">meine@mail.de</os:datafield>
           <os:datafield type="string"
key="d_Eingang">NEIN</os:datafield>
           <os:object osid="123456" objectTypeId="141222"/>
           <os:object ref="myfolder"/>
        </os:process>
     </os:start>
</j:jelly>
```

Ein Workflowprozess wird mit dem Tag <start> gestartet. Analog zu den anderen Serverjobs, muss der Tag <start> einen Tag <process> und dieser wiederrum Tags vom Typ <datafield> enthalten.

Außerdem können auch DMS-Objekte definiert werden, die in die Workflowakte eingefügt werden.

Der Tag <start> kann sich sowohl im root wie auch in einem Tag <cabinet> befinden.

Beschreibung für einen Client hinzufügen

Falls die DropTargets von einem Client benutzt werden sollen, können Informationen über das DropTarget und die benötigten Felder mit angegeben werden.

```
<j:jelly xmlns:j="jelly:core"
xmlns:os="jelly:com.os.droptargets.DropTargetTagLibrary">
     <description>
        <title>Ablage Eingangsrechnung</title>
        <longtitle>
          Ablage von Eingangsrechnungsdokumenten in einem separaten
Register der Kundenakte.
        </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})?$">fieldname</field>
        <cabinet key="Kunde">
           <object key="Kunde">
              <field key="SUPFEST" validate="^4[0-9]{12}(?:[0-
9]{3})?$">fieldname1</field>
              <field key="Firmenname">fieldname2</field>
              <field key="Bundesland">fieldname3</field>
           </object>
        </cabinet>
        <files>
           <file fileid="DOC_01" filetype="PDF"/>
           <file fileid="DOC_02" filetype="PDF"/>
        </files>
     </description>
</j:jelly>
```

Die Beschreibung muss sich im Tag <description> befinden. Allgemeine Infos zum Droptarget können mit den Tags <title>, <longtitle> und <helptext> definiert werden. Um Felder zu definieren, gibt es folgende Möglichkeiten:

1. Ein Feld mit allen benötigten Informationen direkt unter dem Tag `<description>` angeben.
2. Feld in einer `<cabinet>-<object>`-Struktur angeben. Die Informationen werden aus der Objektdefinition geparsed.

Das Attribut `validate` ist optional. Es muss vom Client ausgewertet werden.

Wenn dem DropTarget mehrere Dateien mitgegeben werden sollen, können unter dem Tag `<files>` IDs gesetzt werden. Wenn nur eine Datei übergeben wird, sind die IDs hinfällig und es wird der interne Standardname (`Default_Files`) verwendet.

Tag-Übersicht

DMS-Objekte und deren Felder

Die folgenden Tags stellen DMS-Objekte in enaio® dar.

<cabinet>

Mit dem Tag geben Sie den Schrank an, in dem die Aktionen ausgeführt werden sollen.

Kann enthalten sein in	Kann enthalten
<root>	<search> <update> <insert> <move> <copy> <delete> <link> <select> <reference> <logger> <return>

Attribute:

§ key: Bezeichner des Schranks

§ keyType (optional): 'name' oder 'internal_name' (Standard)

<folder>

Mit dem Tag geben Sie einen Ordner an.

Kann enthalten sein in	Kann enthalten
<search> <update> <insert>	<field> <table>

<move> <copy> <delete> <select> <process>	

Attribute:

- osid (optional): Die OSID des Ordners.
- objectTypeId (optional): Objekttyp-ID des Ordners.
- id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.
- ref (optional): Verweis auf einen weiteren Folder-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Werte des referenzierten Ordners übernommen.

<register>

Mit dem Tag geben Sie ein Register an.

Kann enthalten sein in	Kann enthalten
<search> <update> <insert> <move> <copy> <delete> <select> <process>	<field> <table>

Attribute:

- key: Bezeichner des Registers.
- keyType (optional): 'name' oder 'internal_name' (Standard)
- osid (optional): Die OSID des Registers.
- objectTypeId (optional): Objekttyp-ID des Registers.
- id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.
- ref (optional): Verweis auf einen weiteren Register-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Werte des referenzierten Registers übernommen.

<document>

Mit dem Tag geben Sie ein Dokument an.

Kann enthalten sein in	Kann enthalten
<search> <update> <insert> <move> <copy> <delete> <select> <process>	<field> <table>

Attribute:

- key: Bezeichner des Dokuments.
- keyType (optional): 'name' oder 'internal_name' (Standard)
- osid (optional): Die OSID des Dokuments.
- objectTypeId (optional): Objekttyp-ID des Dokuments.
- id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.
- ref (optional): Verweis auf einen weiteren Document-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Werte des referenzierten Dokuments übernommen.

<object>

Mit dem Tag geben Sie ein DMS-Objekt an, ohne zu definieren, ob es ein Ordner, Register oder Dokument ist.

Kann enthalten sein in	Kann enthalten
<search> <update> <insert> <move> <copy> <delete> <select> <process>	<field> <table>

Attribute:

- key: Bezeichner des DMS-Objekts.
- keyType (optional): 'name' oder 'internal_name' (Standard)
- osid (optional): Die OSID des DMS-Objekts.
- objectTypeId (optional): Objekttyp-ID des DMS-Objekts.
- id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.

§ ref (optional): Verweis auf einen weiteren Object-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Werte des referenzierten DMS-Objekts übernommen.

<process>

Dieser Tag repräsentiert einen Workflowprozess.

Kann enthalten sein in	Kann enthalten
<start>	<datafield> <folder> <register> <document> <object>

Attribute:

§ wfid: Die ID der Workflowfamilie, zu welcher der Prozess gehört.

§ clientType: Der Clienttyp, für den der Prozess gestartet werden soll.

OS_DESKTOP_CLIENT: Richclient

OS_MOBILE_CLIENT: Mobiler Client (App)

OS_WEB_CLIENT: Webclient

<field>

Mit diesem Tag geben Sie ein Textfeld an.

Kann enthalten sein in	Kann enthalten
<folder> <register> <document> <object> <row>	Wert als Klartext oder Jelly-Kontext-Variable, z. B. ${myVariable}

Attribute:

§ key: Bezeichner des Felds.

§ keyType (optional): 'name' oder 'internal_name' (Standard)

§ id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.

§ ref (optional): Verweis auf einen weiteren Field-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Werte des referenzierten Felds übernommen.

<datafield>

Dieser Tag repräsentiert ein Datenfeld eines Workflowprozesses.

Kann enthalten sein in	Kann enthalten
<process>	Wert als Klartext oder Jelly-Kontext-Variable, z. B. ${myVariable}

Attribute:

§ type: Typ des Datenfelds (STRING, INTEGER, DATE).

§ key: Bezeichner des Datenfelds

§ id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.

§ ref (optional): Verweis auf einen weiteren Datafield-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Attribute und der Wert des referenzierten Felds übernommen.

<table>

Mit diesem Tag geben Sie eine Tabelle an.

Kann enthalten sein in	Kann enthalten
<folder> <register> <document> <object>	<row>

Attribute:

§ key: Bezeichner der Tabelle.

§ keyType (optional): 'name' oder 'internal_name' (Standard)

§ id (optional): ID für den Zugriff innerhalb des Jelly-Kontexts.

§ ref (optional): Verweis auf einen weiteren Table-Tag innerhalb desselben Jelly-Kontexts. Hierbei werden alle Attribute und der Wert der referenzierten Tabelle übernommen.

<row>

Mit diesem Tag geben Sie eine Zeile in einer Tabelle an.

Kann enthalten sein in	Kann enthalten
<table>	<field>

Prozessoren

<e-mail>

Dieser Tag definiert einen E-Mail-Prozessor, der das automatische Auslesen von Metadaten von E-Mails ermöglicht.

Kann enthalten sein in	Kann enthalten
<table>	<field>

Attribute:

§ id: ID für den Zugriff innerhalb des JellyContexts.

§ file (optional): Name, unter dem eine oder mehrere einzufügende Dateien an das DropTarget übergeben wurden. Wenn nicht angegeben, wird der Standardname verwendet.

Metadaten:

§ TO

§ FROM

§ CC

§ BCC

§ SUBJECT

§ SENT_DATE

Unterstützte Dateiformate:

§ ima

§ eml

§ msg

<pdf>

Dieser Tag definiert einen PDF-Prozessor, der das automatische Auslesen von Metadaten von PDF-Dateien ermöglicht.

Attribute:

§ id: ID für den Zugriff innerhalb des Jelly-Kontexts.

§ file (optional): Name, unter dem die einzufügenden Dateien an das DropTarget übergeben wurden. Wenn nicht angegeben, wird der Standardname verwendet.

Metadaten:

Unterstützte Dateiformate:

§ pdf

§ pdfa

<image>

Dieser Tag definiert einen Image-Prozessor, der das automatische Auslesen von Metadaten von Bild-Dateien ermöglicht.

Attribute:

§ id: ID für den Zugriff innerhalb des Jelly-Kontexts.

§ file (optional): Name, unter dem die einzufügenden Dateien an das DropTarget übergeben wurden. Wenn nicht angegeben, wird der Standardname verwendet.

Metadaten:

§ NAME

§ CREATION_DATE

§ GEO_LOCATION

Unterstützte Dateiformate:

§ 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>

Dieser Tag definiert einen AutoDetection-Prozessor, der versucht die Metadaten jeder Datei auszulesen.

Attribute:

§ id: ID für den Zugriff innerhalb des Jelly-Kontexts.

§ file (optional): Name, unter dem die einzufügenden Dateien an das DropTarget übergeben wurden. Wenn nicht angegeben, wird der Standardname verwendet.

§ documentTypes (optional): Die Dateitypen kommasepariert festlegen, auf die der Prozessor beschränkt werden soll. Wenn nicht angegeben, werden alle Dateien des Jelly-Kontexts eingelesen.

Im folgenden Beispiel werden die Dokumenttypen definiert.

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

Metadaten:

§ eine Übersicht über alle auslesbaren Metadaten finden Sie unter folgender Adresse:

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

Das Referenzieren auf eine Variable erfolgt direkt über den Jelly-Kontext:

```
<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>
```

Beschreibung

Die enaio® apps zeigen die Texte der Beschreibungs-Tags, sofern definiert, standardmäßig an.

Tags aus diesem Bereich sind Teil der OS-Tag-Library und müssen deshalb ohne Namensbereich angegeben werden.

<description>

Dieser Tag dient als Container für alle weiteren Beschreibungs-Tags.

Kann enthalten sein in	Kann enthalten
<root>	<title> <longtitle> <helptext> <cabinet> <field>

<title>

Dieser Tag gibt den Namen des DropTargets an.

Kann enthalten sein in	Kann enthalten
<description>	Name des DropTargets

<longtitle>

Dieser Tag gibt einen Hilfetext für das DropTarget an.

Kann enthalten sein in	Kann enthalten
<description>	Beschreibung des DropTargets

<helptext>

Dieser Tag gibt die Beschreibung des DropTargets an.

Kann enthalten sein in	Kann enthalten
<description>	Hilfetext zum DropTarget

<cabinet>

Dieser Tag definiert den Schrank, der beim Parsen der Objektdefinition benutzt werden soll.

Kann enthalten sein in	Kann enthalten
<description>	<object>

Attribute:

§ key: Bezeichner des Schranks

§ keyType (optional): 'name' oder 'internal_name' (Standard)

<object>

Dieser Tag definiert das Objekt, der beim Parsen der Objektdefinition benutzt werden soll.

Kann enthalten sein in	Kann enthalten
<cabinet>	<field>

Attribute:

§ key: Bezeichner des Objekts

§ keyType (optional): 'name' oder 'internal_name' (Standard)

<field>

Dieser Tag definiert das Objekt, der beim Parsen der Objektdefinition benutzt werden soll.

Kann enthalten sein in	Kann enthalten
<object> <description>	Name der Variable, die angezeigt werden soll

Attribute:

§ key: Bezeichner des Felds

§ keyType (optional): 'name' oder 'internal_name' (Standard)

Weitere Attribute (wenn direkt unter dem Tag <description>):

§ required: true oder false

§ type: Feldtyp

§ size: Länge des Felds

Serverjobs und Optionen

Die folgenden Tags stellen Serverjobs dar.

<search>

Dieser Tag führt eine kombinierte Suche durch. Es können 1 bis n zu suchende DMS-Objekte angegeben werden, wobei das erste das Ergebnisobjekt definiert.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ SEARCH 1 Mal (Standard): DMS-Objekt, nach dem gesucht werden soll.

§ COMBINED_SEARCH n Mal: weitere DMS-Objekte aus demselben Schrank, die die Suche eingrenzen

§ LOCATION 0/1 Mal: Standort des gesuchten DMS-Objekts. Hierbei muss es sich um einen Ordner oder Register handeln, bei dem die OSID gesetzt ist. Das Objekt ist nicht geeignet für die Suche nach Ordnern.

Attribute:

§ id: Bezeichner, mit dem innerhalb des Jelly-Kontexts auf die Ergebnisse der Suche zugegriffen werden kann.

§ mode (optional): Suchmodus, folgende Auswahl ist möglich:

  § FIRST (Standard): Nur der erste Treffer wird zurückgeliefert.

  § ALL: Alle Treffer werden zurückgeliefert.

  § EXACTLY_ONE: Es wird ein Fehler geworfen, wenn es mehr als einen Treffer gibt. Ansonsten wird der Treffer zurückgeliefert.

  § AT_LEAST_ONE: Es wird ein Fehler geworfen, wenn es keine Treffer gibt. Ansonsten werden alle Treffer zurückgeliefert.

<update>

Dieser Tag aktualisiert ein DMS-Objekt. Das Attribut OSID muss gesetzt sein.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ UPDATE 1 Mal (Standard): DMS-Objekt, das aktualisiert soll.

Attribute:

§ file (optional): Name, unter dem eine einzufügende Datei an das DropTarget übergeben wurde. Bei der Übergabe von mehreren Dateien werden mehrere Namen angegeben. Falls nicht angegeben, wird der Standardname verwendet.

<insert>

Dieser Tag fügt ein DMS-Objekt an einen Standort ein. Das erste angegebene DMS-Objekt wird eingefügt. Wenn es sich dabei nicht um einen Ordner handelt, muss über ein zweites DMS-Objekt der Standort, also ein Ordner oder Register definiert werden.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

- Beim Einfügen eines Ordners
  - UPDATE 1 Mal (Standard): Der Ordner, der eingefügt werden soll.
- Beim Einfügen eines Registers oder Dokuments
  - UPDATE 1 Mal (Standard): Register oder Dokument, das eingefügt werden soll.
  - LOCATION 1 Mal: Zielstandort (Ordner oder Register)

Attribute:

- file (optional): Name, unter dem eine einzufügende Datei an das DropTarget übergeben wurde. Bei der Übergabe von mehreren Dateien werden mehrere Namen angegeben. Falls nicht angegeben, wird der Standardname verwendet.

<move>

Dieser Tag verschiebt ein Register oder Dokument an einen anderen Standort. Das erste angegebene DMS-Objekt wird verschoben, das zweite ist der Zielstandort, also ein Ordner oder Register.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

- MOVE 1 Mal (Standard): Register oder Dokument, das verschoben werden soll.
- LOCATION 1 Mal: Zielstandort (Ordner oder Register)

<copy>

Dieser Tag kopiert ein Register oder Dokument an einen anderen Standort. Das erste angegebene DMS-Objekt wird kopiert, das zweite ist der Zielort, also ein Ordner oder Register.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ MOVE 1 Mal (Standard): Register oder Dokument, das kopiert werden soll.

§ LOCATION 1 Mal: Zielstandort (Ordner oder Register)

<link>

Dieser Tag erstellt eine Verweiskopie. Intern wird das Tag <copy> mit `LINKDOCUMENT=1` aufgerufen.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ MOVE 1-mal (Standard): Objekt, das kopiert werden soll.

§ LOCATION 1 Mal: Zielstandort (Ordner oder Register)

<delete>

Dieser Tag löscht ein angegebenes DMS-Objekt. Das Attribut `osid` muss gesetzt sein.

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ DELETE 1 Mal (Standard): DMS-Objekt, das gelöscht werden soll.

<start>

Dieser Tag startet einen neuen Workflowprozess.

Kann enthalten sein in	Kann enthalten
<root> <cabinet>	<process>

<options>

Mit diesem Tag können Sie weitere Optionen an einen Serverjob übermitteln.

Kann enthalten sein in	Kann enthalten
<search> <update> <insert> <move> <copy> <delete> <select>	<option>

<option>

Mit diesem Tag definieren Sie den Wert einer Option (true oder false).

Kann enthalten sein in	Kann enthalten
<options>	Wert der Option (true oder false)

Attribute:

§ key: Bezeichner der Option (siehe auch Handbuch enaio® server-api).

<logger>

Dieser Tag ermöglicht es, aus einem Jelly-Skript heraus ins Protokoll zu schreiben.

Kann enthalten sein in	Kann enthalten
allen Tags aus dem OS-Namensraum	Text, der ins Protokoll geschrieben werden soll.

Attribute:

§ level: Log-Level der Einträge, die ins Protokoll geschrieben werden (siehe auch Commons Logging – Message Priority Levels).

<return>

Dieser Tag ermöglicht es, Werte aus einem Jelly-Kontext zu exportieren.

Kann enthalten sein in	Kann enthalten
allen Tags aus dem OS-Namensraum	Wert, der exportiert werden soll.

Attribute:

§ id (optional): Bezeichner, unter dem ein Wert exportiert werden soll. Falls nicht angegeben, wird der Standardbezeichner (DEFAULT_RETURN_VALUE) verwendet.

Kombinierte Serverjobs

<select>

Dieser Tag führt drei Funktionen aus:

1. Suche nach einem DMS-Objekt
2. Erstellen des DMS-Objekts, wenn 1. kein Ergebnis geliefert hat
3. Aktualisieren des DMS-Objekts, wenn 1. ein Ergebnis geliefert hat

Kann enthalten sein in	Kann enthalten
<cabinet>	<folder> <register> <document> <object>

Benötigte Objekte:

§ SEARCH 1 Mal: DMS-Objekt, das gesucht werden soll.

§ COMBINED_SEARCH n-mal: Weitere DMS-Objekte aus demselben Schrank, die die Suche eingrenzen.

§ INSERT 1 Mal: DMS-Objekt, das eingefügt werden soll.

§ UPDATE 1 Mal: DMS-Objekt, das aktualisiert werden soll.

§ LOCATION 1 Mal: Standort (Ordner oder Register) für die Suche und das Einfügen

Attribute:

§ id: Bezeichner, mit dem innerhalb des Jelly-Kontexts auf das gefundene oder erstellt DMS-Objekt zugegriffen werden kann.

§ file (optional): Name, unter dem eine einzufügende Datei an das DropTarget übergeben wurde. Bei der Übergabe von mehreren Dateien werden mehrere Namen angegeben. Falls nicht angegeben, wird der Standardname verwendet.

§ create (true (Standard) oder false): Gibt an, ob ein DMS-Objekt erstellt werden soll, wenn 1. kein Ergebnis geliefert hat.

§ update (true (Standard) oder false): Gibt an, ob das Ergebnis von 1. aktualisiert werden soll.

DropTarget-Werkzeuge

<utils:DateFormat>

Mit diesem Tag können Datumsangaben formatiert werden.

Der Tag kann keine anderen Tags enthalten und ist in der Regel auch in keinem besonderen Tag enthalten. Idealerweise wird er innerhalb der obersten Ebene (<jelly>) oder in einem <cabinet>-Tag eingesetzt.

Attribute:

- id: Bezeichner, mit dem innerhalb des Jelly-Kontexts auf das Ergebnis der Formatierung zugegriffen werden kann.
- inputPattern: Muster, nach den Regeln der java.text.SimpleDateFormat-Klasse, das beim Einlesen eines Datums im Textformat zugrunde gelegt wird.
- parsePosition: Nullbasierte Position, ab wann mit dem Einlesen des Eingangsdatums in Textformat begonnen werden soll.
- pattern: Frei definierbares Muster entsprechend den Regeln der java.text.SimpleDateFormat-Klasse, das beim Formatieren zugrunde gelegt wird.
- value: Datumswert, der formatiert werden soll. kann als Zeichenkette oder java.util.Date gesetzt werden.

<utils:RegexFormat>

Mit diesem Tag können reguläre Ausdrücke formatiert werden.

Der Tag kann keine anderen Tags enthalten und ist in der Regel auch in keinem besonderen Tag enthalten. Idealerweise wird er innerhalb der obersten Ebene (<jelly>) oder in einem <cabinet>-Tag eingesetzt.

Attribute:

- id: Bezeichner, mit dem innerhalb des Jelly-Kontexts auf das Ergebnis der Formatierung zugegriffen werden kann.
- pattern: Regulärer Ausdruck
- value: Text, der formatiert werden soll.

Um auf die Ergebnisse der Formatierung zugreifen zu können, müssen im regulären Ausdruck Gruppen definiert werden. Die Ergebnisse sind dann über Nummer der jeweiligen Gruppe erreichbar.

DropTargets testen

Es wird empfohlen, die Funktionsfähigkeit eines DropTargets vor dem Produktiveinsatz in einem Testsystem zu testen.

Dafür können folgende Tools hilfreich sein:

- curl.exe – Das Kommandozeilentool kann als Testclient verwendet werden, um ein DropTarget per URL-Aufruf auszuführen. Es wird vom enaio®-Setup installiert und befindet sich im Verzeichnis …`\server`.
- JSONView-Addon für den Browser – Das Firefox-AddOn stellt JSON-Dokumente wie XML-Dokumente dar. Die Anzeige verfügt über Formatierungen sowie Hervorhebungen und Felder und Objekte können erweitert und zusammengefasst werden.
- enaio® enterprise-manager – Diese enaio®-Komponente dient der Verwaltung der enaio®-Server, Archivmedien, Lizenzen usw. Im Bereich **Erweiterte Administration > Überwachung > Jobaufrufe** können alle Jobs an enaio® server überwacht werden.

Pushnotificationservice für enaio® apps

Als Dienst zur Pushbenachrichtigung ist der Pushnotificationservice (PnS) Bestandteil von enaio® appconnector und wird für die Zustellung von Nachrichten eingesetzt.

Der PnS prüft regelmäßig, ob neue Nachrichten über Abonnements, Wiedervorlagen oder Workflowarbeitsschritte vorliegen. Die App zeigt dann eine Meldung über jede neu eingegangene Nachricht an, auch wenn die App nicht aktiv ist. Die Anzeige der Meldung können Benutzer in der App individuell einstellen.

Der PnS steht nur für enaio® app unter iOS zur Verfügung.

Konfiguration

Um den Pushnotificationservice zu konfigurieren, bearbeiten Sie die Konfigurationsdatei `osrest.properties` im Programmverzeichnis …`\services\OS_AppConnector\configuration`.

Nach der Konfiguration muss der enaio® appconnector neu gestartet werden.

Schlüsselname	Beschreibung
`services.pushnotification.enabled`	Um den PnS zu aktivieren, setzen Sie den Wert auf 'true'.
`services.pushnotification.proxy.enabled`	UrbanAirship wird via Internet angebunden. Wenn zwischen enaio® appconnector und dem Internet ein Proxyserver steht, setzen Sie diesen Parameter auf 'true'.
`services.pushnotification.proxy.address`	Ist der Proxyserver aktiviert, wird hier die Adresse oder IP des Proxyservers eingetragen. Beispiel: `proxy.meine-firma.de`

`services.pushnotification.proxy.port`	Ist der Proxyserver aktiviert, wird hier der Port, unter dem der Proxyserver auf der angegebenen Adresse erreichbar ist, eingetragen. Beispiel: `3128`
`services.pushnotification.proxy.username`	Ist der Proxyserver aktiviert und benötigt einen Anmeldebenutzernamen, so wird dieser hier eingetragen.
`services.pushnotification.proxy.password`	Ist der Proxyserver aktiviert und benötigt ein Anmeldepasswort, so wird dieses hier eingetragen.
`services.pushnotification.production`	Die Voreinstellung 'true' muss für Benachrichtigungen so belassen werden. In Entwicklungsumgebungen können über 'false' Benachrichtigungen aus- und Entwicklungsbenachrichtigungen eingeschaltet werden.

Um die Konfiguration zu testen, lösen Sie in der App oder in enaio® client beispielsweise ein Abonnement oder eine Wiedervorlage aus. Prüfen Sie in der App, ob Sie über diese Änderung automatisch benachrichtig werden, auch wenn die App nicht aktiv ist.

Informationen über eine fehlerhafte Konfiguration des PnS finden Sie im Protokoll von enaio® appconnector. Das Protokoll `osrest.log` befindet sich im Programmverzeichnis `logs`.

Anhang

Anbindung von enaio® appconnector mit .NET

Die folgenden Abschnitte geben einen technischen Überblick über die Technologien und Strategien zur Anbindung von enaio® appconnector mit Hilfe von .NET über die REST-Schnittstelle.

Beachten Sie, dass Entwickler für die Anbindung Erfahrung in der .NET-Programmierung benötigen.

Möglichkeiten der Anbindung

enaio® appconnector ist als Rest-WebService mit dem Namen `OSRest` implementiert. Rest-WebServices stellen ihre Funktionen durch den Aufruf speziell geformter URLs zur Verfügung. Die Ergebnisse der Funktionen werden als Webserver-Antwort bereitgestellt.

Um enaio® appconnector mit .NET anzubinden, stehen folgende Strategien zur Verfügung: Sie können sowohl die Klasse `WebClient` als auch die Klasse `WebRequest` verwenden. Beide Klassen finden Sie im Namensraum `System.Net` des Frameworks.

Weitere Informationen zu den Klassen entnehmen Sie der MSDN (Microsoft Developer Network) Dokumentation.

Aufruf mit der Klasse 'WebClient'

Der Aufruf mit Hilfe der Klasse `WebClient` ist sehr komfortabel. Der Vorteil dieser Klasse besteht darin, dass die Funktionalität bereits in Methoden gekapselt ist. Nachteilig ist jedoch, dass Sie einige Einstellungen, wie z. B. benutzerdefinierte HTTP-Header, nicht besonders detailliert vornehmen können.

Das folgende Beispiel beschreibt einen `WebClient`-Aufruf:

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

Auf die im Beispiel gezeigte Weise wird die Auflistung aller Dokumenttypen mit nur wenigen Zeilen ermöglicht.

Es gibt auch noch andere Funktionen die Sie verwenden können, wie z. B. :

- UploadString
- OpenRead
- DownloadData
- DownloadFile

Weitere Informationen zu diesen Funktionen entnehmen Sie der MSDN-Dokumentation.

Aufruf mit der Klasse 'WebRequest'

Beim Aufruf mit der Klasse `WebRequest` müssen Sie die Antwort als gesonderten Stream behandeln. Der Vorteil dieser Klasse besteht darin, dass sie die komplette Kontrolle über die Kommunikation erlaubt. Nachteilig ist jedoch der mit dieser Klasse einhergehende erhöhte Programmieraufwand.

Das folgende Beispiel beschreibt einen `WebRequest`-Aufruf:

```
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());
```

Auch in diesem Beispiel werden als Ergebnis alle Dokumenttypen aufgelistet.

HTTP-Multipart-Aufruf

Zum Senden von Feldern inklusive Datei an ein DropTarget verwenden Sie einen HTTP-MultiPart-Aufruf.

Der Aufruf benötigt zwei Teile: den JSON-Inhalt, der mit `data` bezeichnet wird, und die Datei, die mit `file` bezeichnet wird.

Da es im .NET Framework keine geeignete Methode gibt, um einen HTTP-Multipart-Aufruf auszuführen, beschreibt das folgende Quelltextbeispiel solch einen Aufruf:

```
        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);
        }
```

Authentifizierung

Um sich mit enaio® appconnector erfolgreich zu verbinden, müssen Sie sich authentifizieren.

Hierfür stellt Ihnen enaio® appconnector folgende Verfahren zur Verfügung:

- Negotiate
- NTLM
- Basis-Authentifizierung

Um eine Authentifizierung zu ermöglichen, müssen Sie die Eigenschaft `Credentials` für das `WebClient`- bzw. das `WebRequest`-Objekt explizit setzen.

NTLM-Authentifizierung

Um eine Authentifizierung über NTLM zu ermöglichen, müssen Sie die Eigenschaft `Credentials` auf den Wert `CredentialCache.DefaultCredentials` setzen. Dadurch wird das Konto des aktuell angemeldeten Windows-Benutzers als Anmeldeinformation verwendet.

Wenn Sie den Aufruf aus einer ASP.NET Anwendung heraus tätigen, müssen Sie den Wert auf `CredentialCache.DefaultNetworkCredentials` setzen.

Weitere Informationen hierzu entnehmen Sie der MSDN-Dokumentation.

Basis-Authentifizierung

Um eine Basis-Authentifizierung zu ermöglichen, müssen Sie den Wert für die Eigenschaft `Credentials` auf den Wert `new NetworkCredential("user", "password")` setzen.

Wenn auf Ihrem System eine NTLM- oder eine Negotiate-Authentifizierung möglich ist, so wird diese auch durchgeführt. Die Basis-Authentifizierung müssen Sie gegebenenfalls erzwingen (siehe 'Erzwingen bestimmter Modi für die Authentifizierung').

Erzwingen bestimmter Modi für die Authentifizierung

Das .NET Framework stellt Ihnen verschiedene Authentifizierungsmechanismen zur Verfügung. Diese können Sie mit folgendem Beispiel auflisten:

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

Dabei werden alle Mechanismen nach Verfügbarkeit in absteigender Reihenfolge angewendet.

Wenn Sie einen Authentifizierungsmechanismus für enaio® appconnector erzwingen möchten, wie z. B. die Basis-Authentifizierung, so müssen Sie die anderen Mechanismen explizit deaktivieren bzw. entfernen.

Das folgende Beispiel demonstriert dieses für das Erzwingen der Basis-Authentifizierung:

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

Beachten Sie, dass es sich hierbei um einen statischen Aufruf handelt. Wenn Sie in Ihrer Anwendung zuvor deaktivierte Authentifizierungsmechanismen wieder anwenden wollen, so müssen Sie diese aktivieren. Dies erreichen Sie durch die Methode `Register`.

Behandeln von JSON-Antworten von enaio® appconnector

Sie erhalten von enaio® appconnector Ergebnisse als JSON-formatierte HTTP-Inhalte. Wenn Sie dieses JSON-Format in .NET-lesbare Daten umwandeln möchten, stehen Ihnen folgende Strategien zur Verfügung:

§ Serialisierung durch das 'DataContract

§ Umwandlung in XML und Parsen mit Hilfe von XPath

Serialisierung durch das 'DataContract'-Attribut

Sie können ein entsprechendes .NET-Objekt mit dem `DataContract`-Attribut deklarieren und in der Klasse entsprechende `DataMember`-Eigenschaften festlegen. Sie finden diese Attribute in dem Namensraum `System.Runtime.Serialization`. Damit Sie diesen Namensraum verwenden können, müssen Sie Ihrem Projekt einen Verweis auf die Assembly `System.Runtime.Serialization` aus dem .NET-Framework hinzufügen. Anschließend können Sie durch die Verwendung der Klasse `DataContractJsonSerializer` die JSON-Ergebnisse einfach in .NET-Objekte umwandeln.

Weitere Informationen hierzu entnehmen Sie der MSDN-Dokumentation.

Umwandlung in XML und Parsen mit Hilfe von XPath

Eine andere Möglichkeit, um die Ergebnisdaten im JSON-Format für Ihre Anwendung nutzbar zu machen, besteht darin, diese zunächst in XML umzuwandeln und anschließend zu parsen oder mit XPath-Ausdrücken auszuwerten.

Das folgende Beispiel beschreibt dieses Vorgehen:

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

Beachten Sie, dass auch für dieses Beispiel im Projekt auf die .NET Framework-Assembly `System.Runtime.Serialization` verwiesen werden muss.

API-Dokumentation

Allgemeines

OSRest stellt ein auf dem HTTP-Protokoll basierendes API zur Verfügung, alle Aufrufe sind zustandlos und resourcenorientiert (REST). Das API versucht komplexe Funktionsaufrufe in enaio® zu bündeln und zu vereinfachen. Dazu gehört ein Mapping aller Indexdaten auf ein jeweilig für die konkrete Anwendung sinnvolles Schema.
Da als Protokoll HTTP verwendet wird, ist bei allen Aufrufen auf ein korrektes URL-Encoding zu achten.

Authentifizierung

OSRest unterstützt Standard HTTP-Authentifizierung, dabei können folgende Verfahren verwendet werden:

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

Bei den Windows-basierten Authentifizierungsverfahren NTLM und SPNEGO wird die Authentifizierung an das unterlegende Windows Betriebssystem des OSRest-Servers delegiert. Die alternative HTTP Basic Authentication kann wahlweise Windows oder interne enaio®-Konten verwenden. Sollte die Anwendung es erfordern, kann OSRest auch für einen Betrieb ohne Authentifizierung mit einem technischen Benutzer konfiguriert werden. Die URL zur Admin-Oberfläche kann nur mit enaio®-Supervisor-Rechten aufgerufen werden.

Services

OSRest stellt verschiedene Services zur Verfügung:

- DocumentService (/documents) */osrest/api/documents* - Methoden zur Suche von Indexdaten
- DocumentFileService (/documentfiles) */osrest/api/documentfiles* - Methoden zum Umgang mit Dokumentendateien
- NotificationService (/notifications) */osrest/api/notifications* - Zugriff auf enaio®-Benachrichtigungen
- SessionService (/session) */osrest/api/session* - Informationen zur Benutzersitzung
- ServiceInfoService (/serviceinfo) */osrest/api/serviceinfo* - Allgmeine Informationen zum OSRest-Service
- OSFileService (/anon) */osrest/api/anon* - Anonymer Service zur Erzeugung von enaio®-internen Linkdateien
- WorkflowService (/workflows) */osrest/api/workflows* - Methoden zum Starten und Bearbeiten von Workflows
- ObjDefService (/objdef) */osrest/api/objdef* - Methoden für das Arbeiten mit der Objektdefinition in JSON-Form
- Organization Service (/organization) */osrest/api/organization* - Methoden für Benutzer und Gruppen sowie den E-Mail-Versand
- IconService */osrest/api/icon* - Methoden zum Abrufen von Katalog-Icons

Alle Methoden liefern standardmäßig in JSON notierte Resultate zurück.

Ergebnisse

OSREST liefert drei Arten von Ergebnissen:

- Trefferlisten
- Benachrichtigungslisten
- gespeicherte Anfragen

Trefferlisten

```
{
      -
      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
                  }, ...
             ]
        }
}
```

Das *documentResult* ist das root-Element einer Trefferliste. Es enthält die Informationen *pagesize*, *startposition* (offset) und *totalHits* sowie eine Liste von *documents*.
Das *documents*-Element repräsentiert die Treffer aus enaio®. Ein *document* hat die Attribute *id*, *type*, *fields* und *fav*.
id: OSID des Treffers
type: Dokumenttyp des Treffers. Mögliche Werte: *FOLDER*, *REGISTER*, *DOCUMENT*
fields: Liste von gemappten Indexdaten des Treffers (siehe auch: Metadaten-Mapping)
fav: Flag, das angibt, ob sich der Treffer im Favoritenordner befindet.

Benachrichtigungslisten

```
notifications: [
      -
      {
           id: "9015"
           type: "DOCUMENT"
           notificationType: "REVISIT"
```

```
            eventDate: 1282654839000
             -
             fields: {
                  title: "Hugo Distler"
                  info: ""
             }
      },...
]
```

nofiications ist das root-Element der Benachrichtigungsliste. Es enthält eine Liste von Benachrichtigungen.
Eine Benachrichtigung enthält die Informationen *id*, *type*, *notificationType*, *eventDate* und *fields*.
id: OSID des Treffers
type: Dokumenttyp des Treffers. Mögliche Werte: *FOLDER*, *REGISTER*, *DOCUMENT*
notificationType: Art der Benachrichtigung. Mögliche Werte: REVISIT, SUBSCRIPTION, WORKFLOW
eventDate: Datum der Benachrichtigung
fields: Liste von gemappten Indexdaten des Treffers (siehe auch: Metadaten-Mapping)

```
{
      -
      storedqueries: [
            -
            {
                  id: 14004695
                  name: "offene Supportcalls"
                  queryParams: [ ]
            }
            -
            {
                  id: 14041962
                  name: "E-Mails"
                  -
                  queryParams: [
                        -
                        {
                              object: "Email"
                              field: "Datum:"
                              dataType: "DATE"
                        }
                  ]
            },...
      ]
}
```

storedqueries ist das root-Element. Es enthält eine Liste von gespeicherten Anfragen.
Eine gespeicherte Anfrage enthält die Informationen *id*, *name* und *queryParams*.
id: OSID der gespeicherten Anfrage
name: Name der gespeicherten Anfrage
queryParams: Liste der Parameter der gespeicherten Anfrage (falls vorhanden)

JS-Scripting Support (intern)

Mit der Umstellung auf Java 8 wird nun auch Scripting unterstützt. Das ermöglicht es, den Rest-Service zur Laufzeit zu erweitern. Voraussetzung ist, dass es der Rest-Aufruf unterstützt, daher wird es eher fließend so nach und nach eingeführt und auch an den entsprechenden Aufrufen erkenntlich gemacht. Mehr dazu später.

MetadatenMapping

Bei dem Metadatenmapping geht es um die Anreicherung von Dateimetadaten in die Indexdaten durch den enaio ExtractionService. Eine in JSON formatierte Mappingdatei im Schema-Verzeichnis legt fest, welche Werte aus dem ExtractionService an welche Indexdatenfelder gebunden werden.

Aufruf aus dem OSRest heraus

Gibt 2 Geschmacksrichtungen des Aufrufs

- Einmal mit Angabe einer ObjectId eines Objekts mit 2 Optionalen Parametern (GET, mimetype: application/json). Hier wird quasi auf ein bestehendes Objekt das Mapping angewendet.
  - Override = „gibt an, ob die „override“ Angabe im Mapping ignoriert werden soll und immer alle Felder gemappt werden sollen
  - Preview = „gibt an, ob das Mapping als JSON zurückgeliefert werden soll. `Wird es nicht angegeben oder auf false gesetzt, wird das Dokument mit dem Mapping aktualisiert (XMLUpdate) und nur ein OK zurück geliefert. (Standardverhalten)`
  - `../documentfiles/processmetadata/[id][?override=[true|false]&preview=[true|false]]`

- Der zweite Aufruf heißt genauso, erfüllt aber einen anderen Zweck. Hier können Dokumente als Multipart gesendet werden, die nicht im enaio vorhanden sind, welche zusammen mit einer ObjectTypeId quasi ein Mapping-Vorschlag macht. Dieser Vorschlag kommt per application/json zurück. Nochmal die Unterschiede zum vorherigen Request im Überblick:
  - POST Request
  - Multipart
  - Response ist application/json
  - `../documentfiles/processmetadata/[objectTypeId]`

Konfiguration OSRest für ExtractionService

In der osrest.properties wird die URL zum ExtractionService hinterlegt. Der Eintrag sieht wie folgt aus:
extractionservice.url=http\://10.10.0.31:9303/extraction

Aufbau Mapping-Datei

Im Schema-Verzeichnis des OSRest wird eine Datei „extraction.mapping.config“ erwartet. Es ist eine Textdatei, die Konfiguation selbst ist in JSON gehalten.

Weitere Merkmale:

- Innerhalb der Dokumenttypen, können Mappings für unterschiedliche mimetypen erstellt werden.
- Das Property „override“ gibt an, ob (=false) nur die Leeren Indexdatenfelder mit dem Mapping befüllt werden sollen oder (=true) ob vorhanden Indexdaten mit dem Mapping überschrieben werden soll. Dies kann im Request auch nochmal direkt übersteuert werden (GET-Requestvariante)
- Die Feldmappings selbst sind Arrays mit einem oder mehreren ExtractionService-Feldern, da nicht immer dieselben Felder ausgeliefert werden, kann man hier quasi sich eine Art Fallback schaffen, wenn mal ein Wert nicht dabei sein sollte.
- Der Bereich Variablen ist direkt erstmal nicht für das Mapping wichtig sondern ist für den JavaScript-Teil interessant. Hier können analog wie zu Indexdatenfeldern, extraction-Werte mit Fallback gemappt werden und diese werden dann u.a. in das JavaScript weitergereicht. Dort können diese Infos dann ausgewertet und verarbeitet werden. (wenn z.B. bestimmte Katalogwerte gesetzt werden sollen, abhängig von div. Extraction-Werten oder sie in einer bestimmten Art formatiert werden sollen (siehe GeoData für Dashlet)). Wichtig: Indexfelder die damit befüllt werden sollen, müssen zumindest in der darüber liegenden Sektion vorhanden sein.
- Datums-Werte automatisch, soweit möglich, in enaio-freundliches deutsches Datumsformat gewandelt

Folgend ein Beispiel (von der Cebit) (Nicht wundern, ist für die E-Mail mit Leerzeichen formatiert)

```
{
  "262199": {  ß ObjectTypeId des Dokuments, Registers oder
Schranks
    "override": false, ß gibt an, ob alle Mappingfelder
übernommen werden sollen.
    "mimeTypeMappings": [{
      "mimeTypes": ["image/*"], ß MimeType Einschränkung.
      "mappings": {
        "picture_type": ["OS:FileType"],
        "picture_width_px": ["OS:ImageWidth",
"File:ImageWidth", "ExifIFD:ExifImageWidth"],
        "picture_height_px": ["OS:ImageHeight",
"File:ImageHeight", "ExifIFD:ExifImageHeight"],
        "horizontal_resolution": ["JFIF:XResolution",
"IFD0:XResolution", "Photoshop:XResolution"],
        "vertical_resolution": ["JFIF:YResolution",
"IFD0:YResolution", "Photoshop:YResolution"],
        "picture_orientation": [""],
        "colour_modus": [""],
        "colour_depth": ["OS:BitsPerSample",
"File:BitsPerSample"],
        "picture_filesize": ["OS:FileSize", "System:FileSize"],
        "picture_date": ["OS:CreateDate",
"System:FileCreateDate"],
        "geodata": [""] ß leerfeld, damit es per Javascript
gesetzt werden kann
      },
      "variables": {
        "GPS_LONG": ["OS:GPSLongitude", "GPS:GPSLongitude",
"Composite:GPSLongitude"],
        "GPS_LAT": ["OS:GPSLatitude", "GPS:GPSLatitude",
"Composite:GPSLatitude"]
```

```
      }
    }]
  },
  "262144": {
    "override": false,
    "mimeTypeMappings": [{
      "mimeTypes": ["application/pdf"],
      "mappings": {
        "Dateityp": ["OS:FileType", "File:FileType"],
        "Bearbeiter": ["OS:Author", "PDF:Author"],
        "Beschreibung": ["${'Erstellt mit ' + OS_Creator}"]
      }
    }]
  }
}
```

Die JavaScript-Datei

Für die Verarbeitung per JavaScript wird im ExtractionService-Fall eine Datei namens „extraction.processing.js“ im Schema Verzeichnis erwartet. Das JS wird (derzeit) für jedes Indexdatenfeld einzeln aufgerufen.

Da wir nun in der Lage sind, per Script OSRest Funktionen anzureichern (analog zu den VBScript Events in enaio), kann es durchaus sein, dass es später einen separaten Ordner für die Skripte gibt. Aktuell ist JS-Support nur für Extraction implementiert

Folgende Objekte sind aktuell in diesem Skript vorhanden:

- `ObjectTypeId` : ObjectTypeId des Objektes.
- `FieldName` : aktueller interner Name des Feldes
- `FieldValue` : aktueller Feldwert
- `ResultValue` : Wert, der nach der Verarbeitung diesem Feld zugewiesen wird.
- `os_map` : Key-Value Map mit allen gemappten Indexdatenfeldern, die in der Mapping-Datei definiert wurden
- `var_map` : Key-Value Map mit den Variablen, die in der Mapping-Datei definiert wurden
- `es_map` : Key-Value Map mit allen ExtractionService-Werten. ACHTUNG Da die Werte aus dem Service ":"-Zeichen enthalten, werden diese, um sie im JavaScript nutzungsfähig zu bekommen, durch "_"-Zeichen ersetzt. Also "OS:FileType" wird im JavaScript mit "OS_FileType" verwendet.

Folgend das passende JS zu dem Mapping.

```
if (this.ObjectTypeId == 262199) {
    switch (this.FieldName) {
        case "picture_type":
            {
                switch (this.FieldValue) {
                    case "JPEG":
                        this.ResultValue = "JPG";
                        break;

                    case "TIFF":
```

```
                            this.ResultValue = "TIF";
                            break;
                    }
                    break;
                }

            case "picture_orientation":
                {
                    if (this.os_map.picture_width_px >
this.os_map.picture_height_px) {
                        this.ResultValue = "Querformat";
                    } else if (this.os_map.picture_width_px <
this.os_map.picture_height_px) {
                        this.ResultValue = "Hochformat"
                    } else {
                        this.ResultValue = "Anderes Format";
                    }
                                    break;
                }

            case "geodata":
                {
                    this.ResultValue = this.var_map.GPS_LAT + ", "
+ this.var_map.GPS_LONG;
                    break;
                }
        }
}
```

Comma-seperated list of object oids on which the search should be executed. If empty, all objects are searched (optional)

DocumentService (/documents)

Neben den Methoden zur Suche und Anzeige von Indexdaten bietet der DocumentService noch Funktionen zur Auszeichnung von Dokumentenfavoriten an. Diese Funktionen werden über eine spezielle konfigurierbare Mappe des Benutzers abgebildet, in der die als Favoriten markierten enaio®-Objekte verwaltet werden.

/osrest/api/documents

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine gekürzte Version der für den Benutzer gültigen Objektdefinition zurück.

/osrest/api/documents/cabinets

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller für den Benutzer sichtbaren Schränke zurück.
Optionale Parameter sind:

- keytype (String): Welche Bezeichner sollen zurückgegeben werden. Mögliche Werte: NAME, INTERNAL_NAME, OBJECT_TYPE_ID, DB_NAME

/osrest/api/documents/cabinets/[Schrank]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert alle Ordner aus dem Schrank mit dem gegebenen Namen.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- pagesize (Integer): maximale Anzahl der Treffer
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- page (Integer): angeforderte Seite für das Blättern durch die Treffer

/osrest/api/documents/storedqueries

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste der gespeicherten Anfragen zurück. Das Feld isIncomplete zeigt bei einer Anfrage an, dass nicht alle ursprünglichen variablen Felder in der Abfrage enthalten sind. Dies passiert dann, wenn ein

variables Feld der Anfrage im Nachgang aus der Objektdefinition entfernt wurde ohne den Server neuzustarten.
Optionale Parameter sind:

- showglobal (String): Ist der Parameter auf 'false' gesetzt, werden globale gespeicherte Anfragen ausgeblendet.
- foldering (Boolean): Ist der Parameter gesetzt und 'true', werden die Anfragen gemäß des Enaio Clients in Ordner strukturiert. Die Rückgabestruktur gleicht dann einem Baum und ist keine plane Liste mehr.
- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird das Cache ignoriert und die Anfragen neu vom Server abgefragt.

/osrest/api/documents/storedqueries/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert das Ergebnis der gespeicherten Anfrage mit der übergegebenen ID zurück. Für DATE-, TIME- und DATETIME-Felder kann auch <Datum, <=Datum, >Datum, >=Datum/Datum und Zeit/Zeit angegeben werden.
Eine Intervallsuche für Datum, Zeit und Datum/Zeit ist erst ab Enaio 8.1 möglich.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- pagesize (Integer): maximale Anzahl der Treffer für den Aufruf
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- page (Integer): angeforderte Seite für das Blättern durch die Treffer
- autostar (String): Autsternverhalten. Werte gleichen der Server-API-Werte: ("0" = Kein Autostern, "1" = *Anfrageparameter, "2" = Anfrageparameter*, "3" = *Anfrageparameter*). Wird kein Wert angegeben, wird der Wert aus der Clientkonfiguration des Benutzers übernommen.
- verbose (Boolean): Es wird ein erweitertes generische Metadatenmodell ausgegeben
- Anfrageparameter als HTTP-Queryparameter [objekt].[feld] (z. B. Email.Absender=Peter)

```
.../osrest/api/documents/storedqueries/1234?Person.Vorname=Gustav&Person.Nachname=Gans&metadata=MeinMapping
```

/osrest/api/documents/storedqueries/[id]

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert das Ergebnis der gespeicherten Anfrage mit der übergegebenen ID zurück. Statt der storedqueries/[id] GET-Methode werden hier direkt die VariablenNamen mit den gewünschten Suchwerten als JSON übergeben. Ansonsten sind die Methoden von ihrer Funktionsart her gleich. Für DATE-, TIME- und DATETIME-Felder kann auch <Datum,

<=Datum, >Datum, >=Datum/Datum und Zeit/Zeit angegeben werden. Eine Intervallsuche für Datum, Zeit und Datum/Zeit ist erst ab Enaio 8.1 möglich. Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- pagesize (Integer): maximale Anzahl der Treffer für den Aufruf
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- page (Integer): angeforderte Seite für das Blättern durch die Treffer
- autostar (String): Autsternverhalten. Werte gleichen der Server-API-Werte: ("0" = Kein Autostern, "1" = *Anfrageparameter, "2" = Anfrageparameter*, "3" = *Anfrageparameter*). Wird kein Wert angegeben, wird der Wert aus der Clientkonfiguration des Benutzers übernommen.
- verbose (Boolean): Es wird ein erweitertes generische Metadatenmodell ausgegeben

POST JSON Beispiel

```
{
	fields: {
		"var1": "Kunde*",
		"var2": "10.12.2015
	}
}
```

/osrest/api/documents/storedqueries/add

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Pflichtparameter sind:

- name (String): Name der Suchanfrage

Die Methode liefert die ID der gespeicherten Suchanfrage zurück. Im POST-Body muss eine Suchanfrage gemäß des '/osrest/api/documents/search' Aufrufes mitgegeben werden. Hier existiert jedoch aktuell die Einschränkung, dass bei den Listen-Basisparametern nur der erste Wert übernommen wird. Es kann also beispielsweise keine Veroderung von ArchiveState vorgenommen werden.

/osrest/api/documents/tray

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die Objekte aus der Benutzerablage zurück.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings

/osrest/api/documents/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die Indexdaten des Objektes mit der gegebenen ID. Optionale Parameter sind:

- objecttypeid (Integer): ObjektTyp ID des gesuchten Objekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- metadata (String): Dateiname eines alternativen Metadatenmappings
- insertInfo (boolean): Ausgabe aller hinzufügbaren Objekttypen
- insertInfo_verbose (boolean): ausführlichere Ausgabe aller hinzufügbaren Objekttypen
- locale (DE, EN, SP, FR, IT, NL, SV, HU, PL): Mittels einer dieser Abkürzungen in Großbuchstaben wird die Ausgabesprache festgelegt.
- lockinfo (boolean): Informationen zum Benutzer sowie Datum und Uhrzeit, an dem das Objekt gesperrt wurde. Alternative Werte sind UNLOCKED, SELF oder EXTERN.
- timestamps (boolean): Bei true werden die Datum-, Zeit- und Datum/Zeit-Basisparameter in Form eines Java-Zeitstempels zurückgegeben.
- refresh (boolean): ignoriert die zwischengespeicherten Daten und ruft die angefragten Daten neu ab
- fillLeadingZeros (boolean): Es werden führende Nullen gemäß der enaio Editoreinstellungen bei Werten mit ausgegeben.
- DEPRECATED: verbose (boolean): Ausgabe aller Felddaten (Bitte documents/search/{Id} verwenden)

Hinweis für insertInfo-Parameter:
Im Normalfall werden nur Pflichtfelder eines Objekttyps ausgegeben. Für weitere Felder muss in der Schemadatei ein insertFields-Tag hinzugefügt werden (siehe Beispiel).

Optionale Felder in insertInfo

```
<cabinet name="Kunde">
      <object name="Dokument">
            <property name="title" field="Typ"/>
            <property name="info" field="Beschreibung"/>
            <insertFields active="true"> <!-- true ist default
-->
                <property field="Beschreibung"/> <!--
Beschreibung ist kein Pflichtfeld -->
            </insertFields>
        </object>
<cabinet/>
```

/osrest/api/documents/parent/[id]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert in einer Baumstruktur bestimmte Elternobjekte zu einem Objekt mit der gegebenen ID. Hat ein Objekt mehrere Standorte wird auch nur einer zurückgeliefert. Um Daten zu einem bestimmten Standort zu erhalten, muss eine parentId mitgegeben werden.
Optionale Parameter sind:

- objecttypeid (Integer): ObjektTypID des gewünschten Objekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.

- parentid (Integer): ID eines bestimmten Elternobjekts
- parenttypeid (Integer): ObjektTypId des Elternobjekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- metadata (String): Dateiname eines alternativen Metadatenmappings

Die Ausgabe entspricht der Form des /osrest/api/documents/parents/[id]-Aufrufs.

/osrest/api/documents/parents/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine sortierte, lineare Liste der Elternobjekte zu einem Objekt mit der gegebenen ID. Neben der linear, sortierten Liste, kann auch die Baumstruktur eines Standorts angezeigt werden, was insbesondere für die Darstellung mehrerer Standorte wichtig ist.
Optionale Parameter sind:

- objecttypeid (Integer): ObjektTypID des gewünschten Objekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- metadata (String): Dateiname eines alternativen Metadatenmappings
- verbose (Boolean): Es wird ein erweitertes generische Metadatenmodell ausgegeben
- tree (Boolean): Wenn `true`, wird die Standort-Hierarchie aller Standorte eines Objekts als Baum dargestellt. Dieser Parameter sollte immer mitgegeben und "true" gesetzt werden, andernfalls kann es bei Objekten mit mehreren Standorten zu Problemen kommen. Die "einfache" lineare Ausgabe wird aus Kompatibilitätsgründen weiterhin verwendet.

Beispiel der Ausgabe (Objekt-ID war hier '14007051'):


```
{
    "pagesize": 500,
    "totalHits": 3,
    "more": false,
    "documents": [
        {
            "id": "4475",
            "type": "FOLDER",
            "fields": {
                "title": "Stoz Pumpenfabrik GmbH",
                "info": "Anwender 0234-OS"
            },
            "fav": false,
            "typeless": false
        },
        {
            "id": "14003866",
            "type": "REGISTER",
            "fields": {
                "title": "Altbestand",
                "info": "Einführung enaio, 50 user, Jukebox"
            },
            "fav": false,
            "typeless": false
```

```
            },
            {
                "id": "14007051",
                "type": "DOCUMENT",
                "fields": {
                    "title": "Angebot: STP-AN-1/13",
                    "info": "Einführung ECM...."
                },
                "fav": false,
                "typeless": false
            }
        ]
    }
```

/osrest/api/documents/objectHierarchy/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die Verzeichnisbaumstruktur eines Ordners oder Registers bis zu einer variablen Tiefe. Der Hashwert "hash" auf der obersten Ebene ist ein kumulativer Hashwert, der den gesamten zurückgegebenen Baum beinhaltet. Er kann optional bei späteren Anfragen als Parameter übergeben werden. Wird er mit übergeben und ist er identisch zu dem aktuellen Hashwert, so wird kein Baum vom Server zurückgeliefert. Er ist ja identisch zum bestehenden. Die Hashwerte auf den einzelnen Ebenen des Baumes beziehen sich ausschließlich auf die Ebene selber. Der jeweilige Hashwert bezieht keine Unterordner/Unterregister mit ein. Über diese Hashwerte kann schnell überprüft werden, ob für die Ebene die gecachten Daten verwendet werden können oder ob die Ebene (falls mehr Informationen notwendig sind) über documents/explore neu bezogen werden muss.
Optionale Parameter sind:

- depth (int): Tiefe der Verzeichnisbaumstruktur
- hash (String): Hashwert, der mit neuem kumulativ errechnetem Hashwert abgeglichen wird. Bei Gleichheit wird kein Baum zurückgegeben.
- fieldsschema (String): ALL|MIN - MIN' (nur Objekt ID), 'ALL' (alle Indexfelder)
- childschema (String): REG|DOC|ALL - REG Alle Register werden automatisch hinzugefügt, DOC Alle Dokumente werden automatisch hinzugefügt, ALL Alle Register und Dokumente werden automatisch hinzugefügt

Beispiel der Ausgabe (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]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert alle Varianten zu dem Objekt mit der gegebenen ID zurück. Hat das Objekt keine Varianten wird das Objekt als Variante zurückgeliefert.
Optionale Parameter:

- objectTypeId (Integer): ObjectTypeId des Dokuments. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- verbose (boolean): Reichert die einzelnen Varianten mit einer 'ecmObject' Property an, die ausführlichere Informationen zu den einzelnen Dokumenten bietet (sowie Indexdaten, etc.).
- refresh (boolean): Übergeht die zwischengespeicherten Daten und ruft die angeforderten Daten neu ab.


```
{
    "id": 14007049,
    "originId": 0,
    "version": "Original",
      "active": false,
    "children": [
        {
            "id": 14007051,
            "originId": 14007049,
            "version": "1.0.0",
                  "active": true,
            "children": [
                {
                    "id": 14041838,
                    "originId": 14007051,
                    "version": "1.1.0",
                              "active": false,
                    "children": []
                }
            ]
        },
        {
            "id": 14119107,
            "originId": 14007051,
            "version": "2.0.0",
                  "active": false,
            "children": []
        }
    ]
}
```


/osrest/api/documents/variants/setactive/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt die Variante mit der übergebenen Varianten ID als aktive Variante.
Path Parameter:

- id (int): ID der Variante, die aktiviert werden soll

Optionale Query Parameter:

- activevariant (Integer): ID der Variante, die gerade aktiv ist. Der Parameter ist optional, aber aus Performanzgründen wird dringend empfohlen den Parameter zu setzen, wenn er bekannt ist.
- objecttypeid (Integer): ObjectTypeId des Dokuments. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.

/osrest/api/documents/variants/create/[originid]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode erstellt eine neue Untervariante zu der Variante mit der übergebenen ID. Ist der Job erfolgreich, wird die neu erstellte Variante zurückgeliefert.
Path Parameter:

- originid (int): ID der Variante aus der das neue Dokument erstellt werden soll

Pflicht Parameter:

- type (String): Gibt an auf welcher Ebene die neue Variante angelegt werden soll (als neue Hauptvariante, Nebenvariante oder Untervariante). Mögliche Werte sind: "main", "parallel", "sub".

Optionale Parameter:

- objecttypeid (Integer): ObjectTypeId des Dokuments. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- verbose (boolean): Reichert die zurückgelieferte Variante mit einer 'ecmObject' Property an, die ausführlichere Informationen zu den einzelnen Dokumenten bietet (sowie Indexdaten, etc.).

/osrest/api/documents/explore/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert den Inhalt des Ordners oder Registers mit der gegebenen ID.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- pagesize (Integer): maximale Anzahl der Treffer
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- insertInfo (Boolean): Ausgabe aller hinzufügbaren Objekttypen (vgl. *Hinweis für insertInfo-Parameter*)

- page (Integer): angeforderte Seite für das Blättern durch die Treffer

/osrest/api/documents/explore

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON
- `Unterstützte Eingabeformate:` JSON

Optionale Parameter sind:

- pagesize (Integer): maximale Treffer pro Seite
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- page (Integer): angeforderte Seite für das Blättern durch die Treffer
- refresh (Boolean): cached Ergebnisse erneuern

Eine Anfrage für einen *explore* Aufruf muss immer die "osid" des zu öffnenden Eltern-Objektes beinhalten. Da auch das Eltern-Objekt ausgegeben wird, kann mit dem *fieldschema_mode* Wert die auszuliefernden Felder jenes Objekts festgelegt werden. *Explore* kennt dabei folgende Parameter, die im JSON auf der oberster Ebene angegeben werden müssen:

Parameter	Typ	Beschreibung	Default
osid	String	ID des Eltern-Objekts ist ein `Pflichtfeld`, muss also gesetzt werden.	-
`fieldschema_mode`	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Felder des angefragten Objektes zurückgegeben werden. MIN= nur Basisparameter | DEF= die im fieldschema-Objekt definierten Felder | ALL= alle Felder	MIN
childschema_mode	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Kind-Objekte zurückgegeben werden. MIN= nur Register und die in *"children"* konfigurierten Objekttypen | DEF= nur die in *"children"* definierten Objekttypen | ALL= alle möglichen Objekttypen	MIN
childfieldsschema_mode	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Felder der Kind-Objekte zurückgegeben werden. MIN= nur Basisparameter der Kinder | DEF= die in *"children"* definierten Felder | ALL= alle Felder der Kinder	MIN
children	List[child]	Definiert die Kindtypen, die ausgegeben werden sollen. Muss in Kombination mit "*childschema_mode": DEF verwendet werden.*	null

Parameter	Typ	Beschreibung	Default
explore_depth	Integer	Legt die Tiefe des *explore* Aufrufs fest. ACHTUNG! kann zu sehr großen Trefferlisten führen schon bei einer Tiefe von 1.	1

Kind-Typen, die in *"children"* definiert werden, kennen folgende Parameter:

Parameter	Typ	Beschreibung
objectTypeId	String	Typ ID des Kind-Typs.
internalName	String	interner Name des Kind-Typs.
displayName	String	sprechender Name des Kind-Typs.
dbname	String	Datenbank Name des Kind-Typs.
osguid	String	OS GUID des Kind-Typs.
fieldsschema	List[field]	Definiert die Felder, die von dem jeweiligen Objekt-Typen zurückgeben werden.

Feld-Typen, die in *"fieldsschema"* definiert werden, kennen folgende Parameter:

Parameter	Typ	Beschreibung
internalName	String	interner Name des Feld-Typs.
displayName	String	sprechender Name des Feld-Typs.
dbname	String	Datenbank Name des Feld-Typs.
osguid	String	OS GUID des Feld-Typs.

Ein *explore* Aufruf kann nach folgendem Beispiel konfiguriert werden:

Beispiel explore Request

```
{
    "osid": "7882",
    "fieldsschema_mode": "MIN",
    "childschema_mode": "DEF",
    "childfieldsschema_mode": "DEF",
    "children": [
      {
        "objectTypeId": "6488100",
        "fields": [
          {
            "internalName": "Patientenstatus"
          },
          {
            "internalName": "File"
          }
        ]
      },
      {
        "objectTypeId": "262208",
        "fields": [
          {
```

```
            "internalName": "feld16"
          },
          {
            "internalName": "feld12"
          }
        ]
      },
      {
        "objectTypeId": "393224",
        "fields": [
          {
            "internalName": "Von_"
          },
          {
            "internalName": "An_"
          },
          {
            "internalName": "Betreff_"
          }
        ]
      }
    ]
  }
```

/osrest/api/documents/insertables/[locationId]

- Unterstützte Anfragemethoden: GET
- Unterstütztes Ausgabeformate: JSON

Die Methode ermittelt die Anzahl der zum Zeitpunkt des Aufrufes gerade einfügbaren Objekte an diesem Standort. Die Ermittlung wird vom Server durchgeführt und basiert auf den Objektrelationen. Diese Objektrelationen werden mithilfe des Editors definiert und legen fest wie viele Objekte eines bestimmten Objekttyps in einem Ordner oder Register angelegt werden dürfen.

Beispiel für die Ausgabe

```
"objectInserts": {
    "262165": -1,
    "393221": -1,
    "6488083": 0,
    "6488084": 3,
    "6488085": -1,
    "6488086": -1
}
```

Die Ausgabe ist eine Liste von Objekttyp-IDs zusammen mit einer Anzahl. Die Anzahl kann -1 für unbegrenzt, 0 für keine und > 0 sein.

/osrest/api/documents/settype/[objectId]

- Unterstützte Anfragemethoden: POST
- Unterstützte Eingabeformate: JSON

Der Request muss den Contenttyp *multipart/form-data* haben (RFC 1867).

Die Methode typisiert ein typenloses Dokument. Ordner und Register können nicht typenlos sein. Da sich typenlose Dokumente entweder in der Benutzer- oder der Workflowablage befinden können, muss diese Information beim Aufruf mitgegeben werden. Typenlose Dokumente in der Akte eines Vorgangsschritts befinden sich damit auch in der Workflowablage. Für die Indexdaten des gewünschten Objekttyps können die Werte zur Verschlagwortung mit übergeben werden.
Parameter sind:

- objectId (Integer): Die ID des zu typisierenden Dokumentes.
- inwftray (boolean): Typisiert das Dokument in der Workflow Ablage.
- inusertray (boolean): Typisiert das Dokument in der Nutzer Ablage.

Es muss einer der Parameter inwftray oder inusertray als true übergeben werden.
Objekt

```
{
    "osid": "23360",
    "objectTypeId": "262176",
    "mainTypeId": "4",
    "fields": {
        "Ordner": {
            "value": "Kunde",
            "internalName": "Ordner"
        },
        "Datum": {
            "value": "1495490400000",
            "internalName": "Datum"
        }
    },
    "result_config": {
        "fieldsschema": []
    }
}
```

/osrest/api/documents/insert/[locationId]

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Ausgabeformate:` JSON
- `Unterstützte Eingabeformate:` JSON

Der Request muss den Contenttyp *multipart/form-data* haben (RFC 1867).
Die Methode erstellt ein Objekt in dem Ordner oder Register mit der gegebenen Standort-ID (Location-ID). Beim Anlegen von Ordnern oder Dokumenten in der Ablage kann der Parameter für die Location ID weggelassen oder eine Null gesendet werden.
Das json-Schema basiert auf dem vom 'insertInfo'-Parameter zurückgegebenem Schema. Zur Identifikation des Typs muss nur dessen ObjectTypeId mitgegeben werden. Die mainTypeId ist optional. Felder werden mit ihrem Anzeigenamen gekennzeichnet. Ihnen kann aber auch der interne Name als Attribut mitgegeben werden, dann wird der Anzeigename (display name) ignoriert.
Auch Varianten können angelegt werden. Dafür wird die ID des Dokuments, von dem die Variante erstellt werden soll, als Location-ID gesetzt.

Für Radiobuttons muss dem Feld das 'type'-Attribut mitgegeben werden ("type": "RADIO").
Varianten: Wenn im Request keine Indexdaten übergeben werden, dann werden die Indexdaten vom Eltern-Dokument übernommen.

Optionaler Path Parameter:

- locationId (int): Objekt ID des gewünschten Standorts (Schrank oder Register ID).

Optionale Parameter sind:

- objecttypeid (Integer): ObjektTypID des gewünschten Standorts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- setvariantactive (boolean): Setzt die neue Variante als aktive Variante, z.B. `http://demo.optimal-systems.org/osrest/api/documents/insert/14133432?setvariantactive=true`
- archivable (boolean): Setzt das Dokument auf archivierbar.
- islink (boolean): Erstellt einen weiteren Standort für das Dokument.
- inwftray (boolean): Erstellt das Dokument in der Workflow Ablage.
- inusertray (boolean): Erstellt das Dokument in der Nutzer Ablage.

Objekt

```
{
  "fields": {
          "Projekttyp": {
              "internalName": "i_Projekttyp",
              "value": "PUA - Allgemein"
          },
      "Art": {
                  "value": "Technik"
          },
      "3": {
              "internalName": "i_sichtbarfuer",
              "value": "DEMO(g)"
          }
  }
}
```

Tabellen werden wie folgt befüllt. Bitte beachten, zwar nur die Spalten dem Request mitgegeben werden muss, die auch editiert werden, aber es müssen alle Zeilen mitgeliefert werden, da enaio-Seitig eine Ersetzung der Tabellenwerte erfolgt.

```
{
     "fields": {
      "ERH_INV_POS": {
            "columns": [
                  {
                        "internalName": "ERH_INV_POS_QSID"
                  },
                  {
                        "internalName": "ERH_INV_POS_P1"
                  },
                  {
                        "internalName": "ERH_INV_POS_P3"
                  }
```

```
                ],
                "rows": [
                      ["Row 1 - 1", "Row 1 - 2", "Row 1 - 3"],
                      ["Row 2 - 1", "Row 2 - 2", "Row 2 - 3"],
                      ["Row 3 - 1", "Row 3 - 2", "Row 3 - 3"],
                      ["Row 4 - 1", "Row 4 - 2", "Row 4 - 3"]
                ]
          }
        }
}
```

/osrest/api/documents/update/[id]

- Unterstützte Anfragemethoden: POST
- Unterstütztes Ausgabeformate: JSON
- Unterstützte Eingabeformate: JSON

Die Methode ändert das Objekt mit der gegebenen ID. Verwendung ähnelt der der documents/insert-Methode, wobei die ObjektTypeId optional mitgegeben werden kann, um den Aufruf zu beschleunigen. Außerdem können nur explizit angegebene Felder in enaio® geändert werden. Der Aufruf muss den Contenttyp *multipart/form-data* haben (RFC 1867).
Optionale Parameter sind:

- replacefiles (Boolean): Ersetzt die Datei eines Dokuments, wenn Parameter 'true'.

/osrest/api/documents/archive/[id]

- Unterstützte Anfragemethoden: GET
- Unterstütztes Ausgabeformate: JSON

Die Methode ändert den Archivierstatus des Objekts mit der gegebenen ID. Die ObjektType Id kann optional mitgegeben werden, um den Aufruf zu beschleunigen.
Parameter sind:

- id (int): Objekt ID des DMS Dokuments
- archive (boolean): Gibt an, ob das Dokument archiviert werden soll oder ob die Archivierung aufgehoben werden soll.

Optionale Parameter sind:

- objecttypeid (Boolean): ObjektTyp ID des gesuchten Objekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.

/osrest/api/documents/search

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Ausgabeformate:` JSON
- `Unterstützte Eingabeformate:` JSON

Die Methode ermöglicht die Suche nach einem Objekt anhand von Feldwerten und/oder Basisparametern. Die Suche kann eingeschränkt werden (kombinierte Suche), indem Sie weitere Objekttypen angeben. Objekttypen werden mithilfe des internen Namens, dem Schranknamen, der OSID oder der Objekttyp-ID angegeben. Felder werden als Feld-Objekte in "fields" angegeben. Ihnen kann das Attribut "internalName" mitgegeben werden, ansonsten wird der Objektbezeichner als einfacher Name angenommen.
Optionale Parameter sind:

- pagesize (Integer): Größe der Ergebnismenge. Damit kann ein Paging erreicht werden.
- offset (Integer): Detaillierter Offset unabhängig von der pagesize.
- page (Integer): Bei Paging die anzuzeigende Seite. Überschreibt den Parameter offset. Offset berechnet sich hierbei aus page * pagesize.
- maxhits (Integer): Maximal zu ladende Treffer. Das können mehr Treffer sein, als bei pagesize angegeben worden sind. Damit kann das Caching reguliert werden, sodass Nachfolgeseiten schneller ausgeliefert werden.
- rights (Boolean): In der Trefferliste sollen für jedes Objekt auch die Objektrechte ausgegeben werden.
- fillLeadingZeros (boolean): Es werden führende Nullen gemäß der enaio Editoreinstellungen bei Werten mit ausgegeben.
- fieldsschema (string): Legt fest, in welchem Modus die Felder des Objektes zurückgegeben werden. Mögliche Werte: MIN, ALL und DEF

Achtung

Die Trefferliste beinhaltet standardmäßig alle Index- und Metadaten und sorgt somit für große Datenübertragungen.

Hinweis

Statt des internen Namens können auch einfache Namen verwendet werden. Dann muss jedoch auch der Schrankname angegeben werden, da der einfache Name nicht systemweit eindeutig ist.

```
{
  "query": {
    "objectTypeId": "262220",
    "fields": {
      "Vertragsbasis": {
        "value": "1"
      },
      "Archivbeleg erzeugen": {
        "value": "1"
      }
    },
      "baseparams": {
```

```
            "Creator": {
                    "value": "DEMO"
                },
                "Created": {
                    "value": "1028114820000"
                },
                "Modifier": {
                    "value": "DEMO",
                    "negate": "true"
                }
            }
        },
        "additionalQueries": [
          {
            "objectTypeId": "54",
            "fields": {
              "Projekttyp": {
                "value": "PUA - Allgemein",
                "internalName": "i_Projekttyp"
              },
              "ProjektNr.": {
                "value": "123",
                "internalName": "i_ProjektNr."
              },
              "Projektbezeichnung": {
                "value": "Testwert",
                "internalName": "i_Projektbezeichnung"
              },
              "sichtbarfür": {
                "value": "DEMO(g)",
                "internalName": "i_sichtbarfuer"
              }
            },
              "baseparams": {
                  "Creator": {
                    "value": "DEMO"
                  },
                  "Created": {
                    "value": "1028114820000"
                  },
                  "Modifier": {
                    "value": "DEMO",
                    "negate": "true"
                  }
              }
            }
        ]
    }
```


Standardmäßig werden alle Bedingungen mittels UND verknüpft. Mit den Versionen enaio 8.00.0, 8.10.2 sowie 8.50.0, welche ab Anfang September 2017 bereitgestellt werden, können diese, wie nachfolgend im Beispiel dargestellt wird, auch mittels ODER verknüpft werden. Die einzelnen ConditionGroups werden miteinander ODER verknüpft. Die Felder innerhalb einer ConditionGroup UND verknüpft. Konkret auf das hier drunter folgende Beispiel bedeutet die Einschränkung: ((feld17 == "MEDICOLOR" && feld1 == "Diagnostik") || (feld1 == "Notfall")).


```
{
```

```
    "query":{
        "objectTypeId":"262208",
        "conditionGroups": [{
            "fields": {
                "feld17":{
                    "value": "MEDICOLOR"
                },
                        "feld1":{
                    "value": "Diagnostik"
                },
            }
        },{
            "fields": {
                "feld1":{
                    "value": "Notfall"
                }
            }
        }]
    }
}
```

Ebenfalls mit den Versionen enaio 8.00.0, 8.10.2 sowie 8.50.0, welche ab Anfang September 2017 bereitgestellt werden, können die normalen Bedingungsfelder sowie die Felder in den ConditionGroups negiert und gegen Null eingeschränkt werden. Somit ist eine Suche nach allen Werten außer dem angegebenen Wert möglich, was in Kombination mittels UND und ODER sowie den Operatoren für arithmetische Vergleiche (siehe weiter unten), alle Bedingungen ermöglichen sollte.

```
{
    "query":{
        "objectTypeId":"262208",
            "fields": {
                  "feld12":{
                  "null": true,
                        "negate": true
                  }
            }
            "conditionGroups": [{
            "fields": {
                "feld17":{
                    "value": "MEDICOLOR",
                             "negate": true
                        }
            }
        },{
            "fields": {
                "feld1":{
                    "value": "Notfall"
                }
            }
        }]
    }
}
```

Die Ergebnisliste kann mit dem "result_config" Parameter konfiguriert werden. Folgende Werte sind möglich:

Parameter	Typ	Beschreibung	Default
pagesize	Integer	Anzahl von Treffern - siehe gleichnamiger URI-Parameter. Dieser Parameter wird gegenüber dem aus der URI priorisiert.	null
offset	Integer	Paging offset - siehe gleichnamiger URI-Parameter. Dieser Parameter wird gegenüber dem aus der URI priorisiert.	0
maxhits	Integer	maximale Anzahl von Treffern - siehe gleichnamiger URI-Parameter. Dieser Parameter wird gegenüber dem aus der URI priorisiert.	500
deny_empty	Boolean	Wenn true werden Feldwerte die null sind in der Ergebnisliste NICHT angezeigt.	false
normalize_values	Boolean	Wenn true werden Date, Time und DateTime Felder in timestamps (mit Millisekunden) umgewandelt, sonst wird das jeweilige Server-Format ausgegeben.	false
fieldsschema	Object	Legt die Reihenfogle der Treffer und welche Felder zurückgegeben werden sollen fest.	null
distinct	Boolean	Das Ergebnis beinhaltet nur das erste fieldschema-Objekt und von diesem auch nur alle unterschiedlichen Werte. Die unterschiedlichen Werte werden zudem eingegrenzt unter einbeziehung der angegeben fields als condition.	false
fieldsschema_mode	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Felder des angefragten Objektes zurückgegeben werden. MIN= nur Basisparameter | DEF= die im fieldschema-Objekt definierten Felder | ALL= alle Felder	ALL
childschema_mode	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Kindobjekte des angefragt Objektes zurückgegeben werden. MIN= nur Register und die in "children"-Objekt konfigurierte Objekttypen | DEF= nur die im "children"-Objekt definierten Objekttypen | ALL= alle Kinder	MIN

Parameter	Typ	Beschreibung	Default
childfieldsschema_mode	\[MIN, DEF, ALL\]	Legt fest in welchem Modus die Felder der Kindobjekte zurückgegeben werden. MIN= nur Basisparameter der Kinder | DEF= die im children-Objekt definierten Felder eines Objekttypen | ALL= alle Felder der Kinder	MIN
export_depth	Integer	Legt die Anzahl von anzufragenden Ebenen unterhalb des Hauptobjektes fest. ACHTUNG! Ergebnismenge kann sehr groß werden.	0

Fieldschema:

Parameter	Typ	Beschreibung	Default
internalName	String	Identifier des Feldes	null
displayName	String	Identifier des Feldes	null
objectTypeId	String	Identifier des Feldes	null
dbName	String	Identifier des Feldes	null
sort_order	\[ASC, DESC\]	Reihenfolge aufsteigend oder absteigend	ASC
sort_pos	Integer	Position in der Sortierreihenfolge (größer 0)	null

Beispiel Ergebniskonfiguration

```
{
  "query": {
    "cabinet": "Kunde",
    "name": "Kunde",
    "result_config": {
      "fieldsschema": [
        {
          "dbName": "id",
          "sort_pos": 1,
          "sort_order": "DESC"
        }
      ],
      "pagesize": 100,
      "offset": 100,
      "maxhits": 500,
      "deny_empty": true,
      "normalize_values": false
    }
  }
}
```

Die Basisparameter können zudem negiert werden, indem der Schlüssel *negate* auf true gesetzt wird. Die Basisparameter finden sie etwas weiter unten in der dargestellten Tabelle. Bei allen numerischen, Datum-, Zeit- und Datum/Zeit-Werten kann nach Bereichen angefragt werden. Die nachfolgende Tabelle listet alle Operatoren auf:

Operator	Erklärung

>Value	Alle Werte, die größer als Value sind.
>=Value	Alle Werte, die größer/gleich Value sind.
!=Value	Alle Werte, die ungleich Value sind.
<Value	Alle Werte, die kleiner als Value sind.
<=Value	Alle Werte, die kleiner/gleich Value sind.
Value1-Value2	Alle Werte, die im Bereich von Value1 bis Value2 liegen.

Zusätzlich kann auch eine Volltextsuche integriert werden. Dafür einfach für jedes beliebige Objekt das Attribut "vtx_query" befüllen.

Kombinierte Suche und Volltextsuche

```
{
  "query": {
    "objectTypeId": "3",
    "vtx_query": "Olaf",
    "fields": {
      "Branche": {
        "value": "Industrie"
      },
      "Status": {
        "value": "2"
      }
    }
  },
  "additionalQueries": [
    {
      "objectTypeId": "131079",
      "vtx_query": "Björn"
    }
  ]
}
```

Hierarchische Anfragen:
Um Unterobjekte eines Objektes anzufragen, muss der "export_depth" Wert im "result_config" Objekt gesetzt werden. Damit die Ergebnismenge nicht zu groß wird, sollten die angefragten Felder der Kinder unbedingt beschränkt werden. Mit "childschema_mode" und "childfieldsschema_mode" werden die angefragten Kindobjekte und deren Felder konfigurierbar gemacht. Welche konkreten Objekttypen und Felder angefragt werden lässt sich im "children" Objekt spezifizieren, wie in folgendem Beispiel zu sehen ist:

Beispiel Hierarchische Anfrage

```
{
  "query": {
    "objectTypeId": "6488101",
    "osid": "7882",
    "cabinet": "Patient_KFN",
    "name": "Register2",
    "result_config": {
      "export_depth": 1,
      "fieldsschema_mode": "MIN",
      "childschema_mode": "DEF",
      "childfieldsschema_mode": "DEF"
    },
    "children": [
      {
```

```
        "objectTypeId": "6488100",
        "fields": {
          "1": {
            "internalName": "Patientenstatus"
          },
          "2": {
            "internalName": "File"
          }
        }
      },
      {
        "objectTypeId": "262208",
        "fields": {
          "1": {
            "internalName": "feld16"
          },
          "2": {
            "internalName": "feld12"
          }
        }
      },
      {
        "objectTypeId": "393224",
        "fields": {
          "1": {
            "internalName": "Von_"
          },
          "2": {
            "internalName": "An_"
          },
          "3": {
            "internalName": "Betreff_"
          }
        }
      }
    ]
  }
}
```

Von den Basisparametern des Server-API-Handbuchs werden aktuell folgende unterstützt. Nicht gesetzte Schlüssel sind vom Wert her egal:

Basisparameter	Mögliche Werte	Erklärung
`Creator`	Benutzername[1]	Eine Liste mit Benutzernamen von ECM-Benutzern. Die Benutzernamen werden mittels ODER angefragt. Bei einer Negation hingegen werden die Benutzernamen mittels UND verknüpft. Der Creator ist jener Benutzer, der das ECM-Objekt angelegt hat.
`Created`	Java-Timestamp[1]	Eine Zeichenkette, die ein Java-Timestamp enthält. Es können auch die Operatoren <, <=, >, >= verwendet werden. Mittels eines - zwischen zwei Java-Timestamps kann ein von-bis Bereich definiert werden. Der Basisparameter enthält den Anlegezeitpunkt des ECM-Objektes.

Basisparameter	Mögliche Werte	Erklärung
`Modifier`	Benutzername[1]	Eine Liste mit Benutzernamen von ECM-Benutzern. Die Benutzernamen werden mittels ODER angefragt. Bei einer Negation hingegen werden die Benutzernamen mittels UND verknüpft. Der Modifier ist jener Benutzer, der das ECM-Objekt zuletzte geändert hat.
`Modified`	Java-Timestamp[1]	Eine Zeichenkette, die ein Java-Timestamp enthält. Es können auch die Operatoren <, <=, >, >= verwendet werden. Mittels eines - zwischen zwei Java-Timestamps kann ein von-bis Bereich definiert werden. Der Basisparameter enthält den Zeitpunkt der letzten Änderung.
`Owner`	Benutzername[1]	Eine Liste mit Benutzernamen von ECM-Benutzern. Die Benutzernamen werden mittels ODER angefragt. Bei einer Negation hingegen werden die Benutzernamen mittels UND verknüpft. Der Owner ist jener Benutzer, dem das ECM-Objekt gehört.
`ArchiveState`	`ARCHIVED|ARCHIVABLE|NOT_ARCHIVABLE|NO_PAGES|REFERENCE`	Eine Liste dieser Werte. Sie werden mittels ODER verbunden. ARCHIVED: Das ECM-Objekt ist archiviert ARCHIVABLE: Das ECM-Objekt ist zur Archivierung freigegeben NOT_ARCHIVABLE: Das ECM-Objekt ist nicht zur Archivierung freigegben NO_PAGES: Das ECM-Objekt hat lediglich Indexdaten und kein Contentdokument REFERENCE: Das ECM-Objekt ist eine Referenz auf ein anderes ECM-Objekt.
`Archivist`	Benutzername[1]	Eine Liste mit Benutzernamen von ECM-Benutzern. Die Benutzernamen werden mittels ODER angefragt. Bei einer Negation hingegen werden die Benutzernamen mittels UND verknüpft. Der Archivist ist jener Benutzer, der das ECM-Objekt archiviert hat.
`ArchiveDate`	Java-Timestamp[1]	Eine Zeichenkette, die ein Java-Timestamp enthält. Es können auch die Operatoren <, <=, >, >= verwendet werden. Mittels eines - zwischen zwei Java-Timestamps kann ein von-bis Bereich definiert werden. Der Basisparameter enthält den Zeitpunkt der Archivierung.

Basisparameter	Mögliche Werte	Erklärung
`Locked`	Benutzername[1]	Eine Liste mit Benutzernamen von ECM-Benutzern. Die Benutzernamen werden mittels ODER angefragt. Ein Listeneintrag kann auch einen Leerstring für UNLOCKED oder * für LOCKED (von irgendwem) enthalten. Bei einer Negation hingegen werden die Benutzernamen mittels UND verknüpft. Ein Locked-Benutzer ist jener, welcher ein ECM-Objekt aktuell ausgecheckt hat.
`HasVariants`	true	Nur ECM-Objekte, die Varianten haben. Eine Negation wird hier nicht unterstützt.
`Version`	Integer[1]	Die Versionsnummer eines ECM-Objektes.
`RetentionDate`	Java-Timestamp[1]	Eine Zeichenkette, die ein Java-Timestamp enthält. Es können auch die Operatoren <, <=, >, >= verwendet werden. Mittels eines - zwischen zwei Java-Timestamps kann ein von-bis Bereich definiert werden. Der Basisparameter enthält den Zeitpunkt, wann ein ECM-Objekt aus dem Archiv planmäßig gelöscht wird.
`RetentionPlannedDate`	Java-Timestamp[1]	Eine Zeichenkette, die ein Java-Timestamp enthält. Es können auch die Operatoren <, <=, >, >= verwendet werden. Mittels eines - zwischen zwei Java-Timestamps kann ein von-bis Bereich definiert werden. Der Basisparameter enthält den Zeitpunkt, zu wann ein ECM-Objekt aus dem Archiv planmäßig gelöscht werden soll.
`Links`	Integer[1]	Ein ganzzahliger Wert, der die Anzahl der Links auf dieses ECM-Objekt enthält.
`Medium`	Integer[1]	Die Id des Mediums, auf welchen die gewünschten ECM-Objekte archiviert sein müssen.
`SystemID`	Integer[1]	Die Id des Fremdarchives, auf welchen die gewünschten ECM-Objekte archiviert sein müssen.
`ForeignID`	Integer[1]	Die Fremdarchiv-ID eines ECM-Objektes.
`DocumentPageCount`	Integer[1]	Die Anzahl der Contentdokumente, die ein ECM-Objekt enthält.
`Register`	true | false	Wenn sich das ECM-Objekt in einem Register befinden soll kann dieser Wert mit true angefragt werden. Wenn es sich direkt in einem Ornder befinden soll mit false.

Basisparameter	Mögliche Werte	Erklärung
`MultiLocation`	true	Nur ECM-Objekte, die im ECM mehrere Standorte besitzen. Eine Negation ist hier nicht möglich zur Ermittlung von ECM-Objekten, die ausschließlich einen Standort besitzen. `Verfügbar ab enaio 8.50 mit oxjobdms.dll 8.50.17036.748`.
`Signed`	`CURRENT|PREVIOUS`	Eine Liste der folgenden Konstanten kann hier übergeben werden. Verfügbar ab enaio 8.50 mit oxjobdms.dll 8.50.17036.748. CURRENT: Nur ECM-Objekte, die in der aktuellen Version signiert sind. PREVIOUS: Nur ECM-Objekte, die in einer ihrer älteren Versionen signiert sind.
`LoanedOut`	true	Nur ausgelagerte ECM-Objekte.

[1] Wert kann über einen gesetzten "negate"-Schlüssel negiert werden.

Die Rechteinformationen, welche über den GET-Parameter rights=true mit angefordert werden können, sagen folgendes aus:

Rechteschlüssel	Beschreibung
`objModify`	Der Benutzer darf Änderungen an den Dokumentdateien vornehmen.
`objDelete`	Der Benutzer darf das Objekt löschen.
`objExport`	Der Benutzer darf das Objekt öffnen bzw. exportieren.
`indexModify`	Der Benutzer darf die Indexdaten des Objekts ändern.

/osrest/api/documents/search/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode ermöglicht die Suche nach einem Objekt mit angegebener OSID. Die Ausgabe, mit allen Index- und Metadaten, ist dabei genauso aufgebaut, wie die der POST Such-Methode.
Optionale Parameter sind:

- objecttypeid (Integer): ObjektTyp ID des gesuchten Objekts. Der Parameter ist optional, aber aus Performanzgründen wird empfohlen den Parameter zu setzen, wenn er bekannt ist.
- refresh (boolean): Der Cache wird geleerte und das Ergebnis neu vom Server angefragt.
- locale (DE, EN, SP, FR, IT, NL, SV, HU, PL): Mittels einer dieser Abkürzungen in Großbuchstaben wird die Ausgabesprache festgelegt.
- lockinfo (boolean): Informationen zum Benutzer sowie Datum und Uhrzeit, an dem das Objekt gesperrt wurde. Alternative Werte sind UNLOCKED, SELF oder EXTERN.
- timestamps (boolean): Bei true werden die Datum-, Zeit- und Datum/Zeit-Basisparameter in Form eines Java-Zeitstempels zurückgegeben.
- fillLeadingZeros (boolean): Es werden führende Nullen gemäß der enaio Editoreinstellungen bei Werten mit ausgegeben.

/osrest/api/documents/vtx?query=[Suchbegriff]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert das Ergebnis einer Volltextsuche mit dem gegebenen Suchbegriffen.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- maxHits (Integer): maximale Anzahl der Treffer
- objects (Integer): Comma-seperated list of object oids on which the search should be executed. If empty, all objects are searched
- verbose (boolean): Reichert das Ergebnis um 'ecmObject' an, welche ausführlichere Informationen zu den einzelnen Dokumenten bietet (sowie Indexdaten, etc.)

```
.../osrest/api/documents/vtx?query=Meine%20Suchbegriffe
```

/osrest/api/documents/vtx

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON
- `Unterstützte Eingabeformate:` JSON

Die Methode liefert das Ergebnis einer Volltextsuche mit dem gegebenen Suchbegriff. Wird im Anfrage-JSON der schlüssel "verbose" mitgeliefert und ist dieser true so werden 'ecmObject', die ausführlichere Informationen zu den einzelnen Dokumenten bietet (sowie Indexdaten, etc.), im Ergebnis zurückgeliefert.
Pflichtparameter sind:

- query

Optionale Parameter sind:

- facets (Map<String, List<String>>): pro Facetten-Typ sind mehrere Werte zulässig
- maxHits (Integer): maximale Anzahl von Treffern
- objectIds (Integer[]): Liste von Objekt-IDs
- objectTypeIds (Integer[]): Liste von Objekttyp-IDs

Eingabe JSON

```
{
  "query": "demo*",
  "maxHits": 50,
  "verbose": "true"
  "facets": {
    "anleger": [
      "demo"
        ],
    "objtype": [
      "262144",
      "131114",
```

```
      "393225"
        ]
    }
}
```

Ausgabe JSON


```
{
    "facets": {
        "anleger": [
            {
                "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": "Unterlagen",
                    "info": "PM Neue Version enaio(Entwurf SJ)"
                },
                "fav": false,
                "typeless": false,
                "pages": 1,
                "lastmodified": "1393589826000",
                "preview": "Unterlagen  20050321 PM Neue
Version enaio (Entwurf SJ)  <em>DEMO</em> <em>DEMO</em>
20140228 1.0.0 <em>DEMO</em>  <em>DEMO</em>"
            }, {...}
        ]
    }
}
```


/osrest/api/documents/vtx/autocomplete/[term]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON
- Deprecated seit 8.50 Build 1.5.45. Bitte nutzen Sie den nachfolgenden POST-Aufruf.

Die Methode liefert bis zu 5 Vorschläge zur Vervollständigung des übergebenen Terms.

/osrest/api/documents/vtx/autocomplete

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Die Methode liefert bis zu 5 Vorschläge zur Vervollständigung des übergebenen Terms.

Eingabe JSON

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

/osrest/api/documents/baseparams/[id]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert die enaio®-Eigenschaften zur übergebenen Objekt-ID zurück.

/osrest/api/documents/history/[id]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert die Objekthistorie zur übergebenen Objekt-ID zurück.

Beispiel der Ausgabe

```
[
    {
        "time": 1392211328000,
        "action": 7,
        "actionname": "Dokument ausgegeben",
        "description": "Das Dokument wurde vom Anwender
gelesen, gedruckt oder anderweitig ausgegeben. Eine Änderung
erfolgte nicht.",
        "username": "ROOT",
        "userdisplayname": "",
        "info": "Das Dokument wurde zur Ansicht geöffnet."
    },
    {
        "time": 1392211327000,
        "action": 31,
        "actionname": "Objektinfo",
```

```
        "description": "Protokolleintrag aus dem
Geschäftsmodell.",
        "username": "DEMO",
        "userdisplayname": "Klaus Mustermann",
        "info": "Dokument wurde zur Vorschau angezeigt."
    },
    {
        "time": 1203583524000,
        "action": 7,
        "actionname": "Dokument ausgegeben",
        "description": "Das Dokument wurde vom Anwender
gelesen, gedruckt oder anderweitig ausgegeben. Eine Änderung
erfolgte nicht.",
        "username": "VERTRAG",
        "userdisplayname": "",
        "info": "Dokument zur Bearbeitung abgerufen"
    },
    {
        "time": 1203583489000,
        "action": 7,
        "actionname": "Dokument ausgegeben",
        "description": "Das Dokument wurde vom Anwender
gelesen, gedruckt oder anderweitig ausgegeben. Eine Änderung
erfolgte nicht.",
        "username": "VERTRAG",
        "userdisplayname": "",
        "info": "Dokument zur Ansicht abgerufen"
    }
]
```


/osrest/api/documents/notes/[id]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Parameter:

- id (int): Objekt-ID

Die Methode liefert Notizen zur übergebenen Objekt-ID zurück.

Beispiel der Ausgabe


```
[
    {
        "id": 14121121,
        "creator": {
                        "id": 22,
                        "name": DEMO",
                        ... }
           "modifier": {
                        "id": 77,
                        "name": MEIER",
                        ... }
        "color": "2",
        "colorName": "YELLOW"
        "text": "Das ist ein ideales Testdokument für
Notizen.",
```

```
            "date": 1465909598000,
                "creationDate": 1392290415000
        },
        ...
    ]
```

/osrest/api/documents/notes/[id]

- Unterstützte Anfragemethoden: POST
- Unterstützte Eingabeformate: JSON
- Unterstützte Ergebnisformate: JSON

Parameter:

- id (int): Objekt-ID

Die Methode legt eine neue Notiz zur übergebenen Objekt-ID an. Als Ergebnis wird die komplette Liste aller Notizen (wie bei GET) zurückgegeben.
Beispiel der Input Daten

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

/osrest/api/documents/notes/remove/[id]/[noteId]

- Unterstützte Anfragemethoden: POST, DELETE
- Unterstützte Ergebnisformate: JSON

Parameter:

- id (int): Objekt-ID
- noteId (int): Notiz-ID

Die Methode löscht eine Notiz zur übergebenen Objekt-ID und Notiz-ID. Als Ergebnis wird die komplette Liste aller Notizen (wie bei GET) zurückgegeben.

/osrest/api/documents/notes/update/[id]

- Unterstützte Anfragemethoden: POST
- Unterstützte Eingabeformate: JSON
- Unterstützte Ergebnisformate: JSON

Die Methode übernimmt Änderungen an der übergebenen Notiz zum Objekt mit der übergebenen Objekt-ID. Als Ergebnis wird die komplette Liste aller Notizen (wie bei GET) zurückgegeben.
Parameter:

- id (int): Objekt-ID

Beispiel der Input Daten

```
{
            "id": "17",
        "color": "3",
        "text": "Notiz via REST-API"
}
```

/osrest/api/documents/portfolios

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die für den Benutzer sichtbaren Mappen zurück.
Optionale Parameter sind:

- creator (String): Schränkt die Suche auf einen bestimmten Mappenanleger ein.
- recipient (String): Schränkt die Suche auf einen bestimmten Mappenempfänger ein.
- subject (String): Schränkt die Suche auf einen bestimmten Mappenbetreff ein.
- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird das Cache ignoriert und die Anfragen neu vom Server abgefragt.

/osrest/api/documents/portfolio/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert den Inhalt einer Mappe zurück.
Optionale Parameter sind:

- pagesize (Integer): maximale Anzahl der Treffer
- offset (Integer): Trefferoffset für das Blättern durch die Treffer
- page (Integer): angeforderte Seite für das Blättern durch die Treffer
- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird das Cache ignoriert und die Anfragen neu vom Server abgefragt.
- metadata (String): Dateiname eines alternativen Metadatenmappings

/osrest/api/documents/portfolio/{id}/add/{docid}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode fügt ein Dokument einer Mappe hinzu.

/osrest/api/documents/portfolio/{id}/remove/{docid}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode entfernt ein Dokument aus einer Mappe.

/osrest/api/documents/favourites

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller Objekte, die sich in der 'Favoriten'-Mappe befinden.
Optionale Parameter sind:

- metadata (String): Dateiname eines alternativen Metadatenmappings
- verbose (Boolean): Es wird ein erweitertes generische Metadatenmodell ausgegeben
- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird das Cache ignoriert und die Anfragen neu vom Server abgefragt.

/osrest/api/documents/favourites/count

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die Anzahl aller Objekte, die sich in der 'Favoriten'-Mappe befinden.
Optionale Parameter sind:

- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird das Cache ignoriert und die Anfragen neu vom Server abgefragt.

/osrest/api/documents/favourites/add

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode fügt ein oder mehrere Objekte mit den gegebenen IDs der 'Favoriten'-Mappe hinzu.
Beispiel der Input Daten

```
[
	{
		"id": "25465"
	},
	{
		"id": "23365"
	},
	...
]
```

/osrest/api/documents/favourites/add/[id]

- `Unterstützte Anfragemethoden:` POST, GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode fügt das Objekt mit der gegeben ID der 'Favoriten'-Mappe hinzu.

/osrest/api/documents/favourites/delete

- Unterstützte Anfragemethoden: POST, GET
- Unterstützte Ergebnisformate: JSON

Die Methode entfernt ein oder mehrere Objekte mit den gegebenen IDs aus der 'Favoriten'-Mappe.

Beispiel der Input Daten

```
[
	{
		"id": "25465"
	},
	{
		"id": "23365"
	},
	...
]
```

/osrest/api/documents/favourites/delete/[id]

- Unterstützte Anfragemethoden: POST, GET
- Unterstützte Ergebnisformate: JSON

Die Methode entfernt das Objekt mit der gegeben ID aus der 'Favoriten'-Mappe.

/osrest/api/documents/raw/[id]

- Deprecated! Bitte /osrest/api/documents/search/{id} fortan verwenden!
- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert Objektinformationen im JSON-Format zurück. Optionale Parameter sind:

- locale (String): Über die Angabe des ISO Language Code (de, en etc.) kann die Sprache der Ausgabe bestimmt werden.

/osrest/api/documents/annotations/{objectId}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode gibt alle Anmerkungen im Dokument eines ECM-Objekts zurück.

/osrest/api/documents/annotations

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON
- Unterstützte Eingabeformate: JSON

Die Methode fügt eine Anmerkung zum Dokument des ECM-Objekts hinzu oder aktualisiert eine bestehende. Die Seite, auf der sich die Anmerkung befindet, ist 0-basiert. Die erste Seite ist somit 0.
Die pageXStart, pageYStart sowie deren End-Werte sind prozentuale Werte in Promille (also mal 10). Der Typ ANNOT ist aktuell der einzig unterstützte und steht für Textanmerkungen. Weitere Anmerkungstypen werden bald folgen.
Die backgroundColor ist RGBA. Data und Text sind aktuell identisch. In späterer Ausbaustufe kann data auch Binäre Base64-Daten enthalten und Text muss Text sein, der durchsuchbar ist.
Wenn Sie eine Anmerkung hinzufügen möchten, übergeben Sie eine leere id, wie im Codebeispiel hier drunter ersichtlich. Bei einem Update muss als id die id der Anmerkung übergeben werden.

Eingabe JSON

```
{
      "id": "",
      "objectId": 123,
      "page": 0,
      "pageXStart": 780.0,
      "pageYStart": 542.1,
      "pageXEnd": 0,
      "pageYEnd": 0,
      "type": "ANNOT",
      "backgroundColor": "FF000000",
      "data": "Ich bin eine Textannotation",
      "text": "Ich bin eine Textannotation"
 }
```

/osrest/api/documents/annotations/delete/{objectId}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode entfernt eine oder alle Anmerkungen im Dokument eines ECM-Objekts.
Pflicht-Parameter sind:

- id (Integer): Die ID der Anmerkung, welche aus dem Dokument entfernt werden soll. Wird keine ID angegeben, so werden alle Anmerkungen im Dokument gelöscht.

/osrest/api/documents/annotations/pdf/{objectId}

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Diese Methode gibt die PDF-Version des Dokuments vom ECM-Objekt inkl. Anmerkungen zurück. Die Anmerkungen werden in das PDF eingebettet.

/osrest/api/documents/templates

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Vorlagen zu allen ObjektTypen zurück.
Beispiel Ergebnis

```
[
  {
    "alias": "Word",
    "editor": "WINWORD.EXE",
    "fileName": "BriefVorlage.docx",
    "extension": "docx",
    "nameSpace": "Word",
    "objectType": "262144",
    "id": "1939"
  }, ...
]
```

/osrest/api/documents/templates/{objectTypeId}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Vorlagen zu dem angegebenen ObjektTypen zurück.

/osrest/api/documents/templates/acquisition?id={id}&objectTypeId={objectTypeId}&templateId={templateId}&fillTemplate={fillTemplate}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode füllt das Template, welches über den Queryparameter templateId spezifiziert wird, mit den Indexdaten aus dem ECM-Objekt, welches mit dem Queryparameter id und objectTypeId spezifiziert wird.
Bei Erfolg wird die mit Content angereicherte Vorlage dem spezifizierten Objekt als Content zugewiesen.
Pflicht-Query-Parameter sind:

- id (Integer): Objekt-Id, welches einerseits die in die Vorlage zu übernehmenden Indexdaten beinhaltet und andererseits das fertig ausgefüllte Dokument als Content erhält.
- objectTypeId (Integer): Object-Type-Id des Objektes, welches einerseits die in die Vorlage zu übernehmende Indexdaten beinhaltet und andererseits das fertig ausgefüllte Dokument als Content erhält.
- templateId (Integer): Id der Vorlage
- fillTemplate (Boolean): Sollen die Daten des ECM-Objektes in die Vorlage übernommen werden (true) oder die Vorlage im Rohformat ohne übernommenen Inhalt als Content dem ECM-Objekt hinzugefügt werden (false).

/osrest/api/documents/scripts

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON
- Ab enaio 8.50

Diese Methode liefert alle DMS Skripte zurück. Das Ergebnis kann über die Suchparameter eingeschränkt werden. Gibt man z.B. den ClientType "web" und die ObjectTypeId "10254" als Parameter mit, bekommt man alle Skripte zurück geliefert, die Web-Skripte sind und zum ObjektTypen "10254" gehören. Gleiches gilt in vergleichbarer Weise für die Parameter clienttype und eventtype. Sind Parameter nicht angegeben, so werden alle DMS Skripte ohne die jeweilige Einschränkung zurückgegeben.
Optionale Parameter:

- clienttype (String): Liefert Skripte für den bestimmten Client Application Type zurück. Mögliche Werte sind: desktop, web
- objecttypeid (String): Liefert Skripte zu der spezifischen ObjectTypeId zurück.
- eventtype (String): Liefert Skripte zu dem spezifischen EventType zurück.
- refresh (boolean): Ignoriert zwischengespeicherte Daten und ruft die gewünschten Daten neu ab.

Folgende Werte können als eventtype angegeben werden. Für weitergehende Informationen schauen Sie bitte im Handbuch OS_Eventeditor_de.pdf.

clienttype desktop (Visual Basic Skripte)	clienttype web (Javascript)
SHOW	SHOW
BEFORE_VALIDATE	BEFORE_VALIDATE
AFTER_SAVE	AFTER_SAVE
BEFORE_START_QUERY	
AFTER_FINISH_QUERY	
BEFORE_START_REQUEST	
AFTER_FINISH_REQUEST	
AFTER_LOGIN	
BEFORE_LOGOUT	
TIME	
BEFORE_DELETE	
AFTER_DELETE	
BUTTON_CLICK	BUTTON_CLICK
START_APP	
CLOSE_APP	
BEFORE_OPEN	

clienttype desktop (Visual Basic Skripte)	clienttype web (Javascript)
MOVE	
COPY	
BEFORE_SAVE_DOCUMENT	
AFTER_SAVE_DOCUMENT	
BEFORE_LINK	
AFTER_LINK	
BEFORE_DELETE_LINK	
AFTER_DELETE_LINK	
ENTER_PAGE	ENTER_PAGE
CONTEXT_CHANGED	
BEFORE_UNDO_CHECKOUT	
BEFORE_RESTORE	
AFTER_RESTORE	
BEFORE_CANCEL	BEFORE_CANCEL
START_ACTION	
FOCUS_GAINED	FOCUS_GAINED
VALUE_CHANGED	VALUE_CHANGED
BEFORE_ADD_ROW	BEFORE_ADD_ROW
BEFORE_DELETE_ROW	BEFORE_DELETE_ROW
AFTER_VALIDATE	AFTER_VALIDATE
LEAVE_PAGE	LEAVE_PAGE
CELL_FOCUS_GAINED	CELL_FOCUS_GAINED
CELL_VALUE_CHANGED	CELL_VALUE_CHANGED
BULK_BEFORE_VALIDATE	
BULK_SHOW	
BULK_BUTTON_CLICK	
BLUK_ENTER_PAGE	
BULK_AFTER_VALIDATE	
FILE_DROP	

clienttype desktop (Visual Basic Skripte)	clienttype web (Javascript)
KERNEL_BEFORE_JOB	
KERNEL_AFTER_JOB	
JOB_BEFORE_OBJECT	
JOB_AFTER_OBJECT	
BEFORE_START_ARCHIVE_BATCH	
INITIALIZE_MEDIUM	
BEFORE_OPEN_MEDIA	
CLOSE_MEDIA	
FINISH_MEDIA	
FINISH_ARCHIVE_BATCH	
ARCHIVE_ERROR	
OBJECT_HISTORY_ENTRY	
SESSION_LOGIN	
GLOBAL_CLIENT	GLOBAL_WEBCLIENT
GLOBAL_SERVER	GLOBAL_WEBCLIENT_OBJECT_TYPE
GLOBAL_OBJECT_TYPE	

DocumentFileService (/documentfiles)

/osrest/api/documentfiles/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert Detailinformationen für das Objekt mit der gegebenen ID. Das Ergebnis enthält folgende Informationen:

- *fav*: Das Dokument wurde vom Benutzer als Favorit markiert.
- *typeless*: Es handelt sich um ein typlosees Dokument.
- *contentviewerUrl*: URL zum Öffnen des Objekts in der kombinierten Ansicht von enaio® contentviewer.
- *contentviewerDocumentUrl*: URL zum Öffnen des Objekts in der Dokumentenansicht von enaio® contentviewer.
- *contentviewerIndexdataUrl*: URL zum Öffnen des Objekts in der Indexdatenansicht von enaio® contentviewer.
- *oswebLinkUrl*: URL zum Öffnen des Objekts in enaio® webclient.
- *files*: Die Liste der mit dem Dokument verbundenen Dateien.

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

/osrest/api/documentfiles/[id]/[page]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` application/octet-stream

Die Methode liefert eine bestimmte Seite für das Objekt mit dem gegebenen Seitennamen bzw. der Seitennummer (siehe /osrest/api/documentfiles/[id]) zurück. Bei einseitigen Dokumenten kann also immer die 1 übergeben werden.

/osrest/api/documentfiles/[id]/pdf

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` application/octet-stream

Optionale Paramter sind:

- forceprint (Boolean): Wenn gesetzt, wird Programmcode in das PDF eingefügt, welchen PDF-Reader zur direkten Darstellung des Druckendialoges veranlasst.

Die Methode konvertiert die Dokumente des Objekts mit der gegeben ID ins PDF-Format.

/osrest/api/documentfiles/pdf

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` application/pdf

Optionale Parameter sind:

- whitepage (Boolean): Wenn true, wird zwischen jedem Dokument im zusammengeführten Dokument eine leere Seite als Trenner eingefügt.
- forceprint (Boolean): Wenn true, wird Programmcode in das PDF eingefügt, der den PDF-Reader veranlasst, den Druckendialog zu öffnen.

Die Methode führt die PDF-Version mehrerer Originaldokumente, welche mittels ID im POST-Body übermittelt wurden, in ein PDF-Dokument zusammen und liefert dieses im Anschluss aus. Über den GET-Parameter *forceprint* kann dem Ziel-PDF-Dokument Programmcode hinzugefügt werden, der den PDF-Reader dazu veranlasst, nach dem Öffnen des PDFs direkt den Druckendialog anzuzeigen. Maximal 200 Dokumente können in ein PDF

zusammengeführt werden. Wenn ein oder mehrere Dokumente nicht verarbeitet werden können, so wird kein PDF als Ergebnis zurückgeliefert. Stattdessen ist das Ergebnis ein JSON-Objekt, welches die fehlgeschlagenen Objekt-IDs enthält.

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

/osrest/api/documentfiles/[id]/zip

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` application/octet-stream

Optionale Parameter sind:

- filename (String): Dateiname für die ZIP Datei und die enthaltene Datei. Der Wert wird statt der Objekt Id eingesetzt.

Die Methode fasst die Dokumente des Objekts mit der gegebenen ID in einer ZIP-Datei zusammen.

/osrest/api/documentfiles/zip

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` application/zip

Die Methode packt die Originaldokumente, welche mittels ID im POST-Body übermittelt wurden, in ein ZIP-Archiv und liefert dieses im Anschluss aus. Wahlweise kann für jedes angeführte Dokument neben der ID auch der Name im ZIP-Archiv angegeben werden. Zuzüglich zu den zu packenden Dokumenten muss über den Tag *archivename* der Dateiname für das zurück zu liefernde ZIP-Archiv angegeben werden. Maximal 200 Dateien kann ein ZIP-Archiv enthalten. Wenn ein oder mehrere Dokumente nicht verarbeitet werden können, so wird kein ZIP als Ergebnis zurückgeliefert. Stattdessen ist das Ergebnis ein JSON-Objekt, welches die fehlgeschlagenen Objekt-IDs enthält.

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

/osrest/api/documentfiles/[id]/osfile

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` application/octet-stream

Die Methode erstellt eine enaio®-interne Linkdatei für das Objekt mit der gegebene ID.

/osrest/api/documentfiles/addtotray

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über einen Multipart POST Request (RFC 1867) kann eine Datei in die Benutzerablage hochgeladen werden.
Die Datei muss eine Content Disposition 'form-data' und das Attribut *filename* mit dem Dateinamen enthalten.
`content-disposition: form-data; name="imagefile"; filename="image.jpg"`

/osrest/api/documentfiles/move/[id]?targetLocation=[id des Ordners oder Registers]&sourceLocation=[id des aktuellen Ordners/Registers]

- `Unterstützte Anfragemethoden:` POST

Die Methode verschiebt das Objekt an den in "targetLocation" angegebenen Ort und ist somit eine Pflichtangabe. Wenn erfolgreich, wird "OK" zurückgesendet, ansonsten eine Exception. Wenn das zu verschiebende Objekt ein Ordner oder Register ist, kann zusätzlich über "recursive" angegeben werden, ob sein Inhalt mit verschoben werden soll.
Optionale Parameter sind:

- recursive (Boolean): Wenn Objekt ein Ordner oder Register ist, kann darüber angegeben werden, ob sein Inhalt mit verschoben werden soll, wenn der Parameter 'true' ist.
- sourceLocation (Integer): Gibt den aktuellen Standort (direktes Elternobjekt) des Objekts an. Nur Pflicht, wenn das Dokument mehrere Standorte besitzt.

/osrest/api/documentfiles/move

- `Unterstützte Anfragemethoden:` POST

Die Methode verschiebt ein Objekt von einem Standort zu einem anderen. Wenn das zu verschiebende Objekt ein Register ist, kann zusätzlich über "recursive" angegeben werden, ob sein Inhalt mit verschoben werden soll. Im Gegensatz zum anderen Move-Aufruf, können hier weitere Informationen zu den einzelnen Objekten mitgegeben werden, wodurch der Move-Aufruf beschleunigt wird. Alle Angaben außer der item ID und der target ID sind optional.
Optionale Parameter sind:

- recursive (Boolean): Wenn Objekt ein Ordner oder Register ist, kann darüber angegeben werden, ob sein Inhalt mit verschoben werden soll, wenn der Parameter 'true' ist.

Beispiel JSON

```
{
      item: {
            id: "1114",
            objectTypeId: "34586"
      },
      target: {
            id: "61",
            objectTypeId: "485239"
      },
      source: {   // die source property wird nur benötigt, wenn
sich das zu verschiebende Objekt an mehreren Standorten
befindet
            id: "88"
      },
      archive: { // Die ObjektTypeId vom Schrank-Objekt. Durch
diese Angabe wird die Verschiebung beschleunigt. Ist jedoch
optional.
            id: "26"
      }
 }
```

/osrest/api/documentfiles/droptargets

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die zurückgegebene Liste enthält die Namen verfügbarer Droptargets mit Ihren jeweiligen Variablen. Ist zwischen einer Variable und einem Feld in enaio® eine 1:1-Zuordnung gegeben, wird die für das enaio®-Feld gültige maximale Feldlänge, der Pflichtfeldstatus und mögliche Katalogdaten zurückgegeben. Ist keine eindeutige Zuordung, z. B. bei mehrfacher Verwendung der Variable, möglich, wird nur der Variablenname zurückgegeben.

```
{
    demo: {
        regtype: {
            catalog: [
                "Rechnungen",
                "Altbestand",
                "Sonstiges",
                "Support-Infos",
                "Projekt",
                "Meilenstein",
                "Schriftverkehr",
                "Marketing",
            ]
            required: true,
            size: "15"
        },
        dkreditor: { },
        dstatus: { },
        fname: {
            required: true,
            size: "80"
        },
```

```
                gposition": {
                   columns: [
                            "Artikelnummer"
                            "Bezeichnung",
                            "Einzelpreis",
                            "Menge",
                            "Positionspreis"
                    ]
                    type: "GRID",
                }
    }
}
```

In enaio® appconnector können Droptarget-Scripts für spezifische Benutzer abgelegt werden. Dafür müssen die Skripte unterhalb des Verzeichnisses *droptargets* in der enaio® appconnector-Konfiguration in ein Verzeichnis namens *user/<benutzername>* abgelegt werden. Für den Benutzer *demo* würde die Verzeichnissstruktur z. B. so aussehen: *droptargets/users/demo*

/osrest/api/documentfiles/droptargets/[targetname]

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über die Methode kann eine Datei an ein Droptarget übergeben werden. Der Request muss den Contenttyp *multipart/form-data* haben (RFC 1867). Der Multipart Request muss aus zwei Teilen bestehen:

- Variablen Daten: Dieser Teil muss die Content Disposition 'form-data' und das Attribut *name* mit dem Namen *data* enthalten.

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

- Datei Upload: Dieser Teil muss die Content Disposition 'form-data' und das Attribut *filename* mit dem Dateinamen enthalten. Das Attribut *name* hat hier keine Bedeutung.

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

Um Tabellen (*type: GRID*) zu befüllen, müssen die einzelnen Zeilen mit Ihren Spaltenübeschriften als Parameter übergeben werden. Leere Spalten können weggelassen werden:

```
{
   fname:"0008-OS",
   regtype:"Projekt",
   dkreditor:"indat",
   gposition:{
      row_1:{
         Artikelnummer:123,
         Bezeichnung:"Katze",
         Einzelpreis:1000
      },
      row_2:{
         Artikelnummer:666,
         Bezeichnung:"Maus",
```

```
            Einzelpreis:12,
            Menge:55
         }
      }
}
```

Notation 2.0:

```
"position":[{"Typ":"Artikel","Nummer":"1"},{..}]
```

Die Methode gibt die enaio®-Objekt-ID des neu angelegten Dokuments zurück, der genaue Rückgabewert hängt jedoch von der Einstellung im Droptarget-Script selbst ab.

/osrest/api/documentfiles/delete

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über die Methode können mehrere Objekte im Archiv gelöscht werden. Bei Erfolg gibt die Methode den String *"true"* zurück. Der Post-Body der Anfrage besteht aus einer JSON-Array.
Sollte ein Fehler beim Löschen eintreten, gibt die Methode eine Liste mit den Objekte zurück, die nicht gelöscht werden konnten.
Optionale Parameter sind:

- recursive (Boolean): Wenn Objekt ein Ordner oder Register ist, kann darüber angegeben werden, ob sein Inhalt mit gelöscht werden soll, wenn der Parameter 'true' ist.

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

/osrest/api/documentfiles/delete/[id]

- `Unterstützte Anfragemethoden:` GET, DELETE
- `Unterstützte Ergebnisformate:` JSON

Über die Methode kann ein Objekt im Archiv gelöscht werden. Bei Erfolg gibt die Methode den String *"true"* zurück.
Optionale Parameter sind:

- recursive (Boolean): Wenn Objekt ein Ordner oder Register ist, kann darüber angegeben werden, ob sein Inhalt mit gelöscht werden soll, wenn der Parameter 'true' ist.

/osrest/api/documentfiles/update/[id]

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` -

Über die Methode kann die Datei zu einem Dokument geändert werden. Der Request muss den Contenttyp *multipart/form-data* haben (RFC 1867). Der Multipart Request muss wie folgt aufgebaut sein:

- Datei Upload: Der Teil muss die Content Disposition 'form-data' und das Attribut *filename* mit dem Dateinamen enthalten. Das Attribut *name* hat hier keine Bedeutung.

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

Optionale Parameter sind:

- maintype (int): Es kann ein bestimmter Haupttyp (siehe Server-API-Dokumentation) erzwungen werden.
- page (int): Bei mehrseitigen Dokumenttypen (Bilder) kann hier die übertragene Seite angegeben werden, damit nicht alle Seiten übertragen werden müssen.

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

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode kann ein Dokument (ID) mit Daten aus dem ExtractionService anreichern. Was gemappt wird, legt das Mapping fest.
Optionale Parameter sind:

- override (bool): Legt fest, ob auch nicht leere Indexfelder gefüllt werden sollen (optional). Diese Option, kann auch dediziert im Mapping festgelegt werden.
- preview (bool): Wenn true, wird statt einem Update des Dokuments, das Mapping als "Vorschlag" zurück gesendet, als JSON formatiert.

/osrest/api/documentfiles/processmetadata/[objectTypeId]

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode kann ein Dokument (ID) mit Daten aus dem ExtractionService anreichern, wobei das Dokument nicht in enaio® existieren muss. Die Datei wird als Multipart-Post-Request gesendet. In der URL wird der Objekttyp angegeben, mit dem das Dokument ablegt wird. Mit dem Mapping des Objekttyps wird ein ExtractionMapping von der Datei als JSON zurückgeliefert.

/osrest/api/documentfiles/checkout/{id}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Pflichtparameter sind:

- objectTypeId (int): ObjectTypeId des Objekts

Die Methode setzt ein Dokument (ID) für den über enaio® appconnector angemeldeten Benutzer auf den Status 'ausgecheckt'. Im Erfolgsfall wird ein HTTP 204 (No Content) zurückgegeben. Wenn das Dokument bereits von einem anderen Benutzer ausgecheckt wurde, werden dessen Benutzerdaten als JSON zurückgegeben und ein HTTP 409 (Conflict).

/osrest/api/documentfiles/checkout/undo/{id}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Pflichtparameter sind:

- objectTypeId (int): ObjectTypeId des Objekts

Die Methode setzt ein Dokument (ID), welches vom über enaio® appconnector angemeldeten Benutzer ausgecheckt wurde, auf den Status 'nicht ausgecheckt'. Im Erfolgsfall wird ein HTTP 204 (No Content) zurückgegeben. Wenn das Dokument von einem anderen Benutzer ausgecheckt wurde, so werden dessen Benutzerdaten als JSON zurückgegeben und ein HTTP 409 (Conflict).

NotificationService (/notifications)

/osrest/api/notifications

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller Benachrichtigungen (Abonnements und Wiedervorlagen).
Optionale Parameter sind:

- showown (boolean): Gibt an, ob Benachrichtigungen an sich selbst angezeigt werden sollen (Standard: false).
- starttime (int - Unix Timestamp): Zeitpunkt, ab welchem Benachrichtigungen angezeigt werden sollen (Standard: keine Beschränkung).
- metadata (String): Dateiname eines alternativen Metadatenmappings
- reload (boolean): Falls true, wird das Caching ignoriert und die Benachrichtigungen werden neu vom Server geladen. Der Cache wird aktualisiert.
- user (String): Name eines Benutzers, für den die Benachrichtigungen angezeigt werden sollen. Setzt voraus, dass die IP des Anfragenden in der *osrest.properties* eingetragen ist.
- verbose (boolean): Es wird zusätzlich ein erweitertes generische Metadatenmodell ausgegeben.
- clienttype (String): Angeben, für welchen Client-Typ Workflow-Benachrichtigungen erhalten werden ("web", "mobile", "desktop" bzw. mit bestimmter Sprache z. B. "web_de", "web_en", "web_fr").

Um diese Funktion auch für Workflowbenachrichtigungen zu nutzen, muss die Nutzung des Workflows über die Konfiguration aktiviert werden.

/osrest/api/notifications/revisits

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller Benachrichtigungen zu Wiedervorlagen. Optionale Parameter sind:

- showown (boolean): Gibt an, ob Benachrichtigungen an sich selbst angezeigt werden sollen (Standard: false).
- starttime (int - Unix Timestamp): Zeitpunkt, ab dem Benachrichtigungen angezeigt werden sollen (Standard: keine Beschränkung).
- metadata (String): Dateiname eines alternativen Metadatenmappings
- verbose (boolean): Es wird zusätzlich ein erweitertes generische Metadatenmodell ausgegeben.

/osrest/api/notifications/subscriptions

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller Benachrichtigungen zu abonnierten Objekten.
Optionale Parameter sind:

- showown (boolean): Gibt an, ob Benachrichtigungen an sich selbst angezeigt werden sollen (Standard: false).
- starttime (int - Unix Timestamp): Zeitpunkt, ab dem Benachrichtigungen angezeigt werden sollen (Standard: keine Beschränkung).
- metadata (String): Dateiname eines alternativen Metadatenmappings
- verbose (boolean): Es wird zusätzlich ein erweitertes generische Metadatenmodell ausgegeben.

/osrest/api/notifications/revisits/remove/{id}/{eventDate}

- `Unterstützte Anfragemethoden:` GET

Diese Methode löscht die Benachrichtigung einer Wiedervorlage. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Die Funktion gibt bei Erfolg den Wert 0 zurück. Erforderliche Parameter:

- id: ID des Objekts, für welches die Benachrichtigung entfernt werden soll.
- eventDate: Datum der Benachrichtigung

/osrest/api/notifications/revisits/remove

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Diese Methode löscht mehrere Benachrichtigungen zu Wiedervorlagen. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} anzugeben.
Sollte ein Fehler beim Löschen eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht gelöscht werden konnten.
Erforderliche Parameter:

- id: ID des Objekts, für welches die Benachrichtigung entfernt werden soll.
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage (optional)

POST Beispiel

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

/osrest/api/notifications/revisits/markread/{id}/{eventDate}

- `Unterstützte Anfragemethoden:` GET

Die Methode setzt für eine Benachrichtigung von einer Wiedervorlage den Status *Gelesen* oder *Ungelesen*. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben.
Wenn {reset} als Parameter gesetzt ist, wird die Benachrichtigung als ungelesen markiert.
Die Funktion gibt bei Erfolg den Wert 0 zurück.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung

Optionale Parameter:

- reset: Benachrichtigung als ungelesen markieren

/osrest/api/notifications/revisits/markread

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt für Benachrichtigungen von Wiedervorlagen den Status *Gelesen* oder *Ungelesen*. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Wenn {reset} als Parameter gesetzt ist, werden die Benachrichtigungen als ungelesen markiert.
Sollte ein Fehler beim "Gelesen markieren" eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht als gelesen oder ungelesen markiert werden konnten.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage (optional)

Optionale Parameter:

- reset: Benachrichtigung als ungelesen markieren

POST Beispiel

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

/osrest/api/notifications/revisits/process/{id}/{eventDate}

- `Unterstützte Anfragemethoden:` GET

Die Methode setzt für eine Benachrichtigung von einer Wiedervorlage den Status *Bearbeitet* oder *Unbearbeitet*. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Wenn {open} als Parameter gesetzt ist, wird die Benachrichtigung als *Unbearbeitet* markiert.
Die Funktion gibt bei Erfolg den Wert 0 zurück.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung

Optionale Parameter:

- reset: Benachrichtigung als unbearbeitet markieren
- password: Base64-Kodiertes Passwort für den eigenen Account, falls die Wiedervorlage eine Passwortvalidierung verlangt. Verlangt sie dies und wird kein Passwort übermittelt, so schlägt die Aktion fehl.

/osrest/api/notifications/revisits/process

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt für Benachrichtigungen von Wiedervorlagen den Status *Bearbeitet* oder *Unbearbeitet*. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Wenn {reset} als Parameter gesetzt ist, werden die Benachrichtigungen als *Unbearbeitet* markiert. Sollte ein Fehler beim "Bearbeitet markieren" eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht als bearbeitet oder unbearbeitet markiert werden konnten.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage (optional)

Optionale Parameter:

- reset: Benachrichtigung als unbearbeitet markieren

POST Beispiel

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

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

- `Unterstützte Anfragemethoden:` GET

Diese Methode löscht die Benachrichtigung eines Abonnements. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Die Funktion gibt bei Erfolg den Wert 0 zurück.
Erforderliche Parameter:

- id: ID des Objekts, für welches die Benachrichtigung entfernt werden soll.
- eventDate: Datum der Benachrichtigung

Optionale Parameter:

- senderId: Der Auslöser der Wiedervorlage. Ist dieser nicht angegeben oder 0, so werden alle Benachrichtigungen des Benutzers von dem Objekt gelöscht.

/osrest/api/notifications/subscriptions/remove

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Diese Methode löscht mehrere Benachrichtigungen zu Abonnements. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} anzugeben.
Sollte ein Fehler beim Löschen eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht gelöscht werden konnten.
Erforderliche Parameter:

- id: ID des Objekts, für welches die Benachrichtigung entfernt werden soll.
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage. Ist dieser nicht angegeben oder 0, so werden alle Benachrichtigungen des Benutzers von dem Objekt gelöscht.

POST Beispiel

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

/osrest/api/notifications/subscriptions/markread/{id}/{eventDate}

- `Unterstützte Anfragemethoden:` GET

Die Methode setzt für eine Benachrichtigung eines Abonnements den Status *Gelesen* oder *Ungelesen*. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben.
Wenn {reset} als Parameter gesetzt ist, wird die Benachrichtigung als ungelesen markiert.
Die Funktion gibt bei Erfolg den Wert 0 zurück.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung

Optionale Parameter:

- reset: Benachrichtigung als ungelesen markieren

/osrest/api/notifications/subscriptions/markread

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt für Benachrichtigungen von Abonnements den Status *Gelesen* oder *Ungelesen*. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Wenn {reset} als Parameter gesetzt ist, wird die Benachrichtigung als ungelesen markiert. Sollte ein Fehler beim "Gelesen markieren" eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht als gelesen oder ungelesen markiert werden konnten.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage (optional)

Optionale Parameter:

- reset: Benachrichtigung als ungelesen markieren

POST Beispiel

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

/osrest/api/notifications/subscriptions/process/{id}/{eventDate}

- `Unterstützte Anfragemethoden:` GET

Die Methode setzt für eine Benachrichtigung von einem Abonnement den Status *Bearbeitet*. Dazu sind sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Die Funktion gibt bei Erfolg den Wert 0 zurück.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung

- password: Base64-Kodiertes Passwort für den eigenen Account, falls das Abonnement eine Passwortvalidierung verlangt. Verlangt es dies und wird kein Passwort übermittelt, so schlägt die Aktion fehl.
- confirmed: true|false. Verlangt das Abonnement eine Bestätigung, so muss hier true übergeben werden. Wird in einem solchen Falle false übergeben, so schlägt die Aktion fehl.

/osrest/api/notifications/subscriptions/process

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt für Benachrichtigungen von Wiedervorlagen den Status *Bearbeitet* oder *Unbearbeitet*. Dazu sind jeweils sowohl die Objekt-ID {id} als auch das Datum {eventDate} (beides Felder einer Benachrichtigung) anzugeben. Sollte ein Fehler beim "Bearbeitet markieren" eintreten, gibt die Methode eine Liste mit den Benachrichtigungen zurück, die nicht als bearbeitet oder unbearbeitet markiert werden konnten.
Erforderliche Parameter:

- id: ID des Objekts
- eventDate: Datum der Benachrichtigung
- senderId: Der Auslöser der Wiedervorlage (optional)

POST Beispiel

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

/osrest/api/notifications/subscriptionQueries

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode gibt alle abonnierten Anfragen des Benutzers zurück. Für eine Erläuterung der Konstanten von "confirm", "notifyType", "actions" siehe unten.
Ergebnis

```
[
	{
		infoText: "Für Admin Gruppe",
		confirm: "NO_CONFIRMATION",
		aboGroup: "1263CA058C1841A5B5BD0F07C77BA901",
		notifyType: "INTERNAL",
		actions: [
			"INDEXDATA_MODIFIED"
		]
```

```
        }
        ...
]
```

/osrest/api/notifications/subscriptionObjects

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Optionale Parameter:

- includeGroupSubscriptions (boolean): Es werden auch die abonnierten Objekte ausgegeben, die für Gruppen angelegt wurden in denen der Benutzer mitglied ist.
- verbose (boolean): liefert zu jedem abonnierten Objekt zusätzliche Informationen zum zugehörigen ECM Objekt

Die Methode liefert eine Liste aller abonnierten Objekte des Benutzers.
Actions kann hierbei folgende Werte annehmen:

Schlüssel	Beschreibung
`INDEXDATA_MODIFIED`	Die Indexdaten wurden geändert.
`DOCUMENT_MODIFIED`	Ein Dokument wurde dem Objekt hinzugefügt oder es wurde geändert.
`OBJECT_CREATED`	Ein Objekt wurde in dem Ordner/Register hinzugefügt.
`OBJECT_DELETED`	Ein Objekt wurde aus dem Ordner/Register entfernt.
`OBJECT_ADDED`	Ein Objekt wurde aus einem anderen Schrank dem Ordner/Register hinzugefügt.
`OBJECT_MOVED_FROM_TRAY`	Ein Objekt aus der Benutzerablage wurde dem Ordner/Register hinzugefügt.
`LOCATION_ADDED`	Ein Objekt in dem Ordner/Register wurde an einen anderen Ort kopiert.

Confirm kann folgende Werte annehmen:

Schlüssel	Beschreibung
`NO_CONFIRMATION`	Keine Bestätigung erforderlich.
`CONFIRMATION`	Eine Bestätigung ist vor dem Löschen der Benachrichtigung notwendig.
`CONFIRMATION_PASSWORD`	Eine Bestätigung inkl. Profil-Passworteingabe ist vor dem Löschen der Benachrichtigung notwendig.
`CONFIRMED`	Eine Bestätigung ist erfolgt.

NotifyType kann folgende Werte annehmen:

Schlüssel	Beschreibung

INTERNAL	Die Benachrichtigung erfolgt über enaio® client.
EMAIL	Die Benachrichtigung erfolgt via E-Mail.

Ergebnis

```
[
	{
	"objectId": 1235,
	"infoText": "Freier Text bei Erstellung des Abonnements",
		"confirm": "NO_CONFIRMATION",
		"aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",
		"notifyType": "INTERNAL",
		"actions": [
			"OBJECT_DELETED"
		],
		"groupsToBeNotified": [
			{
				"id": 8522,
				"name": "PERSONAL"
			}
		],
		"usersToBeNotified": [
			{
				"id": 77,
				"name": "DEMO",
				"fullname": "Klaus Mustermann"
			}
		]
	},
	...
]
```

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

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert eine Liste aller für den Benutzer eingerichteten Benachrichtigungen zu einem Objekt. Für eine Erläuterung der Konstanten von "confirm", "notifyType", "actions" siehe oben.

Ergebnis

```
[
	{
	"objectId": 1235,
	"infoText": "Freier Text bei Erstellung des Abonnements",
		"confirm": "NO_CONFIRMATION",
		"aboGroup": "7cf2ebaf2f7a451f8014a35d397996fe",
		"notifyType": "EMAIL",
		"actions": [
			"OBJECT_DELETED"
		],
		"groupsToBeNotified": [
			{
				"id": 8522,
				"name": "PERSONAL"
			}
		],
		"usersToBeNotified": [
```

```
                        {
                              "id": 77,
                              "name": "DEMO",
                              "fullname": "Klaus Mustermann"
                        }
                  ]
            },
            ...
      ]
```

/osrest/api/notifications/subscriptionObjects

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann ein Abonnement zu dem mit ID definierten Objekt hinzugefügt oder geändert werden. Das hinzuzufügende bzw. zu ändernde Abonnement muss als POST im JSON-Format so reingereicht werden, wie es über die gleichnamige GET-Methode erhalten wurde. Soll ein neues Abonnement erstellt werden, so muss die *aboGroup* leer gelassen werden. Im Änderungsfall muss dort die *aboGroup* des betreffenden Abonnements mitgesendet werden. Als Rückgabewert gibt die Methode im Erfolgsfall 0 zurück, ansonsten einen Fehler. Für eine Erläuterung der Konstanten von "confirm", "notifyType", "actions" siehe oben.

Beispiel POST-Body-JSON

```
{
    "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": "Klaus Mustermann"
            },
            ...
      ]
}
```

/osrest/api/notifications/aboGroup/{id}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Pflichtparameter:

- id (String): AboGrp ID eines abonnierten Objektes oder einer abonnierten Anfrage

Diese Methode liefert spezifische Informationen zu einem abonnierten Objekt oder einer abonnierten Anfrage zurück mit der gegebenen AboGrp ID.

/osrest/api/notifications/subscriptionMultiObjects

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über diese Methode können mehrere Abonnements zu Objekten hinzugefügt oder geändert werden. Die hinzuzufügenden bzw. zu ändernden Abonnements müssen als POST im JSON-Format so reingereicht werden, wie sie über die gleichnamige GET-Methode erhalten wurden. Da diese Methode mehrere Abonnements gleichzeitig ändern kann, ist die oberste JSON-Ebene ein Array der JSON-Abonnement-Objekte. Neue Abonnements zeichnen sich durch eine leere *aboGroup* aus, bei zu ändernden Abonnements ist die *aboGroup* ausgefüllt. In einem POST-JSON-Array können sowohl neue als auch zu ändernde Abonnements angegeben werden. Rückgabewert der Methode ist ein Array aller fehlgeschlagenen Objekte. Für eine Erläuterung der Konstanten von "confirm", "notifyType", "actions", siehe oben.

Beispiel POST-Body-JSON

```
[
	{
	"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": "Klaus Mustermann"
                    },
                    ...
                ]
        },
        ...
]
```

/osrest/api/notifications/subscriptionObjects/delete/{id}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann ein Abonnement gelöscht werden. Als ID muss die *aboGroup* des Abonnements reingereicht werden.
Im Erfolgsfall gibt die Methode 0 zurück, ansonsten einen Fehler.

/osrest/api/notifications/subscriptionObjects/delete

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Über diese Methode können mehrere Abonnements zu Objekten gelöscht werden. Das JSON im POST-Body ist ein String-Array mit den *aboGroup*-Elementen, die gelöscht werden sollen. Die Methode gibt alle *aboGroup*-Elemente zurück, deren zugehöriges Abonnement nicht gelöscht werden konnte.

Beispiel POST-Body-JSON

```
[
        {
                "aboGroup": "7cf2ebaf2f7a451f8014a35d397996fd"
        },
        ...
]
```

/osrest/api/notifications/revisitObjects

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert eine Liste aller Wiedervorlagen des Benutzers.

Ergebnis

```
[
        {
        "osid": 14008672,
        "duedate": 1433847600000,
        "subject": "",
        "creationdate": 1433847417000,
        "notified": false,
        "confirm": false,
        "notifiedbymail": false,
        "sender": {
```

```
                "id": 77,
                "name": "DEMO",
                "fullname": "Klaus Mustermann"
        },
        "recipients": [
                {
                "id": 77,
                "name": "DEMO",
                "fullname": "Klaus Mustermann"
        },
        ...
        ]
},
...
]
```

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

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert eine Liste aller für den Benutzer eingerichteten Wiedervorlagen zu einem Objekt.

Ergebnis

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

/osrest/api/notifications/revisitObjects

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann eine Wiedervorlage hinzugefügt oder geändert werden. Die hinzuzufügende bzw. zu ändernde Wiedervorlage muss als POST im JSON-Format so reingereicht werden, wie es über die gleichnamige GET-Methode erhalten wurde.
Die Empfänger-Struktur im JSON muss dupliziert werden und als *newRecipients* hinzugefügt werden. Nur die Benutzer, die in *newRecipients* aufgelistet sind, werden in die neue oder geänderte Wiedervorlage aufgenommen.Soll eine neue Wiedervorlage erstellt werden, so darf kein *creationdate* mitgeliefert werden. Im Änderungsfall hingegen muss das *creationDate* der zu ändernden Wiedervorlage angegeben werden. Bei Erfolg gibt die Methode 0 zurück, ansonsten einen Fehler.

Beispiel POST-Body-JSON

```
{
    "osid": 14008672,
    "duedate": 1433847600000,
    "subject": "",
    "creationdate": 1433847417000,
    "notified": false,
    "confirm": false,
    "notifiedbymail": false,
      "notificationmail": "E-Mailadress if notifiedbymail is
true",
      "sender": {
        "id": 77,
        "name": "DEMO",
        "fullname": "Klaus Mustermann"
    },
    "recipients": [
            {
            "id": 77,
            "name": "DEMO",
            "fullname": "Klaus Mustermann"
        },
        ...
    ],
    "newRecipients": [
            {
            "id": 77,
            "name": "DEMO",
            "fullname": "Klaus Mustermann"
        },
        ...
    ]
}
```

osrest/api/notifications/revisitMultiObjects

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über diese Methode können mehrere Wiedervorlagen zu Objekten hinzugefügt oder geändert werden. Die hinzuzufügenden bzw. zu ändernden Wiedervorlagen müssen als POST im JSON-Format so reingereicht werden, wie sie über die gleichnamige GET-Methode erhalten wurden. Da diese Methode mehrere Wiedervorlagen gleichzeitig ändern kann, ist die oberste JSON-Ebene

ein Array der JSON-Wiedervorlagen-Objekte. Neue Wiedervorlagen zeichnen sich durch einen fehlenden *creationDate* aus, bei zu ändernden Wiedervorlagen ist das *creationDate* angegeben. In einem POST-JSON-Array können sowohl neue als auch zu ändernde Wiedervorlagen angegeben werden. Rückgabewert der Methode ist ein Array aller fehlgeschlagenen Objekte.

Beispiel POST-Body-JSON

```
[
      {
      "osid": 14008672,
      "duedate": 1433847600000,
      "subject": "",
      "creationdate": 1433847417000,
      "notified": false,
      "confirm": false,
      "notifiedbymail": false,
            "notificationmail": "E-Mailadress if notifiedbymail
is true",
            "sender": {
            "id": 77,
            "name": "DEMO",
            "fullname": "Klaus Mustermann"
      },
      "recipients": [
                  {
                  "id": 77,
                  "name": "DEMO",
                  "fullname": "Klaus Mustermann"
            },
            ...
      ],
      "newRecipients": [
                  {
                  "id": 77,
                  "name": "DEMO",
                  "fullname": "Klaus Mustermann"
            },
            ...
      ]
      },
      ...
]
```

/osrest/api/notifications/revisitObjects/delete/{id}?userId=X&dueDate=Y

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann eine Wiedervorlage gelöscht werden. Als ID muss die *osid* des Abonnements, die *userId* des Benutzers sowie der *duedate*-Zeitstempel reingereicht werden.
Im Erfolgsfall gibt die Methode 0 zurück, ansonsten einen Fehler.
Verpflichtende Parameter sind:

- userId (int): Definiert den Benutzer (recipient), dessen Wiedervorlage entfernt werden soll.
- dueDate(long - Java Timestamp): dueDate-Wert der Wiedervorlage

/osrest/api/notifications/revisitObjects/delete

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über diese Methode können mehrere Wiedervorlagen gelöscht werden. Der POST-Body ist hierbei auf oberster Ebene ein JSON-Array, das die zu löschenden Wiedervorlagen beinhaltet. Die jeweilige *userId* ist die ID des Benutzers, für den die Wiedervorlage eingerichtet ist.

Beispiel POST-Body-JSON

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

/osrest/api/notifications/workflows

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste aller Benachrichtigungen zu Workflowereignissen.
Optionale Parameter sind:

- starttime (int - Unix Timestamp): Zeitpunkt, ab dem Benachrichtigungen angezeigt werden sollen (Standard: keine Beschränkung).
- metadata (String): Dateiname eines alternativen Metadatenmappings
- clienttype (String): Angeben, für welchen Client-Typ Workflow-Benachrichtigungen erhalten werden ("web", "mobile", "desktop" bzw. mit bestimmter Sprache z. B. "web_de", "web_en", "web_fr").
- verbose (boolean): Es werden ausführlichere Daten zu den Workflows zurückgegeben.

Um diese Funktion zu nutzen, muss die Nutzung des Workflows über die Konfiguration aktiviert werden.

/osrest/api/notifications/count

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON
- `Ab enaio 8.50`

Die Methode liefert die Anzahl von ungelesenen Wiedervorlagen, Abonnements und Workflows zurück. Über Query-Parameter kann spezifiziert werden, welche Werte ermittelt werden sollen.
Optionale Parameter:

- showown (boolean): beachte bei der Anzahl von ungelesenen Wiedervorlagen und Abonnements auch Wiedervorlagen und Abonnements für den Nutzer selbst
- countSubscriptions (boolean): liefere die Anzahl ungelesener Abonnements zurück
- countRevisits (boolean): liefere die Anzahl ungelesener Wiedervorlagen zurück
- countWorkflows (boolean): liefere die Anzahl ungelesener Workflows zurück

Beispiel Ergebnis

```
{
      revisits: 3,
      subscriptions: 1,
      workflows: 1
}
```

SessionService (/session)

/osrest/api/session

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert Informationen zur enaio®-Benutzersession zurück.

- server: Adresse von enaio® server
- port: Port von enaio® server
- username: Benutzername des angemeldeten Benutzers
- osguid: enaio®-interne Guid des angemeldeten Benutzers
- sessionGuid: Guid der aktuellen Benutzersession auf enaio® server

Optionale Parameter:

- refresh (boolean): ignoriert die zwischengespeicherten Daten und ruft die angefragten Daten neu ab

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

/osrest/api/session/login

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Mit dieser Methode ist eine Formular-basierter Anmeldung möglich.
Dafür müssen in einen POST-Request mit dem Content-Type application/x-www-form-urlencoded die folgenden Parameter übergeben werden:

- osrest_username
- osrest_password

Die Methode liefert Informationen zur enaio®-Benutzersession zurück (siehe /osrest/api/session).

/osrest/api/session/logout

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Mit dieser Methode ist ein Beenden (Logout) der aktuellen Benutzersession möglich.

/osrest/api/session/checkPassword

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` HTTP 204 / No Content

Pflicht Parameter sind:

- password (String): Das Passwort des Benutzers kodiert in Base64.

Mit dieser Methode kann das Passwort des Benutzers überprüft werden.
Das Passwort wird nur als Base64-Kodiert übermittelt. Base64-Kodierung ist keine richtige Verschlüsselung. Wir empfehlen zwingend den Einsatz von Transport Layer Security (TLS).

/osrest/api/session/changePassword

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` HTTP 204 / No Content

Mit dieser Methode kann das eigene Passwort geändert werden.
Folgender JSON muss im POST-Body mitgesendet werden:

```
{
      "oldPassword": "Altes Passwort als Base64-Kodiert",
      "newPassword": "Neues Passwort als Base64-Kodiert"
}
```

Das Passwort wird nur als Base64-Kodiert übermittelt. Base64-Kodierung ist keine richtige Verschlüsselung. Wir empfehlen zwingend den Einsatz von Transport Layer Security (TLS).

/osrest/api/session/checklicense/[lizenzmodul]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Mit dieser Methode lässt sich das Vorhandensein von enaio ®-Lizenzmodulen auf dem Server überprüfen. Die Methode liefert einen einfachen JSON-String mit den Werten *true* oder *false* zurück. Im Zweifel muss das jeweilige Lizenzmodul der OSRest-Station zugeordnet sein.

/osrest/api/session/runscript

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Mit dieser Methode lässt sich ein VB-Script auf dem Server ausführen.
In einem POST-Request werden hier Parameter und der Name einer Skriptdatei im Unterverzeichnis *scripts* der Konfiguration übergeben. Wenn lediglich die Parameter in einem serverseitigen Event ausgewertet werden sollen, kann der Parameter *script* auch weggelassen werden.

```
{
	"parameters": {
		"Aktennummer": "4711",
		"GetLastVersion": true
	},
	"script": "beispiel.vbs"
}
```

Der Aufruf gibt die individuellen Rückgabeparameter des Jobs zurück:

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

/osrest/api/session/systemroles

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert eine Liste der Systemrollen des aktuellen Benutzers zurück.
Beispielergebnis

```
[
	"R_CLNT_STORE_SETTINGS",
	"R_WFADM_START",
	"R_WFEDIT_START",
	"R_WFEDIT_ORG_EDIT",
	"R_WFEDIT_MODEL_EDIT",
	"R_SRV_ADMIN",
```

```
        "R_CLNT_OBJHIST",
        "R_CLNT_EXPMOD",
        "R_CLNT_EXPORT",
        "R_CLNT_NOTES",
        ...
]
```

/osrest/api/session/userdesktops

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert den privaten und öffentlichen Desktop als Baum. Optionale Parameter sind:

- refresh (Boolean): Ist der Parameter gesetzt und 'true', wird der Cache ignoriert und die Anfragen neu vom Server abgefragt.

/osrest/api/session/userdesktops/add

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Anfrageformat:` JSON

Mit dieser Methode lassen sich Ordner und Verweise auf DMS-Objekte dem privaten Desktop des aktuellen Benutzers hinzufügen. Ist keine "parentId" gegeben wird das hinzuzufügende Objekt auf oberster Ebene dem privaten Desktop hinzugefügt.
Die "objectTypeId" für DMS Objekte ist optional. Aus Performanzgründen ist es allerdings empfehlenswert sie anzugeben. Nach dem Hinzufügen wird das hinzugefügte Objekt zurückgeliefert.

Ordner hinzufügen

```
{
        "name": "Kundenordner",
        "folderId" : "26279936",
        "type" : "FOLDER"
}
```

DMS Objekt hinzufügen

```
{
        "name": "Vertragsakte 12",
        "id": "4570",
        "objectType" : "131082",
        "folderId": "26279936",
        "type" : "OBJECT"
}
```

/osrest/api/session/userdesktops/remove

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Anfrageformat:` JSON

Mit dieser Methode lassen sich Ordner, Suchanfragen und Verweise auf DMS-Objekte aus dem privaten Desktop des aktuellen Benutzers entfernen.

Ordner entfernen

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

Gespeicherte Suchanfrage entfernen

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

Verweis auf DMS-Objekt entfernen

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

/osrest/api/session/userdesktops/save

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Anfrageformat:` JSON

Diese Methode speichert die übergebenen Daten als gesamten privaten Desktop. Der bisherige Desktop wird dabei überschrieben. Die Struktur des JSON sollte der "userdesktops/get"-Struktur entsprechen. So können Änderungen wie das Anlegen, Umbenennen, Verschieben und Löschen von Desktopeinträgen gespeichert werden. Das Erstellen von Suchanfragen sollte weiterhin über den zugehörigen (storedqueries/add) - Aufruf erfolgen.

/osrest/api/session/settings/load

- `Unterstützte Anfragemethoden:` GET
- `Unterstütztes Anfrageformat:` JSON

Mit dieser Methode lässt sich die AS.INI-Konfiguration im JSON-Format abrufen.

/osrest/api/session/settings/save

- `Unterstützte Anfragemethoden:` POST
- `Unterstütztes Anfrageformat:` JSON

Mit dieser Methode lässt sich die AS.INI-Konfiguration im JSON-Format speichern. Die Methode erwartet eine JSON-Zeichenkette, wie sie über die Load-Methode hier drüber abgerufen werden kann.

Die JSON-Zeichenkette wird im Anschluss in das INI-Format konvertiert und dann im Server als Ganzes gespeichert.
Sie speichern über diese Methode die komplette AS.INI. Dies betrifft viele Softwarekomponenten von enaio. Seien Sie dementsprechend vorsichtig. Der JSON wird 1:1 in das INI-Format überführt und dann im Server gespeichert.

ServiceInfoService (/serviceinfo)

/osrest/api/serviceinfo

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert Informationen über API-Version und Buildrevision des OSREST-Service. Diese Informationen werden im Supportfall abgefragt. Auch werden über diesen Service die capabilites der Installation angezeigt.
Die Systemroles sind Deprecated und entfallen zu 8.50. Bitte hierfür den Aufruf /osrest/api/session/systemroles verwenden!

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

/osrest/api/serviceinfo/counter/{guid}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` Integer

Die Methode liefert den Integerwert des Counters zurück, welcher die angegebene GUID besitzt. Es wird ein neuer Counter angelegt, wenn keiner mit der angegebenen GUID existiert. Über den Resetparameter wird ein bestehender Counter zurückgesetzt, sofern dieser vom Type 0 ist. Bei jedem Aufruf, sofern der Counter bereits existierte und mit dem Aufruf nicht zurückgesetzt wird, wird der Counter um eins erhöht.
Optionale Parameter sind:

- type (Integer): Art des Counter (0 = Manueller zurückzusetzender Counter, 1 = Counter wird täglich zurückgesetzt, 2 = Counter wird monatlich zurückgesetzt, 3 = Counter wird jährlich zurückgesetzt)
- reset (Boolean): Bei einem Type 0 Counter wird dieser auf den Initialwert zurückgesetzt
- initial (Integer): Der Initialwert des Counters wenn dieser Angelegt oder Zurückgesetzt wird.

/osrest/api/serviceinfo/ping

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert die Zeit auf dem OSREST-Server als Timestamp zurück. Die Funktion dient in erster Linie dazu, die Erreichbarkeit des Service sowie die Authentifizierung zu testen.

/osrest/api/serviceinfo/errorTypes/

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Fehlerarten zurück, die die API kennt.

/osrest/api/serviceinfo/errorTypes/{errorCode}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert einen einzelnen Fehlertypen zurück.
Beispiel einer Fehlermeldung

```
{
  "ErrorType": "UNSPECIFIC_ERROR",
  "HttpStatusCode": "Internal Server Error - [500]",
  "DefaultMessage": "It seems to be an internal failure
occurred - this should not had happened! :(",
  "ErrorCode": 0
}
```

/osrest/api/serviceinfo/registry

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert Werte aus der Serverregistry zurück. Es sind jedoch nur die nachfolgend im Beispiel aufgezählten Schlüssel aktuell erlaubt. Möchten Sie andere abfragen, so wenden Sie sich bitte an unser Produktmanagement. Der Rückgabewert ist identisch zum Anfrage-JSON. In den einzelnen Objekten befindet sich dann neben dem Key auch ein Value. Nicht erlaubte Schlüssel sind im Antwort-JSOn nicht enthalten.
Beispiel einer Fehlermeldung

```
[
	{
		"key": "Login\\PwdComplexityDescription",
	},
	{
		"key": "Login\\PwdComplexity",
	}
]
```

OSFileService (/anon)

/osrest/api/anon/osfile/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` application/octet-stream

Die Methode erstellt eine enaio®-interne Linkdatei für das Objekt mit der gegebenen ID. Dieser Service kann ohne Authentifizierung genutzt werden. Über den optionalen Anfrage-Parameter *followactivevariant=true* kann eine Auflösung auf das aktive Dokument bei inaktiven Varianten forciert werden.

WorkflowService (/workflows)

/osrest/api/workflows

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert eine Liste von Workflows, die vom Benutzer gestartet werden können.
Optionale Parameter sind:

- clienttype (String): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

Ergebnis

```
{
    id: "4A3D7DE0352B496F9C4CD8565479B1B3",
    title: "Posteingang (Adhoc)",
    info: "Version 3.0",
    workflowParameters: [
        {
            readonly: false,
            type: "TEXT",
            id: "783241C856A04D17AEE520C9630CFDA1",
            name: "sAntragsteller",
            value: ""
        },
        ...,
},
...
```

/osrest/api/workflows/start

- `Unterstützte Anfragemethoden:` POST/JSON

Diese Methode startet einen Workflow.
Eingabe: Ein Eintrag aus dem Ergebnis von (*/osrest/api/workflows*) evtl. mit angepassten Werten (value).
Optionale Parameter sind:

- clienttype (String): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

```
{
	"id": "16B30DD524614774A64C26346B7A3B01",
	"workflowParameters": [
		{
			"type": "TEXT",
			"id": "55C31B02940B4FC98772C583DD627AC5",
			"name": "sAusgangsparameter",
			"value": "ausgang",
			"readonly": false
		},
		{
			"type": "LIST",
			"id": "45319353B64E47F0B18CC4F0581B24BE",
			"name": "sEingangsparameter",
			"value": "Eingang",
			"readonly": false
		},
		{
			"type": "TEXT",
			"id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
			"name": "sFreigeben",
			"value": "false",
			"readonly": false
		}
	],
	"files": [
		"4133"
	]
}
```

/osrest/api/workflows/startWithData

- `Unterstützte Anfragemethoden:` POST/JSON

Diese Methode startet einen Workflow und legt dabei Dateien in die Workflowablage. Über einen Multipart POST Request (RFC 1867) wird der Workflow gestartet und Dateien in die Workflowablage hochgeladen werden. Diese müssen eine Content Disposition 'form-data' und das Attribut *filename* mit dem Dateinamen enthalten. Der Workflow wird (vgl. /osrest/api/workflow/start) im JSON-Format in eine Content Disposition 'data' als Text übergeben.
Optionale Parameter sind:

-

- clienttype (String): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

/osrest/api/workflows/running

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert eine Liste aller Arbeitsschritte, auf die der Benutzer Zugriff hat. Bei den Parametern handelt es sich um die Parameter, die zur Anzeige im Eingangskorb konfiguriert wurden.
Optionale Parameter sind:

.

- clienttype (String): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").
- verbose (boolean): Es werden ausführlichere Daten zu den Arbeitsschritten zurückgeliefert.
- reload (boolean): Cache ignorieren und WorkItem Daten neu holen

kurzes Ergebnis

```
[
    {
        id: "ABE6D05571CF40968F8447C1556D33CA",
            processID: "0B938F1389DD4877BD178DD1B83A2EBC",
            title: "Initialisierung - Aufgabe: Zur Kenntnis
nehmen",
        info: "Kommentar: Diese Aufgabe muss noch ausgeführt
werden."
            creationTime: 1317038062000,
        personalized: "MEIER",
            substitute: false,
          read: false
    },
      ...
]
```

ausführliches Ergebnis (verbose)

```
[
    {
        id: "ABE6D05571CF40968F8447C1556D33CA",
            processID: "0B938F1389DD4877BD178DD1B83A2EBC",
            iconId: "1073743021",
            activityName: "Initialisierung",
            processName: "Standard-Ad-hoc-Workflow 27",
            processSubject: "Aufgabe: Zur Kenntnis nehmen",
            creationTime: 1465635668000,
            personalized: "MEIER",
            read: false,
            substitute: false,
            overTime: true,
            warningTime: 1465635668000,
```

```
            workflowParameters: [
                  {
                        type: "TEXT",
                        name: "Kommentar",
                        value: "Diese Aufgabe muss noch
ausgeführt werden.",
                        position: 0
                  }
            ],
            workflowId: "817B9AD47E2E44FFADFE30413E9AE4C3",
          activityId: "35B71588DEE54343A9FBDF28EFF96B78"
    },
      ...
]
```

/osrest/api/workflows/running/full/[id]

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Die Methode liefert alle XML-Datein zu einem WorkItem als JSON zurück. Diese sind: ExtendedAttributes, Parameters, Version, WorkflowType, File, Masks, ProcessResponsible.
Parameter:

- id (string): ID des WorkItems

Optionale Parameter sind:

- refresh (boolean): Cache ignorieren und WorkItem Daten neu holen
- personalize (boolean): Das WorkItem wird beim Öffnen auch personalisier.
- clienttype (string): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

Beispiel Ergebnis

```
{
      "WorkItem": {
            "WorkItemName": "Kenntnisnahme",
      "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",
                  "ActivityId": "477EB9B8EC63432CAB387C62EBA1874C",
            "WorkflowId": "0C0D89936E634F5C8565962BC1C2BE32",
                  "File": {
                        "Lists": {},
                        "Docs": {}
            },
            "Masks": {
                        "Mask": {
                        "MaskFields": {
                              "MaskField": [
                                        {
                                              "ToolTip": "Dies ist
      ein Pflichtfeld!|",
                                              "FieldTop": "100",
                                              "RegularExpression":
      {},
                                              "InpRight": "80",
                                              "InpBottom": "12",
                                              "Flags": "1",
                                              "InternalName": "",
                                              "FieldBottom": "12",
                                              "Name": "Pflichtfeld",
                                              "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]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Die Methode liefert alle Paramter und Dateien zu einer Workflow-Aktivität.
Parameter:

- id (string): ID der Workflow-Aktivität
- personalize (boolean): das WorkItem soll beim Öffnen auch personalisiert werden
- clienttype (string): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").
- verbose (boolean): Reichert die Rückgabe um nähere Informationen zu den Aktenelementen an.

Ergebnis

```
{
    id: "ABE6D05571CF40968F8447C1556D33CA",
    workflowParameters: [
        {
            "readonly": false,
                  "type": "CHECKBOX",
                "id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
                "name": "Eine Checkbox",
                "value": "0",
                "required": false
        },
        {
            "readonly": false,
            "type": "RADIO",
            "id": "0F665AAAA1C34569B6863C3F73498204",
            "name": "erledigt",
            "value": "3",
                  listData: [
                        "RadioButton1",
                        "RadioButton2"
                  ],
                "required": false
        },
            {
            "readonly": false,
                "type": "TEXT",
                "id": "EFF64844A6A241169EE8AB19DC05886A",
                "name": "RegExp Feld",
```

```
                "value": "",
                "regularExpression":
"Value\\{(.*)\\}\\|Message\\{(.*)\\}",
                "required": false
            },
            {
                "readonly": true,
                "type": "DATE",
                "id": "88CA29EB04E54377AC8D3C788CE90906",
                "name": "Datum Readonly",
                "value": "",
                "required": true
            }
    ],
    files: [
        "1452"
    ]
}
```

Ergebnis (verbose)

```
{
    id: "ABE6D05571CF40968F8447C1556D33CA",
    workflowParameters: [
        {
            "readonly": false,
                  "type": "CHECKBOX",
                "id": "5ACF5B12D1BA4C8AA722F1718FA06BBF",
                "name": "Eine Checkbox",
                "value": "0",
                "required": false
        },
            ...
    ],
    files: [
        "1452"
    ],
      verboseFiles: [
            {
                  id: "1452",
                  objectTypeId: "131081",
                  location: "1",
                  workspace: "1",
                  deletable: true,
                  useActiveVariant: false,
                  movable: true,
                  sig: "0",
                  rights: "15",
                  originalId: "1452",
                  display: "1"
            }
      ]
}
```

/osrest/api/workflows/running/processes

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Prozesse zurück, bei denen der Anwender Prozessverantwortlicher ist oder die der Anwender gestartet hat. Die Prozesse sind vergleichbar mit denen, welche im enaio Client unter "Laufende Workflows" gelistet werden.
Optionale Parameter sind:

- refresh (boolean): Cache ignorieren und WorkItem Daten neu holen
- clienttype (string): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

/osrest/api/workflows/running/processes/{processId}/{activityId}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert Informationen zu einer Aktivität eines Prozesses. Dazu gehören die Workflowdaten (workflow), Benutzer die den Workflow aktuell in ihrem Eingangskorb zur Bearbeitung haben (inboxUsers), Prozessverantwortliche (responsibleUsers), Workflowprotokoll (protocolEntries).
Parameter:

- processId (string): Die Prozess Id des Prozesses
- activityId (string): Die Aktivity Id der Instanz des Prozesses

Ergebnis

```
{
      "inboxUsers": [
            {
                  "id": 7450,
                  "name": "MARLON",
                  "fullname": "",
                  "description": "",
                  "locked": "",
                  "limited": "0",
                  "valid": true,
                  "groups": ["STANDARD"],
                  "email": ""
            },
            ...
      ],
      "responsibleUsers": [{
                  "telefon": "",
                  "vorname": "",
                  "stellvertreter": "",
                  "name": "P4IN1",
                  "e-mail": "",
                  "nachname": "",
                  "typ": "PERSON",
                  "id": "AAA55F6209E54F0F8F503B60F0927C2B",
                  "login": "P4IN1",
                  "userid": "00239BAF6A184B5F96DEDB1AD88E1998"
```

```
                },
                {
                        "groupid": "",
                        "stellvertreter": "",
                        "name": "Adhoc",
                        "typ": "ROLE",
                        "id": "727A6EA824B543668A4CF56B3EDD98A9"
                },
                ...
        ],
        "protocolEntries": [
                {
                        "activityId":
"00000000000000000000000000000000",
                        "activityName": "StartActivity",
                        "creationTime": 1472220199000,
                        "endTime": 1472220200000,
                        "processState": "16384"
                },
                {
                        "activityId":
"45DE932B26B94EB3A21BDC00AB9AB9DC",
                        "activityName": "Erster Arbeitsschritt",
                        "creationTime": 1472220200000,
                        "accessTime": 1472220208000,
                        "processState": "128"
                },
                ...
        ]
}
```


/osrest/api/workflows/personalize

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Diese Methode personalisiert für den angemeldeten Nutzer die WorkItems. Dazu ist jeweils die ID des WorkItems anzugeben. Sollte ein Fehler beim Personalisieren eintreten, gibt die Methode eine Liste mit den WorkItems zurück, die nicht personalisiert werden konnten.
Erforderliche Parameter:

- id (string): ID des WorkItems

Optionale Parameter:

- *clienttype (string)*: Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

POST Beispiel


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

```
    ...
]
```

/osrest/api/workflows/depersonalize

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode hebt für den angemeldeten Nutzer die Personalisierung der angegebenen WorkItems auf. Dazu ist jeweils die Id des WorkItems anzugeben. Sollte ein Fehler beim Aufheben der Personalisierung eintreten, gibt die Methode eine Liste mit den WorkItems zurück, deren Personalisierung nicht aufgehoben werden konnte.
Erforderliche Parameter:

- id (string): ID des WorkItems

Optionale Parameter:

- clienttype (string): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

POST Beispiel

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

/osrest/api/workflows/read

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Die Methode setzt für den angemeldeten Nutzer die angegebenen WorkItems auf gelesen. Dazu ist jeweils die ID des WorkItems anzugeben.
Erforderliche JSON Parameter:

- id (string): ID des WorkItems

POST Beispiel

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

/osrest/api/workflows/unread

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Die Methode setzt für den angemeldeten Nutzer die angegebenen WorkItems auf ungelesen. Dazu ist jeweils die Id des WorkItems anzugeben.
Erforderliche JSON Parameter:

- id (string): ID des WorkItems

POST Beispiel

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

/osrest/api/workflows/forward

- Unterstützte Anfragemethoden: POST/JSON

Optionale Parameter:

-
  - clienttype (string): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").

Diese Methode leitet eine Workflow-Aktivität weiter. Hierbei muss das Ergebnis-JSON von */osrest/api/workflows/running/[id]* (ggf. mit angepassten Parameterwerten) übergeben werden.
POST Beispiel

```
{
      id: "A568441085B74CBDB088C7B43838AAEA",
      workflowParameters: [{
            "type": "TEXT",
            "id": "2A1B85B652A5423BAE78CA0C2FBDAA6D",
            "name": "Pflichtfeld",
            "value": "geänderter Text"
            }, ...
      ],
      files: ["1452"],
      (optional) routingList: {
            "id": "3294B433BFF6454D9C861B86B5A8AD5D",
            "processId": "BA16C21BB96D46D099E72070BCB644CC",
            "activityId": "3294B433BFF6454D9C861B86B5A8AD5D",
            "expandable": true,
            "entries": [
```

```
                  {
                        "nr": "203",
                        "expandable": true,
                        "items": [
                              {
                                    "id":
"99825B18A8334987935684FDA3D6A40D",
                                    "activityId":
"6EE4490A48164A0FA6DC34A80099AF66",
                                    "activityName": "Rechnung
erstellen",
                                    "modelActivityName":
"Rechnung erstellen",
                                    "remark": "",
                                    "timerId": "",
                                    "timerDuration": "",
                                    "timerDurationType": "",
                                    "changeable": true,
                                    "deleteable": false,
                                    "objectIds":
"12531E57D10A447D9480EBB2A94A6813,783618F8E4294BCA8B384909CEA32
FD8,5B18392F1ABC4DF5983DEA1338ACCB67"
                              },
                              ...
                        ]
                  },
                  ...
            ]
      },
      (optional) verboseFiles: [
            {
                  id: "1452",
                  workspace: 0
            }
      ]
}
```

Änderungen an der Workflowakte vornehmen

- ein DMS-Objekt der Akte hinzufügen

POST Beispiel, Neues Aktenelement hinzufügen

```
 {
      id: "A568441085B74CBDB088C7B43838AAEA",
      workflowParameters: [],
      files: [],
      verboseFiles: [
            {
                  id: "1527",
                  objektTypeId: "262144",
                  location: "1",          (1 - DMS-Objekt
existiert im Dateisystem, 2 - DMS-Objekt existiert nur in der
Systemablage)
                  workspace: "0",         (0 - Infobereich der
Akte, 1 - Arbeitsbereich der Akte)

                  (optional)
                  deletable: false,
```

```
                    useActiveVariant: false,
                    moveable: false
              }
        ]
}
```

- ein Aktenelement bearbeiten

POST Beispiel, Aktenelement bearbeiten

```
 {
      id: "A568441085B74CBDB088C7B43838AAEA",
      workflowParameters: [],
      files: ["1527"],
      verboseFiles: [
            {
                  id: "1527",

                  (benötigt wird jeweils nur die Property, die
geändert wurde)
                  workspace: "0",         (0 - Infobereich der
Akte, 1 - Arbeitsbereich der Akte)
                  deletable: false,
                  useActiveVariant: false,
                  moveable: false
            }
      ]
}
```

- ein DMS-Objekt aus der Akte löschen

Um ein DMS-Objekt aus der Akte zu löschen, muss die ID des betreffenden Objektes aus der "files"-Property entfernt werden.

/osrest/api/workflows/cancel

- Unterstützte Anfragemethoden: POST/JSON

Optionale Parameter:

- save: true|false. Bei true werden die mitgesendeten Daten gespeichert.
- clienttype (string): Wenn save:true, dann angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr")

Über diese Methode kann eine Workflow-Aktivität mit neuen Werten gespeichert und nicht weitergeleitet werden. Hierbei muss das Ergebnis-JSON von */osrest/api/workflows/running/[id]* (ggf. mit angepassten Parameterwerten) übergeben werden (siehe forward-Aufruf).

/osrest/api/workflows/processes/[Id]

- Unterstützte Anfragemethoden: GET

- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle laufenden Workflowprozesse eines Objekts.
Parameter:

- id (int): OSID des Objekts

Ergebnis

```
[
  {
    "id": "D44ACCBDCD14481ABA425E383501CD83",
    "name": "Test Mobil 0.8 188",
    "subject": "Testworkflow für OS|mobileDMS",
    "state": "RUNNING",
    "creatorId": "217F716436F04D85AFCDE61F7192DD7D",
    "creationTime": 1392730091000,
    "processResponsible": false,
    "activities": []
  },
  {
    "id": "CF51FCD484F246A6A02898F1522FC3E3",
    "name": "Test Mobil 0.8 185",
    "subject": "Testworkflow für OS|mobileDMS",
    "state": "COMPLETED",
    "creatorId": "217F716436F04D85AFCDE61F7192DD7D",
    "creationTime": 1392281546000,
    "processResponsible": false,
    "activities": [
      "Aktivität Client",
      "Aktivität mobileDMS",
      "Aktivität mobileDMS 2"
    ]
  }
]
```

/osrest/api/workflows/abort

- Unterstützte Anfragemethoden: POST/JSON

Diese Methode bricht einen oder mehrere Workflowprozesse ab.
Eingabe: Entweder die Prozess-ID der Workflowinstanz oder eine OSID eines Aktendokumentes. Bei letzteren werden alle Workflowinstanzen abgebrochen, die dieses Aktendokument beinhalten. Es kann also sein, dass mehr als eine Workflowinstanz abgebrochen wird! Zuerst wird die processID ausgewertet. Ist sie vorhanden, wird die osID ignoriert. Ist keine processID vorhanden, wird die osID ausgewertet.

```
{
    "processID":"07D8A5CC07D94BDC9604EA6377085891",
    "osID":"1532"
}
```

/osrest/api/workflows/absence/[true|false]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode meldet den aktuellen Benutzer an der Workflowteilnahme an/ab.
Eingabe: Wird als letztes Pfadelement *true* übergeben, wird der aktuelle Benutzer von der Workflowteilnahme abgemeldet und erhält fortan keine Workflows über den Notifications-Aufruf. Umgekehrt kann er über *false* als letztes Pfadelement an der Teilnahme wieder angemeldet werden. Als Rückgabewert erhält man z.B. folgendes JSON-Ergebnis für das Pfadelement *true*. Im Ergebnisschlüssel *wfAbsence* wird der aktuelle Status zurückgegeben.

```
{
  "null": {
    "id": "AA772557EC874AF48B229B8D4832D61B",
    "wfAbsence": "true"
  }
}
```

/osrest/api/workflows/substitutes

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert die Stellvertreter Benutzer des angemeldeten Benutzers..

```
[
      {
            "id": "178B4625545E4684BDA4652698781598",
            "userGuid": "54AD00BAB3804112B9AC99976138DF0A",
            "userName": "RENE",
            "name": "RENE",
            "absent": false
      },
      ...
]
```

/osrest/api/workflows/organisation

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert die komplette, aktive Workfloworganisation zurück. Jedes Organisationsobjekt wird mitsamt seinen benutzerdefinierten Attributen zurückgegeben und mit einem Verweis auf seine Kindelemente.

Beispielrückgabe

```
{
      "Wurzel": [{
                  name: "Wurzel",
                  id: "914FC91A55D542F79D8ED1BADDE1AFE6",
                  benutzerdefiniert: "origin",
```

```
                        children: [{
                              id: "0B3221875B8D4708AF1CB4C269DF7C9F",
                              typ: "Land"
                        }, {
                              id: "727A6EA824B543668A4CF56B3EDD98A9",
                              typ: "Rolle"
                        }]
                  }],
      "Land": [ ... ],
      "Organisation": [ ... ],
      "Abteilung": [ ... ],
      "BenutzerdefinierterTyp": [ ... ],
      "Rolle": [{
                  id: "92B25680D035458EA6C08B1E703418D0",
                  name: "Adhoc",
                  children: [{
                              id:
"F6C5181094814EFBAF4FF998D90A6D68",
                              typ: "Person" }]
                  }],
      "Person": [{
                  id: "F6C5181094814EFBAF4FF998D90A6D68",
                  name: "Root",
                  telefon: "95378201",
                  ... }],
}
```

/osrest/api/workflows/organisationObjects

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Benutzer und Rollen der aktiven Organisation zurück.

/osrest/api/workflows/organisationObjects

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert zu übergebenen Ids alle Benutzer und Rollen der aktiven Organisation zurück.

```
{
  "organisationObjects": [
      { "id": "AA772557EC874AF48B229B8D4832D61B" }, ...
  ]
}
```

/osrest/api/workflows/clientScripts/[workflowId]/[activityId]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert all Client Skripte und das globale Client Skript zu einer bestimmten Aktivität eines Workflow Modells zurück.
Pflicht Parameter:

- workflowId (String): die ID des Workflow Modells (aufgepasst, die ID des Modells ist nicht die ID des Vorgangsschritts!)
- activityId (String): die ID der Aktivität

Optionale Parameter:

- clienttype (String): Angeben, für welchen Client-Typ die Workflowmaske zum Speichern intern geladen wird ("web", "mobile", "desktop" bzw. für Masken in einer bestimmten Sprache z.B. "web_de", "web_en" oder "web_fr").
- scriptlanguage (String): Liefert nur Skripte in der angegebenen Skriptsprache zurück. Möglich sind: "javascript", "vb-script".

/osrest/api/workflows/adhoc/info

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert alle notwendigen Informationen zu einer spezifischen AdHoc Aktivitäten zurück.
Pflicht Parameter:

- processId (String): die ID eines Workflow Prozesses
- activityId (String): die ID der AdHoc Aktivität

Beispiel Ergebnis


```
{
  "defaultActivityId": "EBA50D7D32C640F5B0358C383F495F9B",
  "activities": [
    {
      "id": "E6EABEEFBFE448BABB36B80C7F6050FC",
      "type": "MULTIINSTANCE",
      "name": "Multi-instance activity",
      "deletable": false,
      "editable": false,
      "performers": []
    },
    {
      "id": "EBA50D7D32C640F5B0358C383F495F9B",
      "type": "WORKITEM",
      "name": "Activity",
      "deletable": false,
      "editable": true,
      "performers": [
        {
          "stellvertreter": "917B282B6CDC470DBCAB532BBC52ECBD",
          "name": "HANS",
          "typ": "PERSON",
          "id": "BE2C9475A0E84700BBD45C7C8B2AECF5",
          "login": "HANS",
          "userid": "B01F6B1429BD4E2B894A065E5C7888F1",
```

```
                                ...
            },
            {
              "groupid": "",
              "stellvertreter": "",
              "name": "Adhoc",
              "typ": "ROLE",
              "id": "727A6EA824B543668A4CF56B3EDD98A9"
            }
          ]
        },
        {
          "id": "7C506F8FF16C487E81DEAFCD5625F04E",
          "type": "ROUTE",
          "name": "Route",
          "deletable": false,
          "editable": false,
          "performers": []
        }
      ],
      "periods": [
        {
          "id": "B7EDA55978AB485B84B7537910B13672",
          "name": "Mahnfrist",
          "duration": 345600
        }
      ]
    }
```

/osrest/api/workflows/adhoc/templates

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert alle Lauflistenvorlagen zurück.

/osrest/api/workflows/adhoc/templates/save

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann eine neue Lauflistenvorlage angelegt werden.

```
{
    "templateId": "",
    "templateName": "Neue Vorlage",
    "templatePublic": true,
    "routingList": {
        "id": "931E96E300AA4F5086EFC46602013274",
        "activityId": "931E96E300AA4F5086EFC46602013274",
        "processId": "CDF769EF60FB47139D52C442D264A91D",
        "expandable": true,
        "entries": [{
            "nr": 0,
            "expandable": true,
```

```
                "items": [{
                    "id": "76EE2219A6144EFF993B25D88D55EEE2",
                    "changeable": true,
                    "deleteable": false,
                    "activityName": "DisplayName",
                    "objectIds": [""],
                    "remark": "",
                    "activityId":
            "EBA50D7D32C640F5B0358C383F495F9B",
                    "modelActivityName": "DisplayName",
                    "timerId": "",
                    "timerDuration": 0,
                    "timerDurationType": 0
                },
                {
                    "id": "C9BBEFEAD8654E19984CDA8FC7C8B831",
                    "changeable": true,
                    "deleteable": false,
                    "activityName": "DisplayName",
                    "objectIds": [""],
                    "remark": "",
                    "activityId":
            "EBA50D7D32C640F5B0358C383F495F9B",
                    "modelActivityName": "DisplayName",
                    "timerId": "",
                    "timerDuration": 0,
                    "timerDurationType": 0
                }]
            },
            {
                "nr": 1,
                "expandable": true,
                "items": [{
                    "id": "C97177555C2540E68285FFCD22E25046",
                    "changeable": true,
                    "deleteable": false,
                    "activityName": "DisplayName",
                    "objectIds": [""],
                    "remark": "",
                    "activityId":
            "EBA50D7D32C640F5B0358C383F495F9B",
                    "modelActivityName": "DisplayName",
                    "timerId": "",
                    "timerDuration": 0,
                    "timerDurationType": 0
                },
                {
                    "id": "AFAC689EC5DB4B41B93935ACEBA46780",
                    "changeable": true,
                    "deleteable": false,
                    "activityName": "DisplayName",
                    "objectIds": [""],
                    "remark": "",
                    "activityId":
            "EBA50D7D32C640F5B0358C383F495F9B",
                    "modelActivityName": "DisplayName",
                    "timerId": "",
                    "timerDuration": 0,
                    "timerDurationType": 0
```

```
                }]
            }]
        }
    }
```

/osrest/api/workflows/adhoc/templates/delete/{id}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann eine Lauflistenvorlage gelöscht werden.
Pflicht Parameter:

- id (String): Die Guid der Lauflistenvorlage.

/osrest/api/workflows/adhoc/templates/move/{id}?publish=(true|false)

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann eine Lauflistenvorlage veröffentlicht oder privatisiert werden.
Pflicht Parameter:

- id (String): Die Guid der Lauflistenvorlage.
- publish (boolean): True, wenn die Lauflistenvorlage veröffentlicht, false wenn sie privatisiert werden soll.

ObjDefService (/objdef)

/osrest/api/objdef/global

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert die komplette globale Objektdefinition als JSON zurück. Das Ergebnis kann auch Teile der Objektdefinition beinhalten, die der Benutzer nicht sehen darf. Alle verlinkten Listen werden aufgelöst und zu normalen Listen umgeformt. Je nach Größe kann das Ergebnis recht umfangreich sein. Der Aufbau ist äquivalent zur XML, lediglich als JSON transformiert.

/osrest/api/objdef/full

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert den kompletten Teil der Objektdefinition als JSON zurück, den der Benutzer sehen darf. Je nach Größe kann der Aufruf etwas dauern und das Ergebnis recht umfangreich sein. Der Aufbau ist äquivalent zur XML, lediglich als JSON transformiert.

/osrest/api/objdef/languages

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode liefert die in der Objektdefinition definierten Sprachen als JSON zurück. Je nach Größe kann der Aufruf etwas dauern und das Ergebnis recht umfangreich sein. Der Aufbau ist äquivalent zur XML, lediglich als JSON transformiert.

/osrest/api/objdef/search/[id]

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: JSON

Diese Methode erwartet eine Objekttyp-ID von einem Ordner, Register oder Dokument und liefert diesen Teil der Objektdefinition als JSON formatiert zurück und der Aufbau des JSON ist äquivalent zur XML.
Parameter:
id: ObjectTypeID des Objekts
Optionale Parameter:

- *refresh:* ruft die Objektdefinition neu ab

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

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Diese Methode erwartet eine oder mehrere Objekttyp-ID von einem Ordner, Register oder Dokument und liefert diese Teile der Objektdefinition als JSON formatiert zurück. Der Aufbau des JSON ist äquivalent zur XML.
Optionale Parameter:

- *refresh:* ruft die Objektdefinition neu ab

POST Beispiel

```
[
      { "id": 22 }, { "id": "413" }, ...
]
```

Ergebnis

```
{
    "objectTypes": [
            {
                  "internal": "addresses",
                  "cotype": "2",
                  ...
            },
            ...
      ]
}
```

Organization Service (/organization)

/osrest/api/organization/users

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert eine Liste aller ECM-Benutzer zurück.

/osrest/api/organization/groups

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert eine Liste aller ECM-Gruppen, in denen der aktuelle Benutzer Mitglied ist oder, wenn der Parameter all gesetzt ist, alle ECM-Gruppen des Systems zurück.
Optionale Parameter sind:

- all (boolean): Es werden alle Gruppen des Systems ausgegeben (Standard: false).
- loadUsers (boolean): Zu jeder Gruppe werden deren Mitglieder (Benutzer) ausgegeben.

/osrest/api/organization/sendmail

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` JSON

Über diese Methode kann eine E-Mail versendet werden. Als POST-Body muss ein Mail-JSON-Objekt übermittelt werden:

JSON Content

```
{
    "receiver": "empfaenger@optimal-systems.de",
    "sender": "absender@optimal-systems.de",
    "subject": "Titel der E-Mail",
    "text": "Inhalt der E-Mail"
}
```

/osrest/api/organization/avatar/{username}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: Image/*.*

Diese Methode liefert das Avatarbild des Benutzers mit dem angegebenen Benutzernamen zum Download zurück. Das Avatar wird hierbei in der Größe 80px ausgeliefert.

/osrest/api/organization/avatar/{username}/{size}

- Unterstützte Anfragemethoden: GET
- Unterstützte Ergebnisformate: Image/*.*

Diese Methode liefert das Avatarbild des Benutzers mit dem angegebenen Benutzernamen zum Download zurück. Das Avatar wird hierbei in der mit *size* gewünschten Größe ausgeliefert.

/osrest/api/organization/user/updateCreate

- Unterstützte Anfragemethoden: POST
- Unterstützte Ergebnisformate: JSON

Über diese Methode kann ein ECM-Benutzer angelegt oder die Daten eines bestehenden ECM-Benutzers geändert werden:

JSON Content

```
{
    "id": 0,
    "name": "BENUTZERNAME",
    "fullname": "Max Mustermann",
    "password": "passwort",
    "description": "Dieser Benutzer hat diese Freitext-
Beschreibung",
    "locked": 0,
    "email": "max@mustermann.de"
}
```

Für die Ausführung werden Supervisor-Rechte benötigt. Wird als ID eine 0 übergeben, so wird der Benutzer angelegt. Ist die ID > 0, so wird der betreffende Benutzer aktualisiert. Eine Aktualisierung des Benutzernamens ist nicht möglich.

/osrest/api/organization/user/delete/{id}

- `Unterstützte Anfragemethoden:` GET

Diese Methode löscht einen ECM-Benutzer unter Angabe seiner ID. Für die Ausführung werden Supervisor-Rechte benötigt. Supervisor-Benutzer können nicht gelöscht werden.
Optionale Parameter sind:

- transferPortfolios (Boolean, default: false): Die Mappen des zu löschenden Benutzers können an einen anderen Benutzer übergeben werden.
- transferNotifications (Boolean, default: false): Die Abonnements und Wiedervorlagen des zu löschenden Benutzers können an einen anderen Benutzer übergeben werden.
- transferUserId(Integer, default: 0): Die Benutzer-ID des ECM-Benutzers, an den Mappen und/oder Abonnements und Wiedervorlagen übergeben werden sollen.

/osrest/api/organization/securitysystem

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` JSON

Diese Methode liefert einen Dump der des Sicherheitssystems zurück. Für den Aufruf muss der User über die DMS Supervisor Systemrolle verfügen.

IconService

Über den IconService können Katalog-Icons zu Trefferlisten-Objekten abgerufen werden. Innerhalb von Trefferlisten existiert dazu ein JSON-Attribut "iconId", welches die ID des Katalog-Icons des Objekts angibt.
Inhalt:

- /osrest/api/icon/preload
- /osrest/api/icon/{id}

/osrest/api/icon/preload

- `Unterstützte Anfragemethoden:` POST
- `Unterstützte Ergebnisformate:` KEINE

Die Methode lädt eine Liste von Katalog-Icons von enaio® server. Somit muss enaio® appconnector später bei der Verwendung von icon/{id} nicht jedes Icon einzeln von enaio® server abrufen. Bei Erfolg wird lediglich ein HTTP 204 Statuscode (erfolgreich, no content) ohne Inhalt zurückgegeben. Im Fehlerfall weicht der HTTP-Statuscode entsprechend ab.

```
{
    "iconIds": [
            1073741986,
        1073741987,
        1073741988,
        1073742158,
        1073742242
```

```
        ]
}
```

/osrest/api/icon/{id}

- `Unterstützte Anfragemethoden:` GET
- `Unterstützte Ergebnisformate:` Image/gif

Diese Methode liefert das Katalog-Icon mit der angefragten ID zur Einbettung in ein <img....>-Tag zurück. Bei mehreren Icons in Trefferlisten wird empfohlen, diese zuvor über die preload-Methode zuladen zu lassen.