API: Interacciones básicas

Control de versiones

Versión	Fecha de modificación	Responsable	Aprobador	Resumen de cambios
1.0	22/11/2023	Anjana Producto	Anjana Producto	Creación del documento. Compatibilidad con la v4.5 de todos los módulos de Anjana

Introducción

Conceptos

La creación de metadatos en Anjana tiene un ciclo de vida específico por los canales habituales (Web App, API). Dicho ciclo consiste en la generación de un objeto con estado DRAFT (o IMPORTED si se ha realizado importando los datos con metadato automático desde TOT), pasado a PENDING mientras el workflow de validación está en proceso hasta su aprobación final, o rechazo.

Este documento presenta distintas operaciones que se pueden realizar para seguir el ciclo propuesto por Anjana y que se presentan en una API. Para información más técnica o necesaria para realizar la llamada, consultar el endpoint en Swagger (URL de acceso: **<http|https>://<host>/swagger/swagger-ui.html**)

En cada endpoint o grupo de endpoints se especifica a qué módulo corresponden, que será el módulo que visitar en Swagger para ver los detalles.

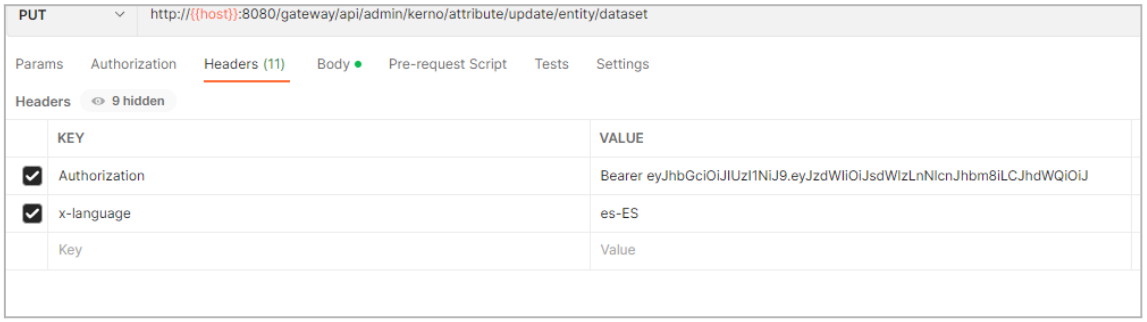
Además, se incluyen instrucciones y ejemplos del mecanismo a emplear para introducir en Anjana metadato que no va a seguir ese ciclo estándar, además de poder obtener listados filtrados del metadato presente.

Al ser simples APIs cualquier herramienta capaz de hacer una petición REST es válida.

En este documento, en los casos que incluyen alguna captura o en los que se especifica alguna acción en las llamadas REST, se utiliza la herramienta de Postman.

Para las peticiones que traten algún texto con multilenguaje o traducciones (recuperar el metadata de un objeto, búsqueda de SolR, etc) se puede especificar en qué idioma se quiere se devuelva la información (el caso más claro es obtener el metadato de un objeto con sus valores con la traducciones en un idioma en particular).

Para ello es necesario incluir la cabecera “x-language”, introduciendo un valor que coincida con uno de los códigos i18n que haya configurados en la aplicación. Como se puede ver en el siguiente ejemplo:



Añadir que la cabecera es totalmente opcional, y si no se recibe se devolverán los resultados en el idioma del usuario con el que se realiza la petición o, si no existe, en el idioma por defecto de la aplicación.

Glosario

En todas las url se van a incluir variables que se tendrán que modificar en el momento de ejecutar la llamada. A continuación se incluye una explicación de qué se tiene que poner en cada una.

- host: es la IP o alias donde está desplegado Anjana (ex: google.com)
- port: es el puerto de acceso al servidor web donde esté desplegado Anjana (ex: 8080)
- provider: el proveedor de gestión de identidades que se quiere utilizar, con el nombre que esté configurado en el yml de zeus (ex: azure)
- objectType: el tipo del objeto involucrado en la petición (ex: ENTITY)
- objectSubType: el subtipo del objeto involucrado en la petición (ex: DATASET)
- idObject: el id del objeto involucrado en la petición (ex: 21)
- state: el estado del objeto involucrado en la petición (ex: APPROVED)
- workflow: tipo de workflow (ex: CREATE)
- target: la colección de solr en la que indexar (ex: KERNO)

Minio

Introducción

Para la gestión de ficheros como metadato (UPLOAD_FILE y ARRAY_UPLOAD_FILE) Anjana usa Minio para almacenar los ficheros en un S3 interno.

Instalación y uso

Se requiere la instalación de un cliente de MinIO para interactuar directamente con el S3 interno sin pasar por el portal de Anjana.

Para usar el cliente MinIO se tiene que instalarlo de la siguiente manera:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

después darle permisos de ejecución

```
chmod +x mc
```

y establecer la conexión con el servidor

```
./mc  alias  set  minio  http://{{host}}:9000  {{user}}  {{password}}
```

```
--api S3v4
```

para finalmente copiar el fichero al bucket que corresponda

```
./mc cp xxxxx.pdf minio/dsa
```

Login

Obtención de token

Antes de poder realizar cualquier operación en las APIs es necesario disponer de un token de acceso. Dicho token se recoge al loguearse.

Según el gestor de identidades que se esté utilizando existen distintos endpoints para obtener el token.

Todos estos endpoint se encuentran en el módulo de Zeus.

El payload a enviar en todos los casos será el siguiente:

```
{
  “u”: “usuario”,
  “p”: “contraseña”
}
```

Donde tanto “u” como “p” deben estar codificados en base64.

LDAP

Consiste en una llamada **POST** a http://{{host}}:{{port}}/gateway/public/v4/auth/login/ldap

Este endpoint se conecta al ldap correspondiente usando las credenciales enviadas y devuelve el token necesario para el resto de llamadas, junto con información del usuario y sus permisos (que varían según el estado de la licencia).

Base de datos

Consiste en una llamada **POST** a http://{{host}}:{{port}}/gateway/public/v4/auth/login/local

Este endpoint se conecta a la BD de zeus usando las credenciales enviadas y devuelve el token necesario para el resto de llamadas, junto con información del usuario y sus permisos (que varían según el estado de la licencia).

Resto de providers

Consiste en una llamada **POST** a http://{{host}}:{{port}}/gateway/public/v4/auth/login/oidc/password/{provider}

Este endpoint se conecta al proveedor que se haya especificado de zeus usando las credenciales enviadas y devuelve el token necesario para el resto de llamadas, junto con información del usuario y sus permisos (que varían según el estado de la licencia)

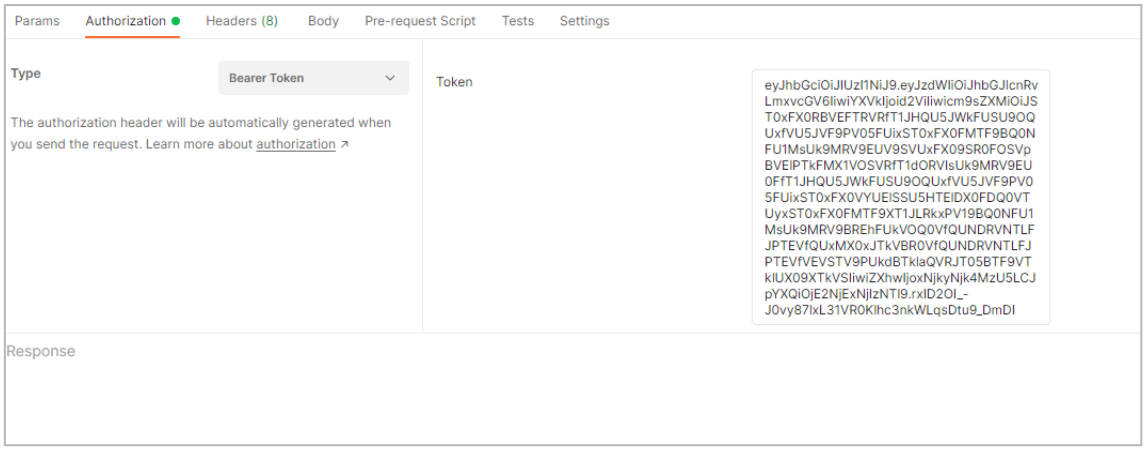
Como añadido actualmente solo es posible este login con AWS y además en Azure en el caso de usuarios sin autenticación multifactor.

Google no permite esta opción de login.

Envío del token

Con el token obtenido en alguno de los endpoints previamente definidos se debe incluir en cualquier petición en la cabecera $Authorization$.

Ejemplo desde POSTMAN:



Ejemplo desde código:

$$headers.add("Authorization", "Bearer " + token);$$

Api Administrativa

Introducción

Para poder insertar, actualizar o borrar metadato en Anjana por la vía administrativa, se necesita un token de un usuario que tenga asignado un rol que contenga el permiso de administración en la API (API_ADMIN).

Inserción/Actualización de Metadato

Estos endpoints permiten la creación o modificación de metadato de los objetos en Anjana.

Todos se encuentran en el módulo de Kerno.

Todos los body que hay que usar para la creación/edición de objetos se pueden conseguir llamando al endpoint

GET
http://{{host}}:{port}/gateway/api/admin/kerno/body-create-update/{objectSubType}/{idObject}

Esta llamada devolverá el body del objeto correspondiente al idObject, que se puede usar para modificaciones sobre este. Para obtener el body necesario para crear un nuevo objeto no se debe mandar el idObject.

Para más información sobre qué se debe incluir como valor según el tipo de campo que es el metadato, consultar la Guía de usuario para ver ejemplos.

La información sobre la composición de las ARIs para campos que utilicen ARIs (aquellos relacionados con ficheros o entidades) se encuentra en el documento de ARIs.

Creación de una entidad no nativa

Este endpoint permite crear una entidad y completar su creación en el estado que se quiera, es decir, se puede crear una entidad con estado APPROVED directamente sin workflow.

Para la creación de entidad consiste en la siguiente llamada:

POST http://{{host}}/gateway/api/admin/kerno/create-update/entity/{objectSubType}

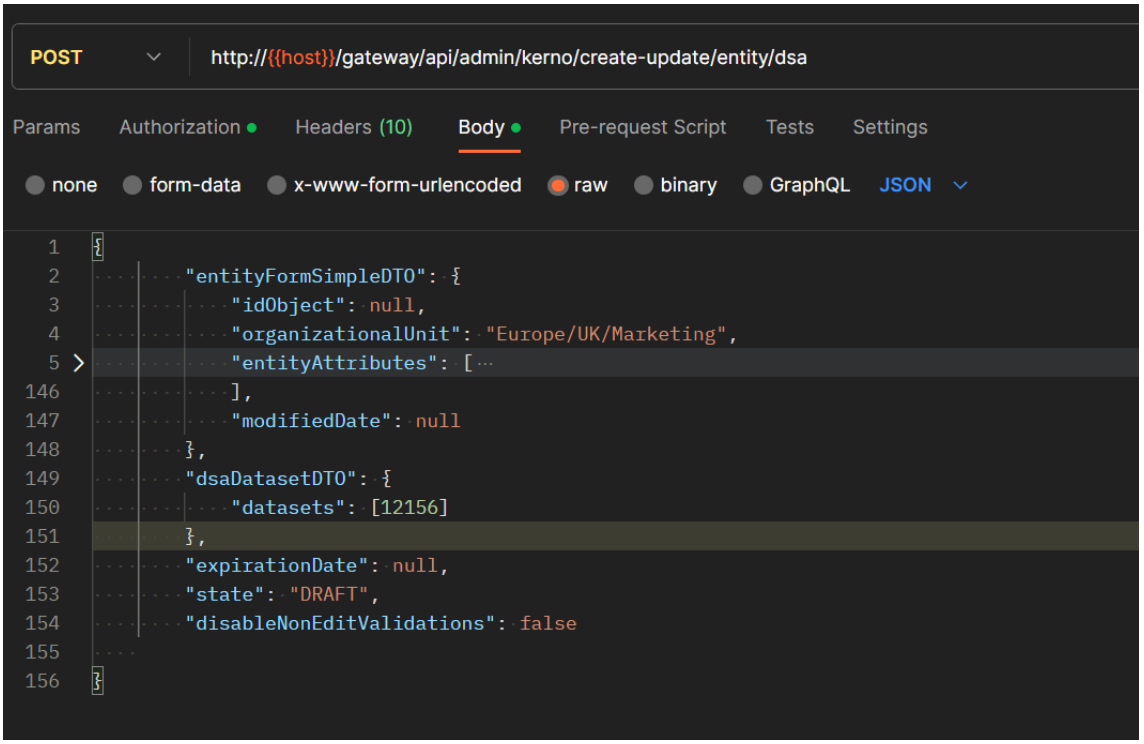
Aplica controles similares a la creación desde el portal, no se permite crear objetos duplicados ni incluir valores incorrectos en los campos.

De esta forma se pueden crear entidades nativas y no nativas.

En caso de querer crear una entidad de tipo DATASET, SOLUTION, INSTANCE o DSA, se recomienda utilizar las apis específicas para estos tipos.

Si se quiere utilizar para la creación de una entidad nativa completa (DATASET, DATASET FIELD, DSA, PROCESS, INSTANCE, SOLUTION), en la sección **Creación de entidades nativas** se amplía la información.

El body que hay que rellenar es similar para todos los subtipos, a continuación se muestra un ejemplo del body necesario para crear un DSA:



Dentro de la propiedad 'entityAttributes' se encuentran todos los atributos del subtipo, el idObject al ser creación debe ser siempre null y las demás propiedades dependen del subtipo que se quiera crear:

- Dataset → Tiene la propiedad 'entityFormDatasetFields' donde se encuentra la información para crear los dataset fields (si se desea crear un dataset sin fields no es necesario rellenarlo, se debe de mandar vacío).
- Dsa → Tiene la propiedad 'dsaDatasetDTO', donde hay que incluir los ids de los datasets que queremos incluir en el DSA.
- Instance → Tiene las propiedades 'wizardDTO' donde hay que incluir la ARI del proceso y de la solución y el nombre que se quiere dar a la instancia, 'datasetInput' y 'datasetOutput' donde se incluyen los ids de los datasets que se quieran añadir, estas dos propiedades son opcionales.
- Process y entidades no nativas → Su estructura es la más simple, con rellenar la unidad organizativa, el estado y los atributos es suficiente.
- Solution → Tiene el atributo 'instances' donde se deben de incluir las instancias relacionadas que se quieran añadir a la solución, es opcional.

Creación de una relación

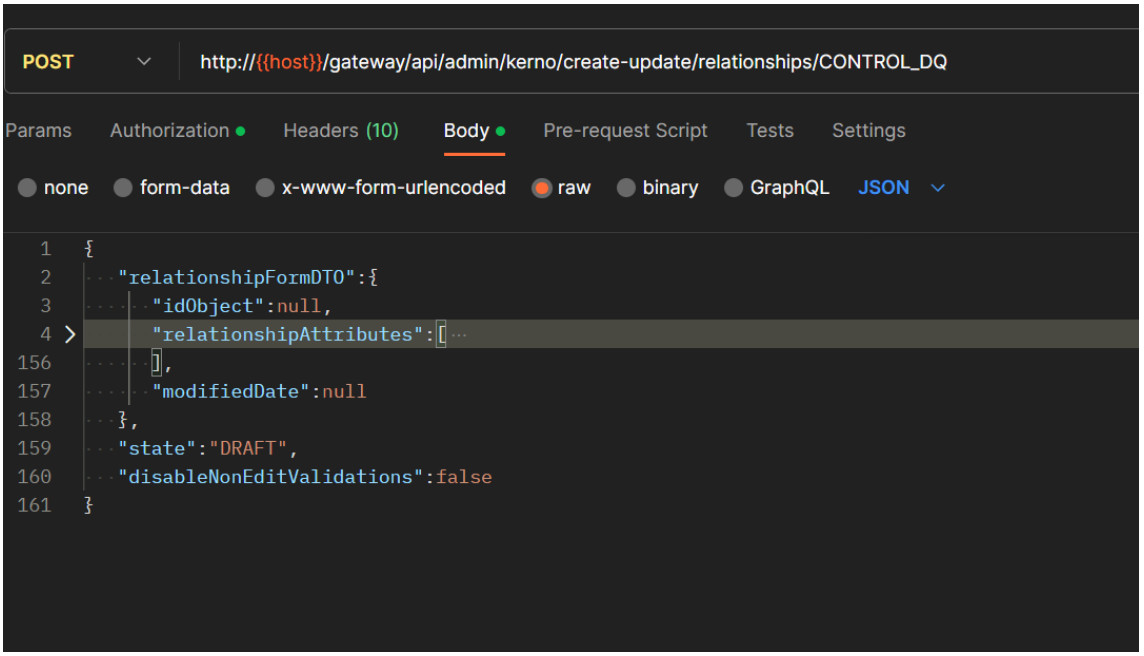
Este endpoint te permite crear una relación y completar su creación en el estado que se quiera, es decir se puede crear algo de cero a APPROVED directamente sin workflow.

Para la creación de relación consiste en la siguiente llamada:

POST http://{{host}}/gateway/api/admin/kerno/create-update/relationships/{objectSubType}

Aplica controles similares a la creación desde el portal, no se permite crear objetos duplicados ni incluir valores incorrectos en los campos.

A continuación se muestra un ejemplo de creación de una relación del subtipo CONTROL_DQ



Dentro de la propiedad 'relationshipAttributes' se encuentran todos los atributos de la relación y el idObject debe ser siempre null para la creación de una nueva relación.

Para relaciones los atributos 'source' y 'destination' son obligatorios

Actualización de una entidad

Este endpoint te permite editar una entidad y completar su edición en el estado que se quiera, es decir se puede editar de APPROVED a DRAFT.

Se trata de la siguiente llamada:

POST
http://{{host}}}/gateway/api/admin/kerno/create-update/entity/{objectSubType}/{idObject}.

El endpoint para actualizar una entidad es el mismo que para la creación pero esta vez hay que rellenar el idObject en la url y en el body que se envía.

Aplica controles similares a la creación desde el portal, no se permite crear objetos duplicados ni incluir valores incorrectos en los campos. El único caso que no se permite es editar una entidad PENDING que ya tiene un workflow en proceso, se debe finalizar el workflow antes de poder editarla.

Se permite editar campos no editables mediante el envío del parámetro de entrada: disableNonEditValidations. Está desactivado por defecto.

En caso de querer actualizar una entidad de tipo DATASET, PROCESS, SOLUTION, INSTANCE o DSA, en la sección Edición de entidades nativas se amplía la información.

Si se quiere utilizar para la actualización de una entidad nativa completa (DATASET, DATASET FIELD, DSA, PROCESS, INSTANCE, SOLUTION) puede ser necesario tener que crear las relaciones internas o usar exclusivamente las relaciones internas si lo único que se quiere editar son los objetos internamente relacionados.

Actualización de una relación

Este endpoint te permite editar una relación y completar su edición en el estado que se quiera, es decir se puede editar de APPROVED a DRAFT.

Para la edición de relación consiste en la siguiente llamada:

POST
http://{{host}}/gateway/api/admin/kerno/create-update/relationships/{objectSubType}/{idObject}

El endpoint para actualizar una relación es el mismo que para la creación pero esta vez hay que rellenar el idObject en la url y en el body que se envía.

Aplica controles similares a la creación desde el portal, no se permite crear objetos duplicados ni incluir valores incorrectos en los campos. El único caso que no se permite es editar una entidad PENDING que ya tiene un workflow en proceso, se debe finalizar el workflow antes de poder editarla.

Se permite editar campos no editables mediante el envío en el body del parámetro de entrada: disableNonEditValidations. Está desactivado por defecto.

Borrado de una entidad

Permite borrar una entidad.

Consiste en la siguiente llamada:

DELETE http://{{host}}/gateway/api/admin/kerno/delete/entity/{idObject}.

Este endpoint borra toda la información de la entidad y aquella relacionada que aplique:

- Borrado de relaciones asociadas a la entidad (con sus atributos)

- Borrado de información de workflows asociados a la entidad (tanto en kerno como el workflow en activiti, que será borrado, junto con todas sus notificaciones)
- Borrado de las posibles combinaciones existentes si el borrado es un DATASET o DSA
- Borrado de la información guardada en el carrito relativa a la entidad a borrar
- Borrado de la propia entidad (con sus atributos)
- Borrado de la información indexada de la entidad (incluyendo snapshots)
- Borrado de la información indexada de las relaciones con la entidad (incluyendo snapshots)

No se podrán borrar procesos ni soluciones que dejen instancias huérfanas.

No se podrán borrar entidades que tengan ningún tipo de relación asociada, se enviará un aviso a los propietarios de los objetos de los extremos de la relación para avisar de que es necesario eliminar primero las relaciones para poder borrar sus objetos relacionados.

Borrado de una relación

Permite borrar una relación.

Consiste en la siguiente llamada:

DELETE http://{{host}}:{port}/gateway/api/admin/kerno/delete/relationship/{idObject}.

Este endpoint borra toda la información de la relación y aquella relacionada que aplique:

- Borrado de información de workflows asociados a la relación(tanto en kerno como el workflow en Activiti, que será borrado, junto con todas sus notificaciones)
- Borrado de la propia relación (con sus atributos)
- Borrado de la información indexada de la relación (incluyendo snapshots)

Obtener metadato de una entidad

Este endpoint permite obtener los datos de un metadato definido de una entidad, dichos datos serán devueltos de la misma manera que reciben en el portal, es decir con la estructura de menús y secciones (incluyendo el menú y secciones ficticias para los campos customs).

Consiste en la llamada:

GET http://{{host}}:{{port}}/gateway/api/admin/kerno/entity/{objectSubType}/{idObject}

Cambiar Unidad Organizativa de una entidad

Te permite cambiar la unidad organizativa de una entidad sin pasar por la validación de workflow que normalmente se aplica.

Consiste en la llamada

POST
http://{{host}}:{{port}}/gateway/api/admin/kerno/change-organizational-unit/{{idObject}}

El cuerpo de la petición que hay que enviar en la llamada es:

```
{
  “organizationalUnit”: “UnidadOrganizativa”
}
```

Las validaciones que aplica son las mismas que desde el portal, no se puede cambiar la ou de una entidad que no existe o de aquellas que no tengan ou por sí mismas como INSTANCE.

Cambiar el estado de una entidad

Permite cambiar el estado de una entidad a cualquier estado, sin aplicar la lógica que suele tener dicho estado (ex: cambiarlo a expired cambiaría solo la entidad, no sus relaciones, no involucraría a tot si es necesario, etc).

Consiste en la llamada:

PUT http://{{host}}:{{port}}/gateway/api/admin/kerno/entity/change-state/{idObject}/{state}

En caso de cambiar el estado de DATASET, se actualizarán sus DATASET FIELD con el mismo estado.

IMPORTANTE: Los cambios de estado a APPROVED usando este endpoint no involucran a Tot de ninguna manera, por lo que pasar algún objeto gobernado de estado DRAFT a APPROVED por esta vía no involucraría a ningún sistema externo.

Cambiar el estado de una relación

Permite cambiar el estado de una relación a cualquier estado permitido para éstas, sin aplicar la lógica que tiene la actualización de una relación entera. No se permitirá el cambio de estado de una relación nativa.

Consisten en la llamada:

PUT
http://{{host}}:{{port}}/gateway/api/admin/kerno/relationship/change-state/{idObject}/{state}

donde **idObject** es el id de la relación a modificar y **state** el estado al que se desea cambiar.

IMPORTANTE: Los cambios de estado a APPROVED desde este endpoint no involucran a Tot de ninguna manera, por lo que pasar algún objeto gobernado de estado DRAFT a APPROVED por esta vía no involucraría a ningún sistema externo.

Cambiar el nombre de una entidad o relación

Realiza un cambio de nombre en la entidad o relación especificada en el cuerpo de la petición.

Antes de cambiar el nombre se aplicarán las validaciones de PKs correspondientes al subtipo de la entidad o relación que se esté intentando modificar. Por tanto, si se introduce un nombre que ya existe para otra (o la misma) entidad, el usuario recibirá una excepción en la respuesta de la petición.

Estructura de la llamada:

PUT http://{{host}}:{{port}}/gateway/api/admin/kerno/name/{objectType}

Cuerpo de la petición:

```
{
    "objectId": 1,
    "name": "relacion_1"
}
```

Parámetros:

- **objectType:** tipo del objeto. Valores permitidos: ENTITY o RELATIONSHIP.
- **objectId:** identificador interno del objeto.
- **name:** nombre que se le quiere aplicar a la entidad o relación.

Observaciones a tener en cuenta:

- Para modificación del nombre de un DSA aprobado que contenga entidades gobernadas conviene revisar la documentación del plugin correspondiente.
- Los workflows que hayan sido lanzados antes de este cambio de nombre tendrán el nombre que tenía la entidad o relación en el momento en el que se crearon.
- En las acciones que se hayan guardado en el histórico y/o auditoría anteriores al cambio tendrán el nombre que tenía la entidad o relación en el momento de la acción.
- No se permite sobreescribir el nombre de objetos en estado DRAFT, PENDING y REJECTED, ni en objetos APPROVED que tengan una versión DRAFT o PENDING

Obtener atributos que dependen de otros atributos

Indica, para el valor de un atributo dado, qué valores de atributos dependen del indicado. Se puede aplicar para todos los subtipos o la relación que aplique a uno solo.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/relationship

Cuerpo de la petición:

```
{
    "nameAttributeSource": "infrastructure",
    "nameAttributeDest": "technology",
```

```
    "valueAttributeSource": ["CPD Environment 1", "CPD Environment
    2"],
    "objectSubType": "DATASET"
}
```

Descripción de las variables, todas son obligatorias:

- **nameAttributeSource** Nombre del atributo origen
- **nameAttributeDest** Nombre del attributo destino
- **valueAttributeSource** Lista de valores de origen de los cuáles queremos saber sus relacionados
- **objectSubType** Subtipo de la plantilla en la cual están definidos los atributos

Creación de entidades nativas

En este apartado se exponen los endpoints preparados para poder crear entidades nativas (DATASET, DSA, INSTANCE, SOLUTION y PROCESS) con todo lo que conllevan (relaciones internas, entidades extra en caso de dataset, etc).

Dataset y dataset-fields

Permite crear un dataset y sus dataset-fields asociados finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/dataset

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato

Por lo que creará la entidad DATASET, tantos DATASET_FIELD como se incluyan en la petición con las relaciones internas entre dataset y sus datasetfield.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el dataset, en caso contrario se informará de los errores.

En ningún caso se permitirá modificar las PK o claves primarias.

Si el dataset es gobernado y su estado es APPROVED se invocará a tot.

Dsa

Permite crear un DSA y asociar las entidades que se quiera finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/dsa

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato.

Por lo que creará la entidad DSA y las relaciones internas entre dsa y sus entidades asociadas.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el DSA, en caso contrario se informará de los errores.

Cuando el estado que se elija sea APPROVED, se invocará a Tot.

Solución

Permite crear una solución y asociar las instancias relacionadas que se quieran, finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/solution

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato.

Por lo que creará la entidad SOLUTION y las relaciones internas entre la solución y sus instancias asociadas.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará la solución, en caso contrario se informará de los errores.

Instancia

Permite crear una instancia con sus dataset input/output asociados, finalizando en el estado que se incluya en la petición.

Consiste en la llamada

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/instance

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato.

Por lo que creará la entidad INSTANCE y las relaciones internas entre la instancia y sus dataset input y output.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará la instancia, en caso contrario se informará de los errores.

Proceso

Permite crear un proceso, finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{host}:{port}/gateway/api/admin/kerno/create-update/entity/process

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato.

Por lo que creará la entidad PROCESS.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el proceso, en caso contrario se informará de los errores.

Edición de entidades nativas

En este apartado se exponen los endpoints preparados para poder editar entidades nativas (DATASET, DSA, INSTANCE, SOLUTION) con todo lo que conllevan (relaciones internas, entidades extra en caso de dataset, etc).

En todas estas ediciones, si hay configuración que indica que los cambios realizados versionan el objeto, se generará una nueva versión de la entidad con los cambios correspondientes y deprecará la que se mandó editar.

Además, se permite editar campos no editables mediante el envío del parámetro de entrada: disableNonEditValidations. Está desactivado por defecto.

Dataset y dataset-fields

Permite editar un dataset y sus dataset-fields asociados finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/dataset/{idObject}

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato. Hay que informar la variable **idObject** en la URL.

Si no se quiere editar los dataset field, enviar la lista a null. Una lista vacía se interpreta como eliminar todos los dataset field.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el dataset, en caso contrario se informará de los errores.

Si el estado de la edición es APPROVED y el dataset es gobernado se invocará a tot.

Dsa

Permite editar un DSA y cambiar las entidades contenidas finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/dsa/{idObject}

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato. Hay que informar la variable **idObject** en la URL.

Si no se quiere editar las entidades contenidas, enviar la lista a null. Una lista vacía se interpreta como que no tiene entidades contenidas.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el DSA, en caso contrario se informará de los errores.

Cuando el estado que se elija sea APPROVED, se invocará a Tot con la información de la edición realizada.

Solución

Permite editar una solución y sus instancias relacionadas, finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST
http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/solution/{idObject}

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato. Hay que informar la variable **idObject** en la URL.

Si no se quiere editar las instancias relacionadas, enviar la lista a null. Una lista vacía se interpreta como que no tiene instancias relacionadas.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará la solución, en caso contrario se informará de los errores.

Instancia

Permite modificar una instancia con sus dataset input/output asociados, finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST
http://{{host}}:{port}/gateway/api/admin/kerno/create-update/entity/instance/{idObject}

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato. Hay que informar la variable **idObject** en la URL.

Si no se quiere editar los dataset input relacionados, enviar la lista a null. Una lista vacía se interpreta como que no tiene dataset input. Lo mismo aplica para los dataset output.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará la instancia, en caso contrario se informará de los errores.

Proceso

Permite editar un proceso, finalizando en el estado que se incluya en la petición.

Consiste en la llamada:

POST http://{host}:{port}/gateway/api/admin/kerno/create-update/entity/process/{idObject}

El cuerpo de la petición a enviar se obtiene como se indica en el apartado Inserción/Actualización de Metadato. Hay que informar la variable **idObject** en la URL.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará el proceso, en caso contrario se informará de los errores.

Edición masiva de objetos

Permite editar los mismos atributos con los mismos valores a una lista de objetos..

Consiste en la llamada:

PUT
http://{host}:{port}/gateway/api/admin/kerno/attribute/update/{objectType}/{objectSubType}

Como opción se incluye otro parámetro en la llamada: validateType. Este parámetro indica si se quiere validar el tipo de los parámetros (que los números son números o que el valor sea válido en aquellos campos con valores limitados, por ejemplo), por defecto está a true y solo se debe incluir si se quiere deshabilitar esas validaciones.

Como resultado se recibirá un OK (200) si todo ha ido bien o un PARTIAL CONTENT (206) si algún objeto no pudo ser editado, indicando el error que sucedió (será una lista con tantos elementos como objetos fallaron al editar).

Desadherencia de un DSA

Permite ejecutar la desadherencia de un DSA para el usuario indicado en la petición.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/kerno/disadhere/{idObject}

Los únicos usuarios que no se permite desadherir son los owners de la unidad organizativa del DSA.

Indexado de todas las entidades

Permite actualizar completamente la colección de SolR de los objetos de Anjana (solo la de kerno, no los snapshots), eliminando todos los objetos presentes y reindexando.

Consiste en la llamada:

PUT http://{{host}}:{port}/gateway/api/admin/kerno/index/all-entities

Este endpoint lanza en segundo plano el procesado de todos los objetos y su indexado. Dependiendo del tamaño de los mismos y las configuraciones del servidor, se puede demorar el proceso.

Internamente se realiza por bloques configurables. Si el proceso no indexa correctamente todos los elementos, revisar y ajustar la configuración de los bloques para que se adapte a lo que el sistema soporte.

Notificaciones

Todos se encuentran en el módulo de Hermes.

Obtener todas las notificaciones enviadas

Permite obtener todas las notificaciones enviadas según los criterios de búsqueda informados, siempre y cuando el usuario que realiza la petición es receptor de ellas (ya sea por nombre o por su rol y unidad organizativa)

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/hermes/notification

El cuerpo de la petición es el siguiente:

```
{
    "objectName": "dataset api admin 1",
    "objectType": ["ENTITY"],
    "objectSubType": ["DATASET"],
    "read": false,
    "notificationType": ["NOTICE"],
    "creationRangeStart": "2023-11-29T14:52:00.000Z",
    "creationRangeEnd": "2023-11-29T15:52:00.000Z",
    "page": 0,
    "size": 100,
    "order": "ASC",
    "sortBy": ["creationDate"]
}
```

Es obligatorio filtrar por al menos una variable, en caso contrario recibiremos un error 400 bad request.

En ese endpoint es recomendable incluir la cabecera de “x-language”, ya que las notificaciones siempre contienen valores traducidos.

Enviar una notificación

Permite enviar una notificación a un usuario o cualquier rol con los parámetros que se quiera.

Se realiza la llamada:

POST http://{{host}}:{{port}}/gateway/api/admin/hermes/send

El cuerpo de la petición es el siguiente:

```
{
    "senderUser": "maria.gonzalez",
    "moduleType": "DC",
    "notificationReceiverType": "USER",
    "notificationType": "NOTICE",
    "read": false,
    "severity": "HIGH",
    "typeObject": "ENTITY",
    "subtypeObject": "DATASET",
    "idObject": 2247,
    "redirectionType": "OBJECT",
    "objectName": "",
    "objectVersion": "0",
    "organizationalUnit": "SPA/Finance",
    "idNotification": 7,
    "execution": null,
    "task": "",
    "creationDate": "2023-11-30T17:09:00.000Z",
    "endDate": null,
    "receiverUser": "maria.gonzalez",
    "receiverRole": null,
    "messageKey": "NOTIFICATION.7.DC.USER",
    "messageParams": null
}
```

A continuación, se explica cada variable:

- **senderUser** Usuario que envía la notificación
- **moduleType** Módulo de la notificación. Valores posibles: ALL, DC y BG

- **notificationReceiverType** Tipo de receptor. Valores posibles: USER y ROLE
- **notificationType** Tipo de notificación. Valores posibles: NOTICE, ALERT y ADMIN_ALERT.
- **read** Indica si la notificación ha sido leída
- **severity** Importancia de la notificación. Valores posibles: LOW, MEDIUM y HIGH
- **typeObject** Tipo de objeto. Valores posibles: ENTITY y RELATIONSHIP
- **subtypeObject** Subtipo de objeto. Ej: DATASET, DSA, INSTANCE… o subtipos propios del cliente definidos en la tabla ObjectSubType
- **idObject** Id interno del objeto
- **redirectionType** tipo de redirección de la notificación
- **objectName** Nombre del objeto
- **objectVersion** Versión del objeto
- **organizationalUnit** Unidad Organizativa
- **idNotification** Identificador de la notificación definida en la tabla Notification.
- **execution** Identificador del workflow en caso de ser una notificación asociada a una ejecución.
- **task** Nombre de la tarea relacionada con la notificación
- **creationDate** Fecha de creación de la notificación
- **endDate** Fecha de fin de la notificación
- **receiverUser** Usuario que recibe la notificación en caso de ser una notificación cuyo notificationReceiverType sea USER
- **receiverRole** Usuario que recibe la notificación en caso de ser una notificación cuyo notificationReceiverType sea ROLE
- **messageKey** Clave de traducción del mensaje de la notificación
- **messageParams** Mapa de variables de la notificación que se sustituirán por su valor correspondiente en la traducción del messageKey.

Workflows

Todos se encuentran en el módulo de Hermes excepto el lanzamiento del paso final de workflow.

Obtener todos los workflows

Permite obtener el resumen de todos los workflows según los criterios de búsqueda informados. Si no se desea filtrar por alguno de estos criterios se tiene que indicar con un null.

Permite obtener todos los workflows lanzados en el sistema. Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/admin/hermes/executions

Obtener un workflow en particular

Permite obtener la información relativa a un workflow en particular incluyendo la información de los distintos pasos que forman el workflow, quien validó cada uno, su estado, etc.

Consiste en la llamada:

GET http://{{host}}:{port}/gateway/api/admin/hermes/execution/{id}

Borrado de un workflow en particular

Este endpoint permite borrar un workflow en particular y toda la información relacionada con él tanto en el motor de workflows como en Anjana.

DELETE http://{{host}}:{{port}}/gateway /api/admin/hermes/delete/workflow/{id}

Lanzar el paso final de un workflow

Este servicio permite lanzar el último paso de cualquier workflow del mismo modo que sucede al finalizar un workflow en Activiti, principalmente pensado para los casos que se produce alguna disincronía y no se finaliza los últimos procesos en los objetos.

Es la siguiente llamada:

PUT http://{{host}}:{port}/gateway/api/admin/kerno/workflow/finish/{workflowExecutionId}

Unidades Organizativas

Todos se encuentran en el módulo de Zeus.

Listado de Unidades Organizativas con permiso de creación/modificación

Permite obtener el listado de Unidades Organizativas sobre las que el usuario que se consulta tiene permiso de creación o modificación para un determinado subtipo de objeto.

Consiste en la llamada:

POST http://{{host}}:{{port}}/gateway/api/admin/zeus/list/{{objectSubtype}}

Recopila información de todos los proveedores configurados, por lo que es posible que haya un retraso en la obtención de respuesta por los retrasos que puedan existir con los proveedores.

Obtener los roles y unidades organizativas de un usuario

Permite obtener los roles y unidades organizativas del usuario que se consulta.

Consiste en la llamada:

POST http://{{host}}:{{port}}/gateway/api/admin/zeus/organizational-unit/role

Recopila información de todos los proveedores configurados, por lo que es posible que haya un retraso en la obtención de respuesta por los retrasos que puedan existir con los proveedores.

Creación de Unidades Organizativas

Permite crear una nueva unidad organizativa.

POST http://{{host}}:{{port}}/gateway/api/admin/zeus/create/organizational-unit

Borrado de Unidades Organizativas

Permite borrar una unidad organizativa a partir del identificador que tiene asignado en la base de datos.

DELETE http://{{host}}:{{port}}/gateway/api/admin/zeus/delete/organizational-unit/{id}

Actualización de Unidades Organizativas

Permite cambiar alguna propiedad de la unidad organizativa indicada en los datos provistos a partir del identificador que tiene en la base de datos

PATCH http://{{host}}:{{port}}/gateway/api/admin/zeus/update/organizational-unit

Indexador

Todos se encuentran en el módulo de minerva.

Indexar un objeto

Permite indexar un objeto (ENTIDAD o RELACIÓN) en particular para la colección KERNO y automáticamente siempre generará un SNAPSHOT.

Consiste en la llamada:

POST http://{{host}}:{{port}}/gateway/api/admin/minerva/index

Obtener todos los documentos

Permite obtener todos los documentos indexados en la colección de Kerno de Anjana.

Consiste en la llamada:

GET http://{{host}}:{{port}}/gateway/api/admin/minerva/kerno/getAll

En ese endpoint es muy recomendable incluir la cabecera de “x-language”, ya que los objetos siempre contienen valores traducidos y valores multilenguaje. Si no se incluye vendrán en el idioma por defecto de la aplicación.

Si, por el contrario, se especifica un idioma que no existe en el sistema los campos internacionales no aparecerán en los datos devueltos.

Para obtener todos los documentos correctamente es necesario realizar la llamada mediante un comando curl indicando un fichero donde guardar la respuesta. Ejemplo:

curl -X GET ^ http://dev44.anjanadata.org/gateway/api/admin/minerva/kerno/getAll \

-H "accept: */*" \

-H "Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwcnVlYmEucHJ1ZWJhIiwiYXVkIjoid2ViIiwicm9sZXMiOiJST0xFX0FTU09DSUFURURfQlVTU5FU1NfUFJPQ0VTU0VTX0NIQU5HRV9TVE…." \

 -o C:\Users\user\Escritorio\file.json

Obtener todos los snapshots de un objeto

Permite obtener todos los snapshots de un objeto.

Consiste en la llamada:

GET http://{{host}}:{{port}}/gateway/api/admin/minerva/snapshots/getAll/{objectSubType}/{idObject}

En ese endpoint es muy recomendable incluir la cabecera de “x-language”, ya que los objetos siempre contienen valores traducidos y valores multilenguaje. Si no se incluye vendrán en el idioma por defecto de la aplicación.

Si, por el contrario, se especifica un idioma que no existe en el sistema los campos internacionales no aparecerán en los datos devueltos.

Indexación de una auditoría

Permite crear la auditoría que se le indique en los datos provistos.

Consiste en la llamada:

POST http://{{host}}:{{port}}/gateway/api/admin/minerva/audit/save

Obtener toda la auditoría existente

Este endpoint permite obtener toda la auditoría existente en Anjana.

GET http://{{host}}:{{port}}/gateway/api/admin/minerva/audit/getAll

Obtener la auditoría de un usuario

Este endpoint permitirá obtener la auditoría existente para el usuario indicado.

GET http://{{host}}:{{port}}/gateway/api/admin/minerva/audit/getAll/{username}

Obtener toda la auditoría de un objeto dado

Este endpoint permitirá obtener toda la auditoría del objeto indicado sobre el subtipo provisto. La información devuelta será aquella que concuerda exactamente con lo provisto, es decir, si no se indica el id de objeto se devolverá aquella auditoría que tenga el subtipo indicado y como identificador del objeto vacío (null).

GET
http://{{host}}:{{port}}/gateway/api/admin/minerva/audit/getAll/{objectSubType}/{objectId}

Creación de la configuración de un filtro

Este endpoint permitirá crear los distintos filtros a utilizar en Anjana según los datos provistos

POST http://{{host}}:{{port}}/gateway/api/admin/minerva/create/filterConf

Actualización de la configuración de un filtro

Este endpoint va a permitir actualizar el filtro indicado con la información provista.

PATCH http://{{host}}:{{port}}/gateway/api/admin/minerva/update/filterConf

Actualizado de collection

Este endpoint realiza un actualizado de la colección. Esto significa que primero la borrará, luego creará los campos (tanto los básicos de la colección como los que se encuentren en la tabla attribute_definition de anjana).

Este endpoint se puede usar para cualquier colección de Anjana.

El endpoint es:

PUT http://{{host}}:{{port}}/gateway/api/admin/minerva/collection/update

Borrado de un campo de una colección

Para el borrado de un campo es necesario borrar todos los datos de la colección, por lo que primero borramos todos los datos y después se borra el campo, por lo tanto, si se quiere volver a tener la colección con los datos cargados hay que llamar el endpoint de indexAll.

El endpoint es:

DELETE
http://{{host}}:{{port}}/gateway/api/admin/minerva/collection/delete-field/STATE/KERNO

Api Pública

Inserción/Actualización de Metadato

Creación de Entidad

Por la complejidad de las entidades nativas, la creación de entidad no permite editar nada salvo los campos primarios y claves de las entidades, para completar todo el metadato es necesario posteriormente utilizar el endpoint de edición de entidad.

La llamada es:

POST http://{{host}}:{{port}}/gateway/api/v4/entity/create/{objectSubType}

Este endpoint genera el esqueleto de una entidad rellenando solo los valores fundamentales (nombre, ou, pks) incluyendo las relaciones internas que se requieran (ex: en el caso de instancia, con un proceso y una solución).

Creación de Relación

Para crear los datos de entrada para la inserción de metadato, es necesario conocer qué atributos tiene la plantilla de la relación que se quiere crear, para ello tomar como referencia la respuesta del servicio del formulario dinámico. Este endpoint es una llamada que se puede encontrar en:

GET http://{{host}}:{{port}}/gateway/api/v4/catalog/relationship/{objectSubType}

Estos endpoints devuelven la estructura del formulario dinámico con los campos que tiene la relación y las validaciones de los mismos.

Para la creación del objeto es necesario enviar una lista de atributos, al ser una creación es necesario incluir el nombre como atributos del objeto.

La lista de atributos se envía llamada:

POST http://{{host}}:{{port}}/gateway/api/v2/relationship/create/{objectSubType}

La relación creada siempre será en estado DRAFT.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se creará la relación.

Edición de entidad

Para crear los datos de entrada para la modificación de metadato, se puede tomar como referencia la respuesta del servicio del formulario dinámico. Este endpoint es una llamada que se puede encontrar en:

GET http://{{host}}:{{port}}/gateway/api/v4/catalog/entity/{objectSubType}

Este endpoint devuelve la estructura del formulario dinámico con sus validaciones.

Para consultar la lista completa de validaciones, incluyendo las de only-on-edition consultar el endpoint siguiente:

GET

http://{{host}}:{{port}}/gateway/api/v4/catalog/complete/entity/{objectSubType}

Para consultar la estructura de una entidad en concreta con sus campos y validaciones consultar:

GET

http://{{host}}:{{port}}/gateway/api/v4/catalog/entity/{objectSubType}/{idObject}

Para la modificación de la entidad es necesario enviar una lista de atributos.

La lista de atributos se envía con la siguiente llamada:

POST http://{{host}}:{{port}}/gateway/api/v2/entity/save/{objectSubType}/{idObject}

Esta edición modificará los valores previos en caso de editar un DRAFT, pero en caso de editar un APPROVED, DEPRECATED o EXPIRED, creará un DRAFT nuevo con nuevo ID (clonando todo lo necesario, por ejemplo datasetfield en caso de DATASET o las relaciones con dataset en el caso de un DSA) dejando el objeto original sin tocar.

Al igual que desde el portal, los datos enviados serán validados y, solo si se pasan todas las validaciones, se editará la entidad.

Edición de relación

Para crear los datos de entrada para la modificación de metadato, se puede tomar como referencia la respuesta del servicio del formulario dinámico. Este endpoint es una llamada que se puede encontrar en:

GET http://{{host}}:{{port}}/gateway/api/v4/catalog/relationship/{objectSubType}/{idObject}

Estos endpoints devuelven la estructura del formulario dinámico con los campos tiene la relación y sus validaciones.

Para la modificación de la relación es necesario enviar una lista de atributos.

La lista de atributos se envía llamada:

POST http://{{host}}:{{port}}/gateway/api/v2/relationship/save/{objectSubType}/{idObject}

Esta edición editará los valores previos en caso de editar un DRAFT, pero en caso de editar un APPROVED, DEPRECATED o EXPIRED, creará un DRAFT nuevo con nuevo ID dejando el objeto original sin tocar.

Al igual que desde el portal, los datos enviados serán validados, y solo si se pasan todas las validaciones se editará la relación.

Obtener todas las relaciones de una entidad

Permite obtener todas las relaciones que tiene una entidad incluyendo las relaciones internas.

Consiste en la siguiente llamada:

GET http://{{host}}:{port}/gateway/api/v2/relationship/all/{objectSubtype}/{id}

Obtener los valores definidos para un atributo

Permite obtener todos los valores posibles de un atributo con valores predefinidos de tipo SELECT.

Consiste en la llamada:

POST http://{{host}}:{port}/gateway/api/v2/attribute/values

El cuerpo de la petición es el siguiente:

```
{
   "attributeName": "dominioraiz"
}
```

Búsqueda filtrada

Permite hacer búsquedas en SolR con diferentes filtros.

Consiste en una llamada **POST** http://{{host}}:{port}/gateway/api/v1/indexer/{target}/search incluyendo en el body los parámetros de búsqueda.

En ese endpoint es muy recomendable incluir la cabecera de “x-language”, ya que los objetos siempre contienen valores traducidos y valores multilenguaje. Si no se incluye vendrán en el idioma por defecto de la aplicación.

Validación de workflows

Permite validar uno o más workflows a la vez mediante sus identificadores

Consiste en la llamada **PUT** http://{{host}}:{port}/gateway/api/v1/task/validate

Envío masivo

Permite enviar a validar múltiples objetos del mismo subtipo.

Consiste en la llamada **POST** http://{{host}}:{{port}}/api/v2/common/massiveSubmit/{{objectSubtype}} incluyendo en el body los ids de los objetos y el role con el que se lanzarán los workflows.

Como respuesta llegarán aquellos ids que han completado la operación con éxito y aquellos que no junto con el porqué.

En ese endpoint es muy recomendable incluir la cabecera de “x-language”, ya que los objetos siempre contienen valores traducidos y valores multilenguaje. Si no se incluye vendrán en el idioma por defecto de la aplicación.

Como ayuda para evitar tener muchos errores se incluye otro endpoint que recibiendo una lista de ids, se devuelve aquellos que cumplen las condiciones para poder ser enviados, dicho endpoint es un **POST** http://{{host}}:{{port}}/api/v2/common/checkSubmit con la lista de ids de los objetos y su subtipo.

Códigos de respuesta de la API según el estado de la licencia

Código	Estado	Descripción
0000	VALID	La licencia es válida y el usuario puede acceder a Anjana correctamente.
99997	EXPIRING	Se trata de un estado de preaviso de que la licencia ya no es válida desde hace más de 3 días hasta 7 días
99998	EXPIRED	La licencia no es válida desde hace más de 7 días, no se puede acceder a la aplicación
99996	TEMPORARY	Licencia temporal. La licencia es válida y el usuario puede acceder a Anjana correctamente.
99995	READ_ONLY	Licencia que indica que solo podrá acceder a la aplicación en modo lectura