Documentación Microservicios Modificación Extra NA ÚNICAS

Implantación NA UNICAS



Minsait

Octubre 2026

Histórico de versiones:

Versión	Realizado por:	Fecha
0.0	Minsait	01/10/2026

Contenido

Documentación Microservicios Modificación Extra NA ÚNICAS 1

1 INTRODUCCIÓN 4

2 Microservicio careteam-mpi-creator 6
2.1 ENDPOINTS DISPONIBLES 7
2.2 ACTUALIZACIÓN HELM CHART 7
2.2.1 ACTUALIZACIÓN 7
2.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN 8
2.2.3 DESINSTALACIÓN 9

3 Microservicio practitioner-sync 10
3.1 ENDPOINTS DISPONIBLES 10
3.2 ACTUALIZACIÓN HELM CHART 13
3.2.1 ACTUALIZACIÓN 14
3.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN 14
3.2.3 DESINSTALACIÓN 15

4 Microservicio patient-proxy 17
4.1 ENDPOINTS DISPONIBLES 17
4.2 ACTUALIZACIÓN HELM CHART 22
4.2.1 ACTUALIZACIÓN 22
4.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN 23
4.2.3 DESINSTALACIÓN 24

5 Microservicio qr-to-condition 25
5.1 ENDPOINTS DISPONIBLES 25
5.2 ACTUALIZACIÓN HELM CHART 34
5.2.1 ACTUALIZACIÓN 34
5.2.2 VERIFICACIÓN DE LA INSTALACIÓN 35
5.2.3 DESINSTALACIÓN 36

6 Microservicio procesar-notificaciones-nc 37
6.1 ENDPOINTS DISPONIBLES 38
6.2 ACTUALIZACIÓN HELM CHART 38
6.2.1 ACTUALIZACIÓN 39
6.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN 40
6.2.3 DESINSTALACIÓN 41

7 Microservicio obtener-info-paciente 43
7.1 ENDPOINTS DISPONIBLES 43
7.2 ACTUALIZACIÓN HELM CHART 45
7.2.1 ACTUALIZACIÓN 45
7.2.2 VERIFICACIÓN DE LA INSTALACIÓN 46
7.2.3 DESINSTALACIÓN 46

8 Microservicio download-binary 48
8.1 ENDPOINTS DISPONIBLES 48
8.2 INSTALACIÓN HELM CHART 49
8.2.1 INSTALACIÓN 49
8.2.2 VERIFICACIÓN DE LA INSTALACIÓN 51
8.2.3 ACTUALIZACIÓN 51
8.2.4 DESINSTALACIÓN 52

9 Microservicio inject-momento-encounter 53
9.1 ENDPOINTS DISPONIBLES 53
9.2 ACTUALIZACIÓN HELM CHART 53
9.2.1 ACTUALIZACIÓN 54
9.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN 54
9.2.3 DESINSTALACIÓN 55

1 INTRODUCCIÓN

Este documento contiene información básica sobre los microservicios generados para integrar las aplicaciones del Nodo Autonómico de ÚNICAS entre sí, con el Nodo Central y con los HIS asociados al nodo autonómico.

Los pasos de este manual, se tiene que aplicar después de tener instalada la última versión entregada de la documentación, tanto de instalación, microservicios, configuración y kafka.

Las integraciones están basadas en Camel K, ya que permite crear flujos de integración construidos con Apache Camel y ejecutados de forma cloud-native sobre Kubernetes/Openshift, donde Camel K se encarga de compilar, desplegar y operar automáticamente la integración. De esta forma se ejecuta como un recurso nativo del clúster (*Integration*), facilitando su escalado, configuración y conectividad con otros servicios del mismo entorno.

También contiene las instrucciones para actualizar o desinstalar los microservicios modificados en esta versión utilizando la tecnología HELM. Esto nos permite empaquetar las integraciones, distribuirlas y desplegarlas sobre Kubernetes/Openshift, versionarlas y parametrizar su configuración, estandarizando así el despliegue de las integraciones en distintos entornos de forma repetible y controlada.

Los microservicios añadidos que deben instalarse son:

- download-binary.

Los microservicios modificados que deben actualizarse son:

- careteam-mpi-creator:
  - Añadido LoggerManager para poder cambiar dinámicamente el nivel de log.
  - Cabecera log con información de versionado
- practitioner-sync:
  - Añadido LoggerManager para poder cambiar dinámicamente el nivel de log.
  - Cabecera log con información de versionado
- patient-proxy:
  - Blindar la ausencia de name.family en alta y modcore para evitar problemas de busquedas por apellidos
  - nuevo logging-manager
  - Cabecera info version servicio
  - Correcciones menores
- qr-to-condition:
  - Nueva gestión de logs mediante logging-manager.
  - Cabecera con info de versión de servicio
  - En la notificacion de alta paciente si NC devuelve que ya esta registrado el CIPSNS se procede a notificar una modificacion core para pasar OK por NC e informar a los demás NA del alta
- procesar-notificaciones-nc:
  - Solo para esta Alpa V7 hasta que se distribuya la beta/release: se consumen las notificaciones NOTIF_SINCRO_DOCUMENTAL para que no se acumulen en NC
  - Paginación en el getAll de notificaciones.
  - Se omite el uso del Kafka para atender las notificaciones del GetAll horario para asegurar un consumo ordenado. Se conservan los topics y el consumer para un reprocesado en caso que alguna de las notificaciones dé error
  - Nuevo LoggerManager
  - Cabecera log con información de versionado
  - En una ModCore si el paciente base no existe en el MPI local se crea a partir de los datos core rebcibido.

- obtener-info-patient:
    - Nuevo logging-manager para mejorar la gestión de logs
    - Cabecera info versionado para identificar la versión en las respuestas
    - Se evita historiar los recursos importados mediante el header `x-hdr-disable-version`.
    - Se blinda el tratamiento de `DocumentReference` cuando el `attachment` venga en un formato diferente al de `Binary`.
- inject-momento-encounter:
    - Añadido LoggerManager para poder cambiar dinámicamente el nivel de log.
    - Cabecera log con información de versionado

A continuación, se procede a explicar cada uno de los microservicios modificados para esta versión y como actualizarlos.

2 Microservicio careteam-mpi-creator

Este microservicio que recibe la información necesaria para crear o actualizar un recurso FHIR "CareTeam". El recurso FHIR "CareTeam" se utiliza para representar las adscripciones médicas, organizativas, de cuidadores y responsables legales del paciente.

Este microservicio no expone endpoints REST: funciona por consumo de KAFKA.

Para crear el topic de kafka, en la instalación de la configuración, se han creado una serie de recursos FHIR Subscription, uno de ellos para ser utilizado por este microservicio. Esa suscripción detecta las Task que tienen la etiqueta correspondiente al enrolamiento del paciente en el proceso ÚNICAS y cuyo estado es completed. Cuando una Task cumple esas condiciones, la suscripción envía un aviso. Ese aviso no llega directamente a la integración, sino que se publica en un topic de Kafka llamado inject-patient-careteam-topic. Este topic contiene el contenido completo del recurso Task que ha generado el evento. El microservicio está suscrito a este topic, por lo que cada vez que la suscripción envía uno de estos avisos, el servicio recibe el Task y extrae de él la información necesaria para saber qué paciente debe procesarse, a qué unidad asistencial pertenece y qué organización está implicada. A partir de esos datos, decide si debe crear un nuevo CareTeam o actualizar uno ya existente. Si se crea uno nuevo, se hace un POST y si se actualiza se hace un PUT, ambos en MPI.

Una vez realizado el proceso, el microservicio devuelve un mensaje indicando si la operación ha sido correcta, si el registro ya existía o si se ha producido algún error. En caso de error, este se envía al siguiente topic de kafka "careteam-mpi-creator-topic-error" para tener un seguimiento.

También se incluye la clase Authenticacion para poder incluir Authorization Bearer **token** en la llamada a MPI.

Los datos necesarios que se utilizan para crear el CareTeam y que llegan a través del topic son:

- Identificador del paciente:
  - Procede de: payload.for.reference.
  - Se usa para identificar al paciente y consultar su información en el MPI.
- Fecha del proceso:
  - Procede de: payload.executionPeriod.end.
  - Se utiliza como period.start del CareTeam.
- Unidad asistencial del paciente:
  - Procede de: payload.contained[].PractitionerRole con id="pr-last-update" y extensión con url="ohbpm.auth.unit".
  - Se usa para definir la unidad participante en el CareTeam.
- Organización asociada:
  - Procede de: payload.contained[].PractitionerRole.organization, siempre dentro del PractitionerRole con id="pr-last-update".
  - Se usa para indicar la organización responsable en el CareTeam.
- Datos del profesional:
  - Procede del mismo PractitionerRole pr-last-update.

- Aunque el Task incluye información del profesional, solo se utiliza su rol para extraer unidad y organización.

2.1 ENDPOINTS DISPONIBLES

No aplica.

2.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **careteam-mpi-creator**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.
- La configuración externa.
- Los recursos adicionales necesarios, incluidas plantillas Qute de respuesta y creación del CareTeam.

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio careteam-mpi-creator-chart.

2.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión. Para ello es necesario sustituir el directorio "careteam-mpi-creator-chart" existente por el nuevo. Previamente se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se quiera actualizar la integración.

```yaml
integration:
  name: careteam-mpi-creator
  namespace: #namespace en el que se quiere desplegar la integración
  dependencies:
    - "camel:core"
    - "camel:direct"
    - "camel:caffeine"
    - "camel:jcache"
```

En este caso, el único valor que debemos cambiar es el "namespace", indicando en cuál de los disponibles queremos que se actualice.

Para efectuar la modificación, hay que ejecutar el siguiente comando: **helm upgrade careteam-mpi-creator ./careteam-mpi-creator-chart -n <namespace>** .

Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade careteam-mpi-creator ./careteam-mpi-creator-chart -n int
Release "careteam-mpi-creator" has been upgraded. Happy Helming!
NAME: careteam-mpi-creator
LAST DEPLOYED: Tue Mar  3 07:39:32 2026
NAMESPACE: int
STATUS: deployed
REVISION: 3
TEST SUITE: None
```

2.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio:

Al ejecutar **kubectl get integration careteam-mpi-creator -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration careteam-mpi-creator -n int
NAME                  PHASE     READY   RUNTIME PROVIDER   RUNTIME VERSION   CATALOG VERSION   KIT                          REPLICAS
careteam-mpi-creator  Running   True    plain-quarkus      3.30.3            3.30.3            kit-d5ss680u9db7hk4bf8eg     1
sh-4.2$
```

En este caso, que también se debe crear el configmap "careteam-mpi-creator-qute", se comprueba su existencia en el namespace con el siguiente comando: **kubectl get cm careteam-mpi-creator-qute -n <namespace>**

Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm careteam-mpi-creator-qute -n int
NAME                          DATA   AGE
careteam-mpi-creator-qute     2      3m41s
```

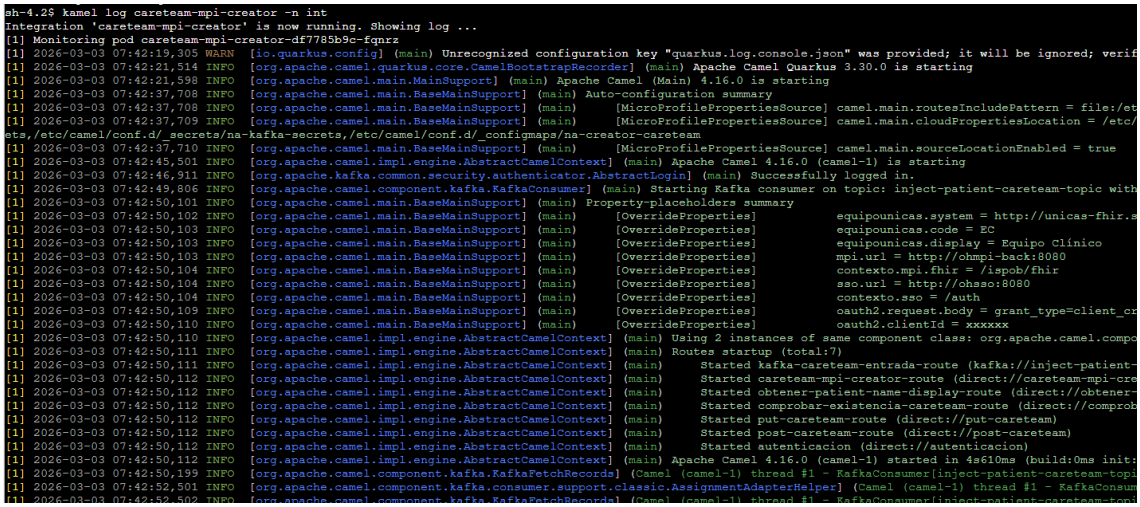
y el configmap "na-creator-careteam" : **kubectl get cm na-creator-careteam -n <namespace>**

Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm na-creator-careteam -n int
NAME                    DATA   AGE
na-creator-careteam     1      85d
```

2. Consultar logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel log careteam-mpi-creator -n <namespace>**. Ejemplo de esta integración en namespace "int":



2.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo, por un error en la definición de parámetros en values.yaml, se ejecuta el siguiente comando: **helm uninstall careteam-mpi-creator -n <namespace>.**

```
sh-4.2$ helm uninstall careteam-mpi-creator -n int
release "careteam-mpi-creator" uninstalled
```

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los ConfigMap definidos dentro del Helm chart "na-creator-careteam" y "careteam-mpi-creator-qute".

3 Microservicio practitioner-sync

Este microservicio recibe una petición con los datos de un profesional sanitario y su ámbito (SISTEMA o UNIDAD), y sincroniza dicha información entre los sistemas HDR (para la validación de la Unidad/Organización) y AUT (para búsqueda o creación del recurso FHIR Practitioner y PractitionerRole). El objetivo es garantizar que tanto el profesional (Practitioner) como su rol asociado (PractitionerRole) existen en el sistema AUT, y que la unidad indicada (cuando aplique) existe previamente en HDR.

Este servicio es invocado automáticamente desde el script "**review_user**" de Keycloak, encargado de evaluar los datos del usuario durante el proceso de autenticación.

Para determinar el ámbito del profesional se valida:

- Si los campos center_code, unit_code y careline_code existen y tienen contenido → Ámbito UNIDAD
- Si alguno de los anteriores falta o llegan vacíos → Ámbito SISTEMA

En función de lo anterior, se realizan las validaciones y operaciones siguientes:

- Validación de la UNIDAD (solo si aplica): Se comprueba en HDR que la unidad existe. Si no existe, se devuelve un error
- Búsqueda del Practitioner: Se consulta en AUT si el practitioner ya existe. Si existe, se reutiliza su ID, si no existe se crea mediante una operación POST. Para su profesión, buscamos si la profesión que llega por parámetros a la integración ya existe en el Practitioner, si no, se la añadimos a las existentes, siempre y cuando esta profesión exista en el CodeSystem 'profession'.
- Búsqueda de PractitionerRole: Se consulta en AUT si el rol asociado al Practitioner existe. Si no existe, se crea mediante una operación PUT

También se incluye la clase Authentication para poder incluir Authorization Bearer **token** en las llamadas a AUT y HDR.

3.1 ENDPOINTS DISPONIBLES

POST /practitionerSync

Realiza la sincronización completa del profesional y su rol en AUT, y valida la existencia de la unidad en HDR cuando corresponde.

Entrada

Authentication: Bearer token

Accept: application/json

Content-Type: application/json

Estructura cuerpo (los datos son ejemplos):

Para UNIDAD

```
{
  "iat": "1776701855",
  "exp": "1776712655",
  "iss": "VH",
  "name": "Nombre Apellido1 Apellido2",
```

```
  "given_name": "Nombre",
  "family_name": "Apellido1 Apellido2",
  "practitioner_document": "123456789Z",
  "sub": "081555789A",
  "center_code": "012345",
  "unit_code": "012345NEU",
  "role_code": "ACCESO_UNICAS_UNIDAD_PACIENTE",
  "profession_code": "DOC",
  "careline_code": "CEX",
  "smart": "ohhda-unicas",
  "compact": false,
  "patient": "urn:oid:2.16.724.4.40|BBBBBBBBMM950152",
  "email": "prueba@ejemplo.com"
}
```


Para SISTEMA


```
{
  "iat": "1776701855",
  "exp": "1776712655",
  "iss": "VH",
  "name": "Nombre Apellido1 Apellido2",
  "given_name": "Nombre",
  "family_name": "Apellido1 Apellido2",
  "practitioner_document": "123456789A",
  "sub": "081555789A",
  "center_code": "",
  "unit_code": "",
  "role_code": "ADMIN_UNICAS",
  "profession_code": "ADV",
  "careline_code": "",
  "smart": "ohhda-unicas",
  "compact": false,
  "patient": "urn:oid:2.16.724.4.40|BBBBBBBBMM950152",
  "email": "pruebaAdv@ejemplo.com"
}
```


ó


```
{
  "iat": "1776701855",
  "exp": "1776712655",
  "iss": "VH",
  "name": "Nombre Apellido1 Apellido2",
  "given_name": "Nombre",
  "family_name": "Apellido1 Apellido2",
  "practitioner_document": "123456789A",
  "sub": "081555789A",
  "role_code": "ADMIN_UNICAS",
  "profession_code": "ADV",
  "smart": "ohhda-unicas",
  "compact": false,
  "patient": "urn:oid:2.16.724.4.40|BBBBBBBBMM950152",
  "email": "pruebaAdv@ejemplo.com"
}
```

Campo	Obligatorio	Descripción
practitioner_document	Sí	DNI/NIF del profesional. Se usa como identifier para buscar/crear el Practitioner.
role_code	Sí	Indicar aquí el role con el que pretende acceder, por el momento los valores posibles son: • ACCESO_UNICAS_PACIENTE_UNIDAD (contexto Paciente) • ADMIN_UNICAS (sin contexto Paciente)
center_code	Sí en UNIDAD	Identificador del Centro (p. ej. 0908005177)
unit_code	Sí en UNIDAD	Identificador de la Unidad/Organization (p. ej. 0908005177NEU)
careline_code	Sí en UNIDAD (Valor fijo)	Valor fijo "CEX"
name	Sí	Nombre completo del profesional para el que se solicita el acceso, con formato <nombre> <primer apellido> {<segundo apellido>} (el segundo apellido es opcional)
given_name	Sí	Nombre del Profesional
family_name	Sí	Apellidos del Profesional (el segundo apellido es opcional)
email	Sí	Dirección de correo electrónico del Profesional (si no llega en el token, lo construimos con el patrón: <DNI>@example.com)
profession_code	Sí	Indicar aquí el código de profesión asociada al profesional, los valores posibles vendrán definidos en el CodeSystem “profession” que se encuentra en el NA UNICAS, a modo de ejemplo: • DOC (médico) • ADV (administrativo)

Las posibles combinaciones de Profesión/Rol/Scope son las siguientes:

Profesión	Role	Scope
OHBI	ROL_OHBI_ADMIN	SISTEMA
OHBI	ROL_OHBI_USER	SISTEMA
ADV	ADMIN_UNICAS	SISTEMA
DOC	ACCESO_UNICAS_PACIENTE	CENTRO
DOC	ACCESO_UNICAS_UNIDAD_PACIENTE	UNIDAD
DOC	GEST_CASO	UNIDAD
ADV	GEST_AUT	SISTEMA
DOC	GEST_HOSP	UNIDAD
DOC	CLI_REF	UNIDAD
TRA	TRA_SOCIAL	UNIDAD
DOC	ASIS	UNIDAD
ASIS	ASIS	UNIDAD

Salida

Estructura mensaje salida:

Si el código de respuesta es 200:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information",
      "code": "informational",
      "details": {
        "text": "Éxito en la operación"
      }
    }
  ]
}
```

Cualquier otro código de respuesta:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "Mensaje de excepción"
    }
  ],
  "source": {
    ... JSON original recibido ...
  }
}
```

Posibles respuestas:

Código	Descripción
200	Éxito en la operación.El Practitioner y PractitionerRole existen o han sido creados.
500	Cualquier excepción en validación HDR, creación FHIR o durante el flujo Camel. Incluye el JSON original en source.

3.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **practitioner-sync**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.
- La configuración externa.
- Los recursos adicionales necesarios, en este caso el configmap "practitioner-sync-qute"

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio practitioner-sync-chart.

3.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión. Para ello es necesario sustituir el directorio "practitioner-sync-chart" existente por el nuevo.

Previamente se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se quiera actualizar la integración.

```yaml
integration:
  name: practitioner-sync
  namespace: #namespace en el que se quiere desplegar la integración
  dependencies:
    - "camel:core"
    - "camel:direct"
    - "camel:rest"
    - "camel:http"
```

En este caso, el único valor que debemos cambiar es el "namespace", indicando en cuál de los disponibles queremos que se actualice.

Para efectuar la modificación, hay que ejecutar el siguiente comando: **helm upgrade practitioner-sync ./practitioner-sync-chart -n <namespace>.**

Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade practitioner-sync ./practitioner-sync-chart -n int
Release "practitioner-sync" has been upgraded. Happy Helming!
NAME: practitioner-sync
LAST DEPLOYED: Tue Mar  3 10:00:57 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

3.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio:

Al ejecutar **kubectl get integration practitioner-sync -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration practitioner-sync -n int
NAME                PHASE     READY   RUNTIME PROVIDER   RUNTIME VERSION   CATALOG VERSION   KIT                          REPLICAS
practitioner-sync   Running   True    plain-quarkus      3.30.6            3.30.6            kit-d62akb971qc4tpljvid0   1
sh-4.2$
```

En este caso, que también se crea el configmap "practitioner-sync-qute", se puede comprobar su existencia en el namespace con el siguiente comando: **kubectl get cm practitioner-sync-qute -n <namespace>.**

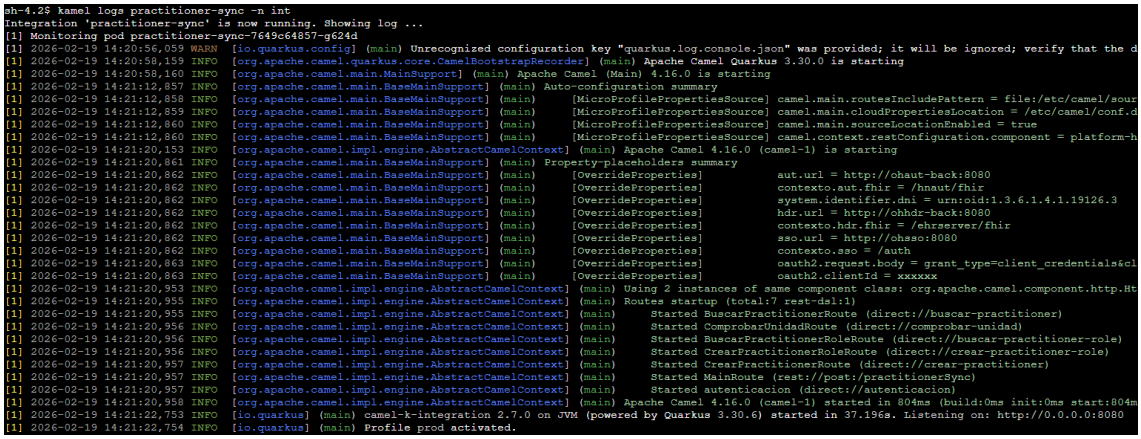
Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm practitioner-sync-qute -n int
NAME                      DATA   AGE
practitioner-sync-qute    2      20d
```

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs practitioner-sync -n <namespace>.**

Ejemplo de esta integración en "int":



3.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo, por un error en la definición de parámetros en values.yaml, se ejecuta el siguiente comando: **helm uninstall practitioner-sync -n <namespace>.**

```
sh-4.2$ helm uninstall practitioner-sync -n int
release "practitioner-sync" uninstalled
```

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.

- El ConfigMap definido dentro del Helm chart "practitioner-sync-qute".

4 Microservicio patient-proxy

Servicio que actúa como un proxy de operaciones contra el MPI.

Su objetivo es dar de alta o modificar pacientes en el MPI a partir de información recibida (formato ministerio), garantizando:

- Seguridad mediante Keycloak (Rol requerido: HNPOB_POB_WRITE).
- Validación del CipSNS proporcionado.
- Normalización del recurso FHIR Patient al modelo UNICAS.
- Alta/modificación del Patient en el MPI local.
- Notificación a NC cuando se producen alta o modificaciones relevantes del paciente (dato CORE o genérico).
- En caso de error se dispone de un topic en Kafka "patient-proxy-error" para aportar más información al respecto.

4.1 ENDPOINTS DISPONIBLES

PUT /ie/patientproxy?identifier=urn:oid:2.16.724.4.40|BBBBBBBBxxxxxxxx

Envío del patient para el alta o modificación de datos

Entrada

Authentication: Bearer token

Accept: application/json, application/*+json, **application/xml, application/*+xml**

Content-Type: application/json, **application/xml**

Estructura cuerpo: Patient Paciente ÚNICAS - ÚNICAS Rare Diseases HL7 FHIR Implementation Guide v0.0.4

Salida

La estructura mensaje salida en caso de resoltado 200 OK es un recurso Patient formateado debidamente para la aplicacion de ÚNICAS:


```
{
    "resourceType": "Patient",
    "id": "AC17714027431135712",
    "meta": {
        "versionId": "5",
        "lastUpdated": "2026-03-03T17:04:04.000+00:00",
        "profile": [
            "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/UNICASPatient",
            "http://hl7.org/fhir/4.0/StructureDefinition/Patient"
        ],
        "security": [
            {
```

```
                "system": "https://unicas-
fhir.sanidad.gob.es/CodeSystem/UNICASConfidenciality",
                "code": "AS-S",
                "display": "Asistencia sanitaria con investigación"
            }
        ],
        "tag": [
            {
                "system": "unicas.origen",
                "code": "ES-CT"
            }
        ]
    },
    "extension": [
        {
            "url": "http://hn.indra.es/fhir/StructureDefinition/patient-
confidential",
            "valueBoolean": false
        },
        {
            "url": "http://hn.indra.es/fhir/StructureDefinition/internal-
origin-identifier",
            "valueString": "OH_MPI"
        },
        {
            "url": "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/MomentosUnicas",
            "valueCodeableConcept": {
                "coding": [
                    {
                        "system": "https://unicas-
fhir.sanidad.gob.es/CodeSystem/CodigosMomentosUNICAS",
                        "code": "DA",
                        "display": "Diagnóstico-Asignación"
                    }
                ],
                "text": "Diagnóstico-Asignación"
            }
        },
        {
            "url": "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/NodoInscripcion",
            "valueCodeableConcept": {
                "coding": [
                    {
                        "system": "https://unicas-
fhir.sanidad.gob.es/CodeSystem/CodigosNodosAutonomicos",
                        "code": "ES-99",
                        "display": "Otros"
                    }
                ],
                "text": "Otros"
            }
        },
        {
            "url": "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/NodoAlta",
```

```
            "valueCodeableConcept": {
                "coding": [
                    {
                        "system": "https://unicas-
fhir.sanidad.gob.es/CodeSystem/CodigosNodosAutonomicos",
                        "code": "ES-CT",
                        "display": "Catalunya"
                    }
                ],
                "text": "Catalunya"
            }
        },
        {
            "url": "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/EstadoEnrolamientoPaciente",
            "extension": [
                {
                    "url": "estado-enrolamiento",
                    "valueBoolean": true
                }
            ]
        },
        {
            "url": "http://hl7.org/fhir/StructureDefinition/match-grade",
            "valueDecimal": 0.0
        }
    ],
    "identifier": [
        {
            "extension": [
                {
                    "url":
"http://hn.indra.es/fhir/StructureDefinition/identifier-guardian",
                    "valueBoolean": false
                }
            ],
            "use": "official",
            "type": {
                "coding": [
                    {
                        "system":
"http://snomed.info/sct/900000001000122104",
                        "code": "1571000122102",
                        "display": "código de identificación del paciente
en la comunidad autónoma"
                    }
                ]
            },
            "system": "urn:cite:80724000015",
            "value": "PECE0230404003"
        },
        {
            "extension": [
                {
                    "url":
"http://hn.indra.es/fhir/StructureDefinition/identifier-guardian",
                    "valueBoolean": false
```

```
                }
            ],
            "use": "official",
            "type": {
                "coding": [
                    {
                        "system":
"http://snomed.info/sct/900000001000122104",
                        "code": "1551000122105",
                        "display": "código de identificación personal en
el Sistema Nacional de Salud"
                    }
                ]
            },
            "system": "urn:oid:2.16.724.4.40",
            "value": "BBBBBBBBHH000848"
        },
        {
            "extension": [
                {
                    "url":
"http://hn.indra.es/fhir/StructureDefinition/identifier-guardian",
                    "valueBoolean": false
                }
            ],
            "use": "official",
            "type": {
                "coding": [
                    {
                        "code": "422549004"
                    }
                ],
                "text": "Identificador MPI"
            },
            "system": "urn:oid:1.1.1.2",
            "value": "AC17714027431135712"
        }
    ],
    "active": true,
    "name": [
        {
            "extension": [
                {
                    "url":
"http://hl7.org/fhir/StructureDefinition/humanname-fathers-family",
                    "valueString": "PEREZ"
                },
                {
                    "url":
"http://hl7.org/fhir/StructureDefinition/humanname-mothers-family",
                    "valueString": "CERCADO"
                }
            ],
            "use": "nickname",
            "text": "JOAQUIN PEREZ CERCADO",
            "family": "PEREZ CERCADO",
            "_family": {
```

```
                "extension": [
                    {
                        "url":
"http://hl7.org/fhir/StructureDefinition/humanname-fathers-family",
                        "valueString": "PEREZ"
                    },
                    {
                        "url":
"http://hl7.org/fhir/StructureDefinition/humanname-mothers-family",
                        "valueString": "CERCADO"
                    }
                ]
            },
            "given": [
                "JOAQUIN"
            ]
        }
    ],
    "gender": "male",
    "birthDate": "2023-04-04",
    "managingOrganization": {
        "reference": "MPI"
    }
}
```


Code block 1 Body de salida

En caso de error controlado se recibe un recurso OperationOutcome:


```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "El CipSNS del header no coincide con el CipSNS del body.
Consulte el topic 'patient-proxy-error' para más información."
    }
  ]
}
```


Posibles respuestas:

Código	Descripción
200	Alta/modificación del paciente realizada correctamente en el MPI local
400	Error en la petición realizada por el cliente
500	Error interno. Consultar topic "patient-proxy-error"

4.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **patient-proxy**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.
- La configuración externa.
- Los recursos adicionales necesarios (incluidas plantillas Qute, schemas,... si los hubiera).
- Configuracion del gateway.

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio.

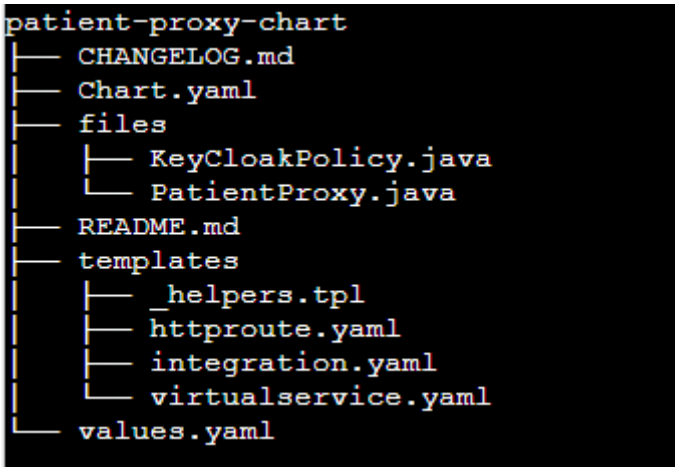
4.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión.

A continuación, se describe el procedimiento para la correcta instalación de la integración **patient-proxy** utilizando Helm.

1. Preparación del chart

Verificar que la estructura del directorio patient-proxy-chart es la siguiente:



2. Configuración Previa

Modificar el archivo **values.yaml**, para indicar el namespace en el que se encuentra la integración.

```
integration:
  name: patient-proxy
  namespace: #namespace en el que se quiere desplegar la integración
  runtime:
    provider: plain-quarkus
```

En este caso también tendremos que indicar el tipo de gateway que hay para que se expongan los endpoints de la integración: **k8_gateway** o **istio**.

```
gateway:
  type: # tipo de gateway. Valores: k8s_gateway/istio
  name: patient-proxy-gateway
  unicasbase: #url base del servidor. Por ejemplo: integracio.unicas.salut.intranet.gencat.cat
```

3. Ejecutar comando de instalación.

Sustituir el directorio "patient-proxy-chart" existente por el nuevo y ejecutar el siguiente comando: **helm upgrade patient-proxy ./patient-proxy-chart-chart -n <namespace>.** Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade patient-proxy ./patient-proxy-chart/ -n int
Release "patient-proxy" has been upgraded. Happy Helming!
NAME: patient-proxy
LAST DEPLOYED: Tue Mar  3 16:22:23 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

4.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN

Para confirmar que se ha creado correctamente la integración, se pueden ejecutar los siguientes comandos:

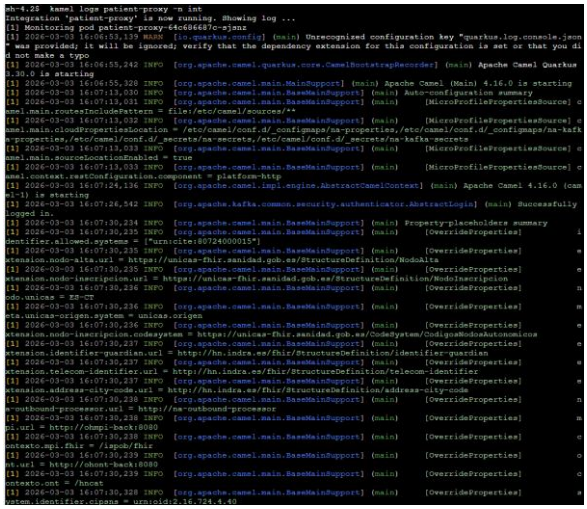
1. Comprobar despliegue del microservicio

Al ejecutar **kubectl get integration patient-proxy -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
NAME            PHASE     READY
EPLICAS
patient-proxy   Running   True
```

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs patient-proxy -n <namespace>**. Ejemplo de esta integración en "int":



Además se crea el gateway "patient-proxy-gateway", se puede comprobar su existencia en el namespace con el siguiente comando:

Si el gateway es un httproute: **kubectl get httproute patient-proxy-gateway -n <namespace>**

Si el gateway es un virtualservice: **kubectl get virtualservice patient-proxy-gateway -n <namespace>**

Ejemplo de un httproute en el namespace "int":

```
sh-4.2$  kubectl get httproute patient-proxy-gateway -n int
NAME                    HOSTNAMES   AGE
patient-proxy-gateway   []          9m18s
sh-4.2$
```

4.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: **helm uninstall patient-proxy -n <namespace>**.

```
sh-4.2$ helm uninstall patient-proxy -n int
release "patient-proxy" uninstalled
```

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- El gateway generado para la exposición de los endpoints.

5 Microservicio qr-to-condition

A partir de un QuestionarieResponse y de un momento Unicas vamos a enrolar un paciente al sistema UNICAS o en caso que ya lo estuviera le modificaremos los datos pertinentes con su correspondiente notificacion a NC tanto en el caso de alta como de modificacion Core.

La validación de la authenticación es por KeyCloakPolicy y el rol necesario es "OHBPM_UNICAS".

Comportamiento:

- valida que el body no esté vacío.
- obtiene el Patient sobre el que se ha rellenado el formulario en el propio MPI.
- Validamos que el CIPSNS contra NC.
- Si es enrolamiento:
    - SI el Patient ya tenia momento únicas:
        - Generar Condition.
        - Notificar la modificacion core a NC.
        - SI hay error en la notificacion se llama a la integración rollbackPatient.
    - si no tenía momento únicas:
        - Generar Condition.
        - Notificar alta paciente UNICAS a NC.
        - SI hay error en la notificación se llama a la integración rollbackPatient.
- No es enrolamiento:
    - Generar Condition.
    - Notificar la modificacion Core a NC.
    - SI hay error en la notificación se llama a la integración rollbackPatient.

5.1 ENDPOINTS DISPONIBLES

POST /fhir/Condition/fromQuestionnaireResponse

Entrada

Authentication: Bearer token

Accept: application/fhir+json; fhirVersion=5.0

Content-Type: application/fhir+json; fhirVersion=5.0

Body de entrada es un "Parameters" que incluye el nuevo momento unicas y el QuestionarieResponse

```
{
    "resourceType": "Parameters",
    "parameter": [
```

```
        {
            "name": "momento",
            "valueCoding": {
                "code": "DS"
            }
        },
        {
            "name": "formulario",
            "resource": {
                "resourceType": "QuestionnaireResponse",
                "id": "BPMQRP-41897",
                "meta": {
                    "versionId": "1",
                    "lastUpdated": "2025-10-02T10:52:16.263+00:00",
                    "tag": [
                        {
                            "system": "dsform.questionnaire.type",
                            "code": "1",
                            "display": "Hojas"
                        },
                        {
                            "system": "module",
                            "code": "ohbpm"
                        },
                        {
                            "system": "ohbpm.task.formKey",
                            "code": "547201119",
                            "display": "Enrolamiento en ÚNICAS"
                        },
                        {
                            "system": "ohbpm.task.priority",
                            "code": "1"
                        }
                    ]
                },
                "extension": [
                    {
                        "url":
"http://hn.indra.es/dsforms/fhir/Questionnaire/title",
                        "valueString": "Diagnostico UNICAS"
                    },
                    {
                        "url":
"http://hn.indra.es/hncli/fhir/QuestionnaireResponse/modification-user-
fullname",
                        "valueString": "Usuario administrador"
                    }
                ],
                "basedOn": [
                    {
                        "reference": "CarePlan/BPM-ES-CTCP-11003"
                    }
                ],
                "subject": {
                    "reference": "Patient/AC123456789"
                },
                "authored": "2025-10-01T16:16:31.305Z",
```

```
                "author": {
                    // "reference": "Practitioner/UC1761910184006",
                    "reference":
"Practitioner?identifier=urn:oid:1.3.6.1.4.1.19126.3|12234567Q",
                    "display": "us_unicas_asistencial_sjd"
                },
                "item": [
                    {
                        "linkId": "40",
                        "definition": "Consentimiento",
                        "text": "Consentimiento informado firmado:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "0",
                                    "display": "Sí"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "62",
                        "definition": "TipoConsentimiento",
                        "text": "Tipo de consentimiento:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "AS",
                                    "display": "Asistencia sanitaria"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "63",
                        "definition": "ConsentimientoInvestigacion",
                        "text": "¿Consiente que los datos (disociados y
anonimizados) sean utilizados como datos de investigación?",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "31874001",
                                    "display": "verdadero"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "41",
                        "definition": "code_system",
                        "text": "Sistema de codificación del diagnóstico",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "2",
                                    "display": "ORPHANET"
                                }
```

```
                        }
                    ]
                },
                {
                    "linkId": "46",
                    "definition": "code_ORPHANET",
                    "text": "Diagnóstico",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "79298",
                                "display": "Síndrome de deficiencia de
Glut1"
                            }
                        }
                    ]
                },
                {
                    "linkId": "55",
                    "definition": "category_ClasificacionDiagnostico",
                    "text": "Clasificación del diagnóstico",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "8319008",
                                "display": "Diagnóstico primario"
                            }
                        }
                    ]
                },
                {
                    "linkId": "57",
                    "definition": "note_text_condition",
                    "text": "Observaciones generales sobre el
diagnóstico",
                    "answer": [
                        {
                            "valueString": "ddddd"
                        }
                    ]
                },
                {
                    "linkId": "11",
                    "definition": "category_TipoDiagnostico",
                    "text": "Tipo de diagnóstico:",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "60022001",
                                "display": "En sospecha"
                            }
                        }
                    ]
                },
                {
                    "linkId": "18",
                    "definition": "evidence_GradoSospecha",
```

```
                        "text": "Grado de sospecha:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "2095291000122104",
                                    "display": "Sospecha alta de
enfermedad minoritaria"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "36",
                        "definition": "Observaciones sobre el grado de
sospecha",
                        "text": "Observaciones sobre el grado de
sospecha:",
                        "answer": [
                            {
                                "valueString": "Observaciones sobre el
grado de sospecha"
                            }
                        ]
                    },
                    {
                        "linkId": "10",
                        "definition": "onset",
                        "text": "Fecha de diagnóstico:",
                        "answer": [
                            {
                                "valueDate": "2025-09-30T22:00:00.000Z"
                            }
                        ]
                    },
                    {
                        "linkId": "58",
                        "definition": "estado_seguimiento",
                        "text": "¿El paciente se encuentra actualmente en
tratamiento y seguimiento?",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "64100000",
                                    "display": "Falso"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "19",
                        "definition":
"evidence_Criterios_DiagnosticoClinico",
                        "text": "Criterio de diagnóstico clínico:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "64100000",
```

```
                                "display": "Falso"
                            }
                        }
                    ]
                },
                {
                    "linkId": "22",
                    "definition": "evidence_Criterios_PruebaGenetica",
                    "text": "Criterio de prueba genética:",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "64100000",
                                "display": "Falso"
                            }
                        }
                    ]
                },
                {
                    "linkId": "25",
                    "definition":
"evidence_Criterios_PruebaBioquimica",
                    "text": "Criterio de prueba bioquímica:",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "64100000",
                                "display": "Falso"
                            }
                        }
                    ]
                },
                {
                    "linkId": "27",
                    "definition":
"evidence_Criterios_PruebaHematologica",
                    "text": "Criterio de prueba hematológica:",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "64100000",
                                "display": "Falso"
                            }
                        }
                    ]
                },
                {
                    "linkId": "29",
                    "definition":
"evidence_Criterios_PruebaHistologica",
                    "text": "Criterio de prueba histológica:",
                    "answer": [
                        {
                            "valueCoding": {
                                "code": "64100000",
                                "display": "Falso"
                            }
```

```
                            }
                        ]
                    },
                    {
                        "linkId": "32",
                        "definition":
"evidence_Criterios_PruebaInmunologica",
                        "text": "Criterio de prueba inmunológica:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "64100000",
                                    "display": "Falso"
                                }
                            }
                        ]
                    },
                    {
                        "linkId": "34",
                        "definition": "evidence_Criterios_PruebaImagen",
                        "text": "Criterio de prueba de imagen:",
                        "answer": [
                            {
                                "valueCoding": {
                                    "code": "64100000",
                                    "display": "Falso"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```


Code block 2 Body Json entrada

Salida

La estructura mensaje salida en caso de resultado 200 OK es recurso Condition derivado del Questionarie Response que se ha rellenado y que nos ha llegado por la entrada:


```
{
    "resourceType": "Condition",
    "id": "BPMCP-2364",
    "meta": {
        "versionId": "7",
        "lastUpdated": "2026-03-03T17:20:16.105+00:00",
        "profile": [
            "https://unicas-
fhir.sanidad.gob.es/StructureDefinition/UNICASConditionDiagnostico"
        ]
    },
```

```
    "identifier": [
        {
            "system": "urn:regcess:0908005177",
            "value": "Condition/BPMCP-2364"
        }
    ],
    "clinicalStatus": {
        "coding": [
            {
                "system":
"http://terminology.hl7.org/CodeSystem/condition-clinical",
                "code": "active",
                "display": "Active"
            }
        ]
    },
    "category": [
        {
            "coding": [
                {
                    "system": "http://snomed.info/sct/900000001000122104",
                    "code": "8319008",
                    "display": "Diagnóstico primario"
                }
            ]
        },
        {
            "coding": [
                {
                    "system": "http://snomed.info/sct/900000001000122104",
                    "code": "60022001",
                    "display": "En sospecha"
                }
            ]
        }
    ],
    "code": {
        "coding": [
            {
                "system": "urn:oid:2.16.724.4.21.5.22",
                "code": "ORPHA-2908",
                "display": "Epidermolisis bullosa de Kindler"
            }
        ],
        "text": "Epidermolisis bullosa de Kindler"
    },
    "subject": {
        "reference": "Patient/AC17636581806854100",
        "type": "Patient",
        "identifier": {
            "extension": [
                {
                    "url":
"http://hn.indra.es/fhir/StructureDefinition/identifier-guardian",
                    "valueBoolean": false
                }
            ],
```

```
            "type": {
                "coding": [
                    {
                        "system":
"http://snomed.info/sct/900000001000122104",
                        "code": "1551000122105",
                        "display": "código de identificación personal en
el Sistema Nacional de Salud"
                    }
                ]
            },
            "system": "urn:oid:2.16.724.4.40",
            "value": "BBBBBBBBHH000849"
        }
    },
    "onsetDateTime": "2025-11-03T23:00:00.000Z",
    "participant": [
        {
            "actor": {
                "reference":
"Practitioner?identifier=urn:oid:1.3.6.1.4.1.19126.3|12234567Q",
                "type": "Practitioner",
                "identifier": {
                    "use": "official",
                    "system": "urn:oid:1.3.6.1.4.1.19126.3",
                    "value": "12234567Q"
                },
                "display": "Facultativo Unicas SJD"
            }
        }
    ],
    "evidence": [
        {
            "concept": {
                "coding": [
                    {
                        "system":
"http://snomed.info/sct/900000001000122104",
                        "code": "2095221000122101",
                        "display": "grado de sospecha de enfermedad
minoritaria"
                    },
                    {
                        "system":
"http://snomed.info/sct/900000001000122104",
                        "code": "2095301000122103",
                        "display": "Sospecha media de enfermedad
minoritaria"
                    }
                ]
            }
        }
    ]
}
```


Code block 3 Body de salida

En caso de error controlado se recibe un recurso OperationOutcome. Por ejemplo:


```
{
   "resourceType": "OperationOutcome",
   "issue": [
      {
         "severity": "error",
         "code": "exception",
         "diagnostics": "El CipSNS no es válido.. Consulte el topic 'qr-to-condition-error' para más información."
      }
   ]
}
```


Posibles respuestas:

Código	Descripción
200	Accion realizada con éxito
400	Error en la petición realizada por el cliente
500	Error interno. Consultar topic "qr-to-condition-error"

5.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **qr-to-condition**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.
- La configuración externa.
- Los recursos adicionales necesarios

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio.

5.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión. Para ello es necesario sustituir el directorio "qr-to-condition-chart" existente por el nuevo. Previamente se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se va a instalar la integración.

```yaml
integration:
  name: qr-to-condition
  namespace: #namespace en el que se quiere desplegar la integración
  runtime:
    provider: plain-quarkus
```

En este caso, el único valor que debemos cambiar es el del campo "namespace", indicando en cuál de los disponibles queremos que se instale.

Una vez modificado, hay que ejecutar el siguiente comando para aplicar la actualización: **helm upgrade qr-to-condition ./qr-to-condition-chart -n <namespace>.** Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
Release "qr-to-condition" has been upgraded. Happy Helming!
NAME: qr-to-condition
LAST DEPLOYED: Tue Mar  3 17:39:49 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

5.2.2 VERIFICACIÓN DE LA INSTALACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio:

Al ejecutar **kubectl get integration qr-to-condition -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration qr-
NAME               PHASE     READY
 REPLICAS
qr-to-condition    Running   True
 1
```

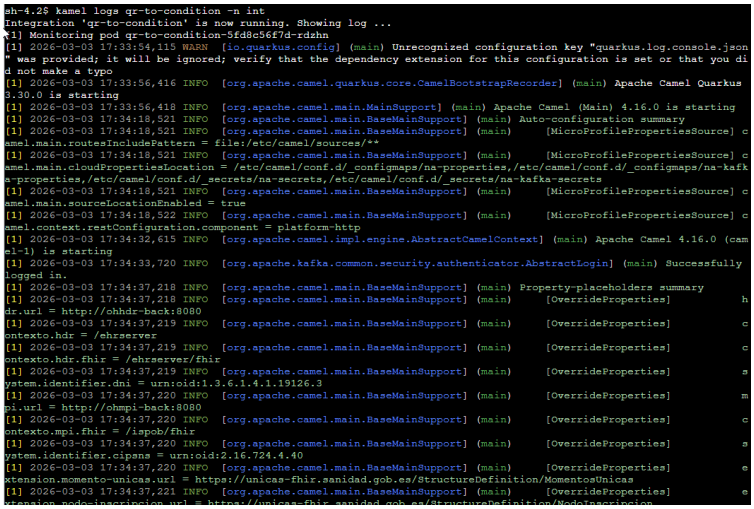
2. Comprobar el despliegue de los configmap:

Al ejecutar **kubectl get cm qr-to-condition -n <namespace>** se puede comprobar su existencia en el namespace con el siguiente comando. Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get cm qr-to-condition-qute -n int
NAME                   DATA   AGE
qr-to-condition-qute   3      3m11s
```

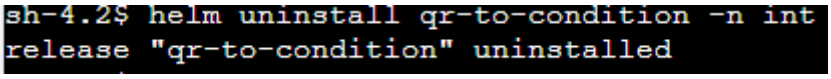
3. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs qr-to-condition -n <namespace>**. Ejemplo en "int":



5.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: **helm uninstall qr-to-condition -n <namespace>**.



Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los ConfigMaps definidos dentro del Helm chart.

6 Microservicio procesar-notificaciones-nc

La integración se encarga de la gestión de las notificaciones que NC manda a los Nodos autonómicos. Consta de 3 partes principales:

- Recepción notificación urgente:
    - des del endpoint antes indicado recibimos una aviso que hay una notificación pendiente de consumir que requiere de urgente atención.
    - Con el mensaje recibido se hace un GetALl que consume todas las notificaciones para nuestro nodo autonómico y los encola a Kafka.
    - Luego el proceso que procesa las notificaciones hará su correspondiente trabajo.
    - Validación de la autenticación mediante KeycloakPolicy. Rol permitido: "HDR_BASE_R".
- Proceso programado:
    - Cada hora, en la hora en punto concretamente, se lanza un GetAll que obtendrá todas las notificaciones par anuestro nodo autonómico pendientes de consumir y las encolara a kafka para que el proceso que las trata haga su trabajo.
- Procesado de notificaciones:
    - tenemos un Consumer kafka a todos los topics correspondientes a las notificaciones a procesar:
        - notif_alta_paciente:
            - si el origen de la notificación es diferente a nuestro propio nodo:
                - si el nodo de inscripción del paciente es la nuestra, obtenemos el paciente completo des de esta comunidad a través de NC, y guardaremos el Patient en nuestro MPI.
                - si el nodo de inscripción no es el nuestro, construiremos el paciente a partir de los datos core recibidos en la notificación de alta, y lo guardaremos en nuestro MPI.
        - notif_modificacion_core:
            - Caso "Exitus":
                - marcamos como deceso al Patient en nuestro MPI.
            - Caso "no exitus":
                - aplicamos los cambios en los datos core proporcionados en la notificacion sobre nuestro Patient.
        - notif_modificacion_generica:
            - Obtenemos el Patient completo en el nodo de origen de la notificación y simplemente le hacemos PUT en nuestro MPI.

- notif_na_inactivo:
  - Actualmente no se procesa nada en el nodo autonómico.
- notif_manual:
  - Actualmente no se procesa nada en el nodo autonómico.

6.1 ENDPOINTS DISPONIBLES

POST /ie/avisourgente

Notificar que hay una notificación urgente para atender. El mismo proceso va a procesar todas las notificaciones pendientes del nodo.

Entrada

Authentication: Bearer token

Accept: application/json, application/*+json

Estructura cuerpo:

```
{
  "mensaje": "Se ha registrado una notificación de carácter urgente que requiere de atención inmediata", ← texto fijo
  "idNotificacionUrgente": "idNotificacion",  ← id de la notificación urgente
  "fechaAviso": "2026-02-06T13:13:20", ← fecha del envio
  "observaciones": ""
}
```

Salida

El peticionario va a recibir un 200 OK si el nodo autonomico ha recibido correctamente la llamada a través de este endpoint.

Código	Descripción
200	Alta/modificación del paciente realizada correctamente en el MPI local
400	Solicitud incorrecta
500	Error interno

6.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **procesar-notificaciones-nc**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.

- La configuración externa.
- Los recursos adicionales necesarios (incluidas plantillas Qute, schemas,... si los hubiera).
- Configuracion del gateway.

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio.

6.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión.

A continuación, se describe el procedimiento para la correcta actualización de la integración **procesar-notificaciones-nc** utilizando Helm.

1. Preparación del chart

Verificar que la estructura del directorio procesar-notificaciones-nc-chart es la siguiente:

```
procesar-notificaciones-nc-chart
├── CHANGELOG.md
├── Chart.yaml
├── files
│   ├── aggregation
│   │   └── DatosCoreAggregationStrategy.java
│   ├── Authentication.java
│   ├── ConsultaNotificacionesNC.java
│   ├── KeyCloakPolicy.java
│   ├── NotificacionCriticaNC.java
│   ├── ProcesadorNotificacionesKafka.java
│   ├── templatesqute
│   │   ├── bundle-searchset.qute.json
│   │   ├── error-notificacion.qute.json
│   │   ├── paciente-core.qute.json
│   │   └── Parameters-patient.qute.json
│   └── templatesschemas
│       ├── AvisoUrgente.schema.json
│       ├── ListaNotificaciones.schema.json
│       ├── NotificacionAltaPaciente.schema.json
│       ├── NotificacionManual.schema.json
│       ├── NotificacionModificacionCORE.schema.json
│       ├── NotificacionModificacionGenerica.schema.json
│       └── NotificacionNAInactivo.schema.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   ├── httproute.yaml
│   ├── integration.yaml
│   └── virtualservice.yaml
└── values.yaml
```

2. Configuración Previa

Modificar el archivo **values.yaml**, para indicar el namespace en el que se quiera instalar la integración.

```
integration:
  name: procesar-notificaciones-nc
  namespace: #namespace en el que se quiere desplegar la integración
  runtime:
    provider: plain-quarkus
```

En este caso también tendremos que indicar el tipo de gateway que hay para que se expongan los endpoints de la integración: **k8_gateway** o **istio**.

```
gateway:
  type: # tipo de gateway. Valores: k8s_gateway/istio
  name: procesar-notificaciones-nc-gateway
  unicasbase: #url base del servidor. Por ejemplo: integracio.unicas.salut.intranet.gencat.cat
```

3. Ejecutar comando de instalación.

es necesario sustituir el directorio "procesar-notificaciones-nc-chart" existente por el nuevo y ejecutar el siguiente comando: **helm upgrade procesar-notificaciones-nc ./procesar-notificaciones-nc-chart -n <namespace>.** Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade procesar-notificaciones-nc ./procesar-notificaciones-nc-chart -n int
Release "procesar-notificaciones-nc" has been upgraded. Happy Helming!
NAME: procesar-notificaciones-nc
LAST DEPLOYED: Wed Mar  4 09:53:24 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
sh-4.2$
```

6.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

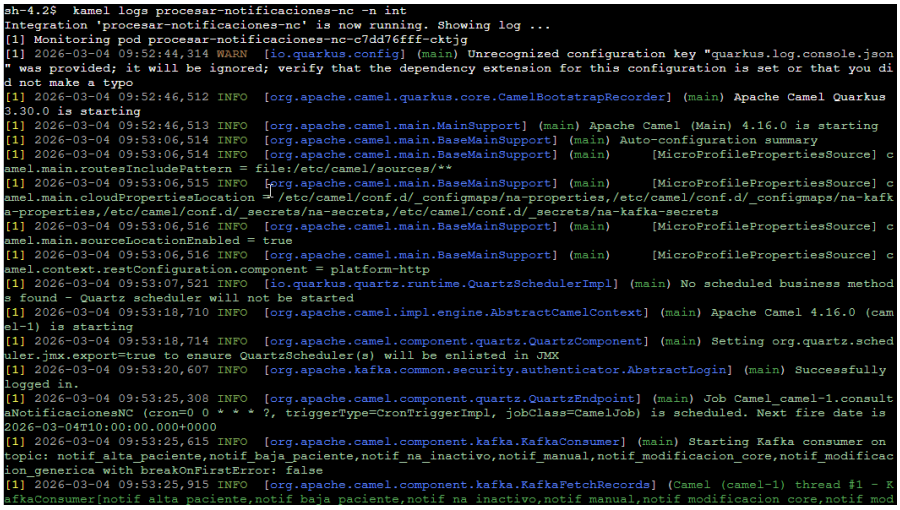
1. Comprobar despliegue del microservicio:

Al ejecutar **kubectl get integration procesar-notificaciones-nc -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$  kubectl get integration procesar-not
NAME                          PHASE     READY
            REPLICAS
procesar-notificaciones-nc    Running   True
4tpljvi0g   1
```

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs procesar-notificaciones-nc -n <namespace>**. Ejemplo de esta integración en "int":

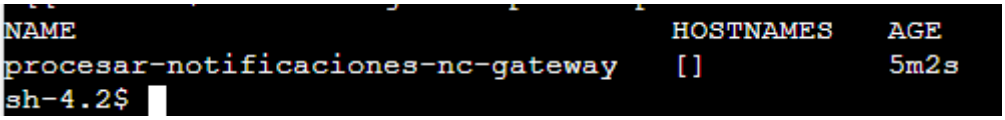


Además se crea el gateway "procesar-notificaciones-nc-gateway", se puede comprobar su existencia en el namespace con el siguiente comando:

Si el gateway es un httproute: **kubectl get httproute procesar-notificaciones-nc-gateway -n <namespace>.**

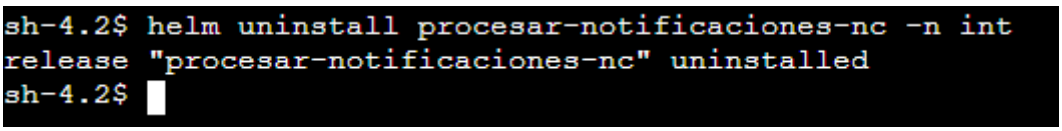
Si el gateway es un virtualservice: **kubectl get virtualservice procesar-notificaciones-nc-gateway -n <namespace>.**

Ejemplo de un httproute en el namespace "int":



6.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: **helm uninstall procesar-notificaciones-nc -n <namespace>**.



Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los configmap generados por la integración (qute y schemas).

- El gateway generado para la exposición de los endpoints.

7 Microservicio obtener-info-paciente

Este proceso solicita información de datos clínicos de un paciente a la red UNICAS mediante NC.
Este devolverá todos los recursos encontrados sobre este paciente en los demás nodos de la red y los guardaremos en nuestro repositorio identificándolos como datos obtenidos a través de la red Únicas.

Una vez obtenidos los recursos se registra la fecha para que en la siguiente solicitud de datos n o se vuelvan a pedir los que ya se han obtenido con anterioridad

Comportamiento detallado:

- Obtenemos el CipSNS del Paciente proporcionado
- Obtener fecha de la ultima obtención de datos
- Enviar primera consulta a NC
- Procesar recursos recibidos, obteniendo los Organization y Practitioner anidados que Tengan dichos recursos e incluyéndolos en el bundle que será guardado en el repositorio del nodo autonómico. Todo recurso obtenido mediante este proceso será identificado con el meta con "system": "urn:unicas:origen" y "code": nodo origen del recurso. Se devuelve de una sola vez toda la información recapitulada en un único nodo y se genera un bundle para guardar de una única consulta todos los recursos recogidos de este primer nodo.
- Si el header "es_consulta_finalizada" esta a false, se consulta el siguiente nodo con información disponible. Y así en bucle hasta que es_consulta_finalizada venga a true. En cada una de estas iteraciones se repite el proceso descrito en el punto anterior.
- Una vez recogida toda la información del paciente se genera un nuevo registro AuditEvent en la que se guarda la fecha de la última actualización de datos del paciente consultado.
- En caso de que algún nodo este inactivo en la ultima "paginación" se devuelve una lista de dichos nodos.

7.1 ENDPOINTS DISPONIBLES

GET /ie/patient-data-request?_format=json&patient=ACxxxxxx

Solicitud de datos clínicos del paciente correspondiente al identificador ÚNICAS indicado en en parámetro "patient" en formato json

Entrada

Authentication: Bearer token. Se requiere rol "HDR_BASE_W"

Accept: application/json, application/*+json

Estructura cuerpo: N/A

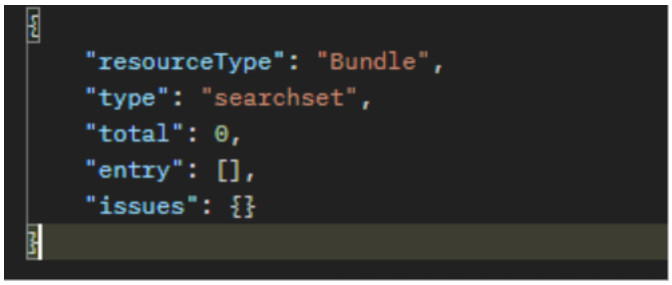
Salida

Código	Descripción
200	Solicitud correcta

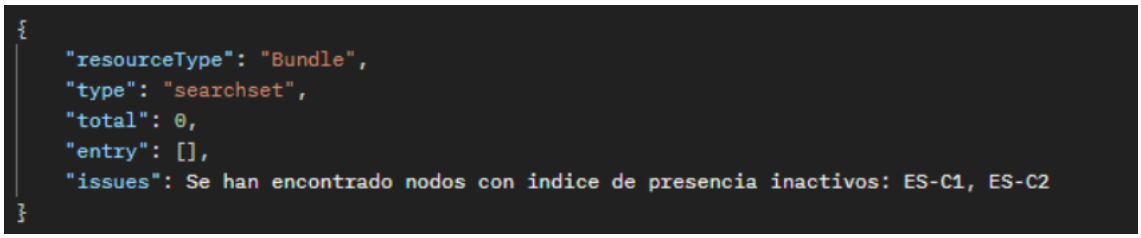
400	Solicitud incorrecta
500	Error interno

Salida correcta (200 OK) devolverá un bundle searchset con el total de registros insertados en nuestro repositório, el array de entry's vacio (por temas de optimización) e "issues" aportará información en caso de haberse producido un inconveniente controlado como que alguno de los nodos consultados estuviera inactivo.

Por ejemplo en este caso el resultado ha sido correcto y no se han obtenido datos nuevos:

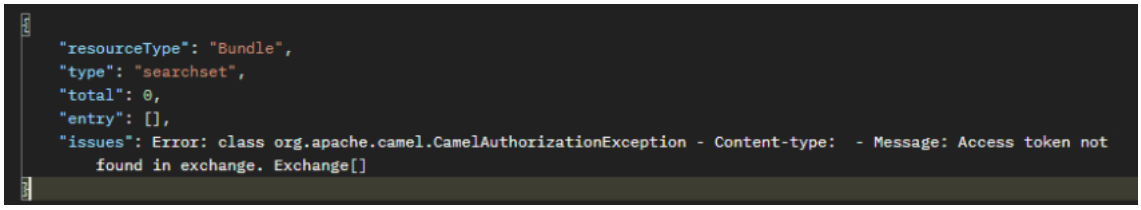


o en esta otra consulta en la que hay presencia en nodos que estan inactivos y por ello no se ha podido devolver los datos que esos nodos contenian si fuera el caso:



En caso de error se devolverá el mismo bundle searchset pero con la informacion del error en "issues" que puede contener tanto un string como un recurso operationOutcome con el detalle del error. Se puede obtener más información del error producido en el topic kafka "obtener-info-paciente-error".

Por ejemplo, en este caso es por falta de autenticación (falta de token)



7.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualizaciñon y desinstalación del microservicio obtener-info-paciente, el cual se despliega como una integración Camel K empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K
- Sus dependencias
- La configuración externa
- Los recursos adicionales necesarios (incluidas plantillas Qute, schemas,... si los hubiera)
- Configuracion del gateway

Para una descripción más detallada del Helm chart, consultar el archivo README.md del directorio.

7.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión. Para ello es necesario sustituir el directorio "obtener-info-paciente-chart" existente por el nuevo. Previamente se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se va a instalar la integración.

```
integration:
  name: obtener-info-paciente
  namespace: #namespace en el que se quiere desplegar la integración
  runtime:
    provider: plain-quarkus
```

En este caso también tendremos que indicar el tipo de gateway que hay para que se expongan los endpoints de la integración: **k8_gateway** o **istio**.

```
gateway:
  type: # tipo de gateway. Valores: k8s_gateway/istio
  name: obtener-info-paciente-gateway
  unicasbase: #url base del servidor. Por ejemplo: integracio.unicas.salut.intranet.gencat.cat
```

Una vez modificado, hay que ejecutar el siguiente comando para aplicar la actualización: **helm upgrade obtener-info-paciente ./obtener-info-paciente-chart -n <namespace>.**
Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade obtener-info-paciente ./obtener-info-paciente-chart/ -n int
Release "obtener-info-paciente" has been upgraded. Happy Helming!
NAME: obtener-info-paciente
LAST DEPLOYED: Wed Mar  4 10:51:57 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

7.2.2 VERIFICACIÓN DE LA INSTALACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio

Al ejecutar **kubectl get integration obtener-info-paciente -n <namespace>** Podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration obtener-ir
NAME                        PHASE     READY
       REPLICAS
obtener-info-paciente       Running   True
vie0   1
```

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs obtener-info-paciente -n <namespace>**. Ejemplo de esta integración en "int":



Además se crea el gateway "obtener-info-paciente-gateway", se puede comprobar su existencia en el namespace con el siguiente comando:

Si el gateway es un httproute: **kubectl get httproute obtener-info-paciente-gateway -n <namespace>**

Si el gateway es un virtualservice: **kubectl get virtualservice obtener-info-paciente-gateway -n <namespace>**

Ejemplo de un httproute en el namespace "int":

```
sh-4.2$ kubectl get httproute obtener-info-paciente-gateway -n int
NAME                              HOSTNAMES   AGE
obtener-info-paciente-gateway     []          7m58s
```

7.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: **helm uninstall obtener-info-paciente -n <namespace>**.

```
sh-4.2$ helm uninstall obtener-info-paciente -n int
release "obtener-info-paciente" uninstalled
sh-4.2$
```

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los configmap generados por la integración
- El gateway generado para la exposición de los endpoints

8 Microservicio download-binary

Este servicio devuelve el Binary con el binaryId indicado de un nodo concreto

8.1 ENDPOINTS DISPONIBLES

GET /ie/Binary/{binaryId}?nodos={nodo-a-consultar}

Parametros posibles:

- Path
  - binaryId: identificador del binary
- Query
  - nodos: nodo al que va dirigida la consulta. Puede contener más de un nodo separados por coma ","

Entrada

- Authentication: Bearer token. Se requiere rol "HDR_FOUNDATION_R"
- Estructura cuerpo: N/A

Salida

Código	Descripción
200	Solicitud correcta
400	Solicitud incorrecta
500	Error interno

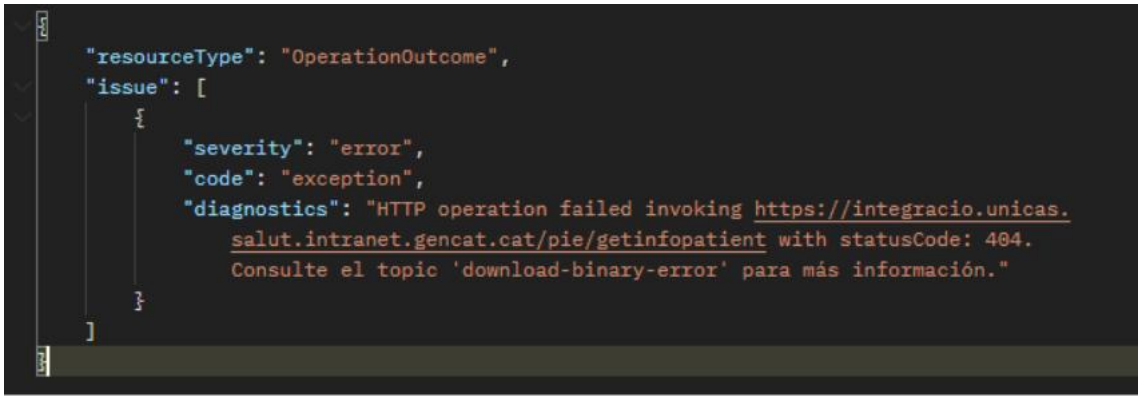
Devuelve por el body el recurso Binary entero



En caso de error se devolverá un OperationPutcome con información del error producido.

Se puede obtener más información del error producido en el topic kafka "download-binary-error".

Por ejemplo,



8.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **download-binary**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K
- Sus dependencias
- La configuración externa
- Los recursos adicionales necesarios (incluidas plantillas Qute, schemas,... si los hubiera)
- Configuracion del gateway

Para una descripción más detallada del Helm chart, consultar el archivo README.md del directorio.

8.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **download-binary** utilizando Helm.

AVISO: los nombres de directorios mostrados en las imagenes pueden diferir del real a utilizar.

1. Preparación del chart

Verificar que la estructura del directorio download-binary-chart es la siguiente:

```
download-binary-chart/
├── CHANGELOG.md
├── Chart.yaml
├── files
│   ├── DownloadBinary.java
│   ├── HeaderFilterStrategy.java
│   ├── KeyCloakPolicy.java
│   └── templatesqute
│       └── bundle-batch.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   ├── httproute.yaml
│   ├── integration.yaml
│   └── virtualservice.yaml
└── values.yaml
```

2. Configuración Previa

Para la instalación, sólo se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se quiera instalar la integración.

```yaml
integration:
  name: download-binary
  namespace: # namespace
  runtime:
    provider: plain-quarkus
```

En este caso también tendremos que indicar el tipo de gateway que hay para que se expongan los endpoints de la integración: **k8_gateway** o **istio**.

```yaml
gateway:
  type: k8s_gateway
  name: download-binary-gateway
  unicasbase: # url base
```

3. Ejecutar comando de instalación.

A continuación, deberemos ejecu4tar el comando siguiente: **helm install download-binary ./download-binary-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm uninstall download-binary -n int
release "download-binary" uninstalled
sh-4.2$ helm install download-binary ./download-binary-chart-1.0.0/ -n int
NAME: download-binary
LAST DEPLOYED: Thu Apr 23 14:52:35 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration download-binary
- Httproute download-binary-gateway

8.2.2 VERIFICACIÓN DE LA INSTALACIÓN

Para confirmar que se ha creado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio

Al ejecutar **kubectl get integration download-binary -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel logs download-binary -n <namespace>**. Ejemplo de esta integración en "int":

Además se crea el gateway "download-binary-gateway", se puede comprobar su existencia en el namespace con el siguiente comando:

Si el gateway es un httproute: **kubectl get httproute download-binary-gateway -n <namespace>**

Si el gateway es un virtualservice: **kubectl get virtualservice download-binary-gateway -n <namespace>**

Ejemplo de un httproute en el namespace "int":

8.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración, es necesario sustituir el directorio "download-binary-chart" existente por el nuevo y ejecutar el siguiente comando: **helm upgrade download-binary ./download-binary-chart -n <namespace>.** Nos aparecerá un mensaje como el siguiente al ejecutarlo:

8.2.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: **helm uninstall download-binary -n <namespace>**.

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los configmap generados por la integración
- El gateway generado para la exposición de los endpoints

9 Microservicio inject-momento-encounter

Este microservicio se encarga de añadir el "Momento UNICAS" del paciente en la etiqueta metadata(meta.tag) de los recursos FHIR Encounter que no lo tengan.

Este microservicio no expone endpoints REST: funciona por consumo de KAFKA .

Para crear el topic de kafka, en la instalación de la configuración, se han creado una serie de recursos FHIR Subscription, uno de ellos para ser utilizado por este microservicio. Esa suscripción detecta los Encounter que aún no tienen ningún "Momento UNICAS" en su metadata y publica un mensaje en el topic **"inject-momento-encounter-topic"**, que incluye el Encounter dentro del campo payload. A partir de este mensaje, el microservicio identifica el Encounter, obtiene la referencia del paciente, recupera del MPI el Momento UNICAS almacenado en las extensiones del Patient y genera un PATCH FHIR que actualiza el Encounter en el servidor HDR. Si la operación se completa correctamente devuelve un OperationOutcome de éxito, y si ocurre algún error, genera un OperationOutcome de error y lo publica en el topic de errores **"inject-momento-encounter-topic-error"**.

También se incluye la clase Authenticacion para poder incluir Authorization Bearer token en la llamada a MPI y HDR.

El mensaje contiene un JSON con un recurso Encounter en payload.
De ese contenido, la integración utiliza solo tres datos clave:

- ID del Encounter
    - Campo: payload.id
    - Usado para construir el PATCH al recurso Encounter
- Referencia del paciente
    - Campo principal: payload.subject.reference
    - Alternativa: payload.subject.identifier.value
    - Usado para consultar el recurso Patient en el MPI

9.1 ENDPOINTS DISPONIBLES

No aplica.

9.2 ACTUALIZACIÓN HELM CHART

Este apartado describe el procedimiento para la actualización y desinstalación del microservicio **inject-momento-encounter**, el cual se despliega como una **integración Camel K** empaquetada en un chart Helm específico.

Este chart incluye:

- La integración Camel K.
- Sus dependencias.
- La configuración externa.

- Los recursos adicionales necesarios, incluidas plantillas Qute de respuesta y patch del Encounter.

Para una descripción más detallada del Helm chart, consultar el archivo **README.md** del directorio inject-momento-encounter-chart.

9.2.1 ACTUALIZACIÓN

Este microservicio se tiene que actualizar de manera obligatoria para esta versión. Para ello es necesario sustituir el directorio " inject-momento-encounter-chart" existente por el nuevo.
Previamente se deberá modificar el archivo **values.yaml**, para indicar el namespace en el que se quiera actualizar la integración.

```
integration:
  name: inject-momento-encounter
  namespace: #namespace en el que se quiere desplegar la integración
  dependencies:
    - "camel:core"
    - "camel:direct"
    - "camel:caffeine"
    - "camel:jcache"
```

En este caso, el único valor que debemos cambiar es el "namespace", indicando en cuál de los disponibles queremos que se actualice.

Para efectuar la modificación, hay que ejecutar el siguiente comando:

helm upgrade inject-momento-encounter ./inject-momento-encounter-chart -n <namespace>

Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade inject-momento-encounter ./inject-momento-encounter-chart -n int
Release "inject-momento-encounter" has been upgraded. Happy Helming!
NAME: inject-momento-encounter
LAST DEPLOYED: Tue Mar  3 08:51:27 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

9.2.2 VERIFICACIÓN DE LA ACTUALIZACIÓN

Para confirmar que se ha actualizado correctamente la integración, se pueden ejecutar los siguientes comandos:

1. Comprobar despliegue del microservicio:

Al ejecutar **kubectl get integration inject-momento-encounter -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration inject-momento-encounter -n int
NAME                      PHASE     READY   RUNTIME PROVIDER   RUNTIME VERSION   CATALOG VERSION   KIT                       REPLICAS
inject-momento-encounter   Running   True    plain-quarkus      3.30.2            3.30.2            kit-daofkjbjin2e6440p2q0   1
sh-4.2$
```

En este caso, que también se debe crear el configmap "inject-momento-encounter-qute", se comprueba su existencia en el namespace con el siguiente comando: **kubectl get cm inject-momento-encounter-qute -n <namespace>**

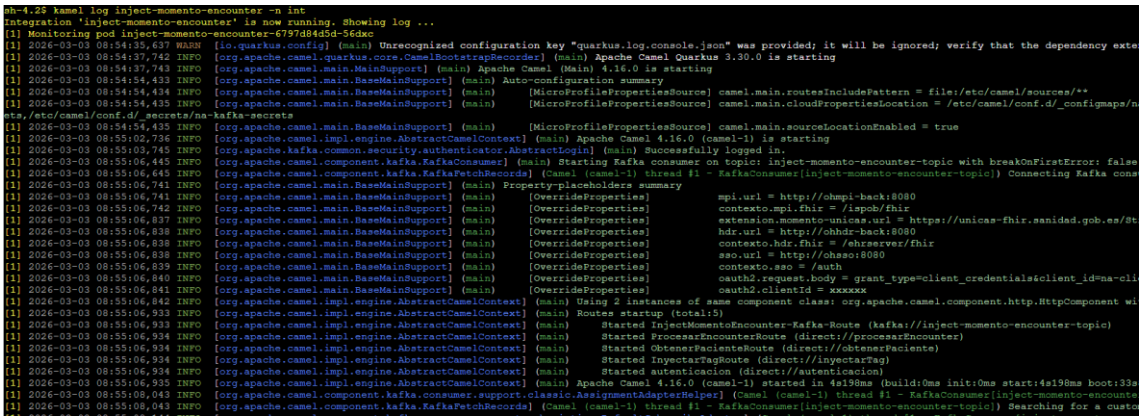
Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm inject-momento-encounter-qute -n int
NAME                                DATA   AGE
inject-momento-encounter-qute       2      20d
sh-4.2$
```

2. Consultar logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: **kamel log inject-momento-encounter -n <namespace>**

Ejemplo de esta integración en namespace "int":



9.2.3 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo por un error en la definición de parámetros en values.yaml, se ejecuta el siguiente comando: **helm uninstall inject-momento-encounter -n <namespace>**

```
sh-4.2$ helm uninstall inject-momento-encounter -n int
release "inject-momento-encounter" uninstalled
```

Al ejecutar el comando, Helm elimina:

- La integración de Camel K creada por el chart.
- El deployment, build u otros recursos generados automáticamente por Camel K al existir la integración.
- Los pods ejecutados por la integración.
- Los ConfigMap definidos dentro del Helm chart "inject-momento-encounter-qute".