

# Documentación Microservicios NA ÚNICAS

Implantación NA UNICAS



Minsait  
Marzo 2026

### Histórico de versiones:

Versión	Realizado por:	Fecha
1.0	Minsait	24/02/2026
1.1	Revisión Oficina Técnica	26/02/2026
1.2	Revisión Minsait	04/03/2026

## Contenido

<b>1</b>	<b>INTRODUCCIÓN</b>	<b>5</b>
<b>2</b>	<b>Recursos comunes para los microservicios</b>	<b>6</b>
2.1	INSTALACIÓN HELM CHART	6
2.1.1	INSTALACIÓN	6
2.1.2	VERIFICACIÓN DE LA INSTALACIÓN	7
2.1.3	ACTUALIZACIÓN	9
2.1.4	DESINSTALACIÓN	9
<b>3</b>	<b>Microservicio na-health-check</b>	<b>10</b>
3.1	ENDPOINTS DISPONIBLES	10
3.2	INSTALACIÓN HELM CHART	11
3.2.1	INSTALACIÓN	11
3.2.2	VERIFICACIÓN DE LA INSTALACIÓN	12
3.2.3	ACTUALIZACIÓN	13
3.2.4	DESINSTALACIÓN	13
<b>4</b>	<b>Microservicio validate-patient</b>	<b>14</b>
4.1	ENDPOINTS DISPONIBLES	14
4.2	INSTALACIÓN HELM CHART	15
4.2.1	INSTALACIÓN	16
4.2.2	VERIFICACIÓN DE LA INSTALACIÓN	17
4.2.3	ACTUALIZACIÓN	17
4.2.4	DESINSTALACIÓN	18
<b>5</b>	<b>Microservicio salida-unicas</b>	<b>19</b>
5.1	ENDPOINTS DISPONIBLES	19
5.2	INSTALACIÓN HELM CHART	21
5.2.1	INSTALACIÓN	21
5.2.2	VERIFICACIÓN DE LA INSTALACIÓN	22
5.2.3	ACTUALIZACIÓN	23
5.2.4	DESINSTALACIÓN	23
<b>6</b>	<b>Microservicio na-outbound-processor</b>	<b>24</b>
6.1	ENDPOINTS DISPONIBLES	24
6.2	INSTALACIÓN HELM CHART	31
6.2.1	INSTALACIÓN	31
6.2.2	VERIFICACIÓN DE LA INSTALACIÓN	33
6.2.3	ACTUALIZACIÓN	33
6.2.4	DESINSTALACIÓN	34
<b>7</b>	<b>Microservicio indice-presencia-nc</b>	<b>35</b>
7.1	ENDPOINTS DISPONIBLES	35
7.2	INSTALACIÓN HELM CHART	35
7.2.1	INSTALACIÓN	35
7.2.2	VERIFICACIÓN DE LA INSTALACIÓN	37
7.2.3	ACTUALIZACIÓN	37
7.2.4	DESINSTALACIÓN	38
<b>8</b>	<b>Microservicio careteam-mpi-creator</b>	<b>39</b>
8.1	ENDPOINTS DISPONIBLES	39
8.2	INSTALACIÓN HELM CHART	40
8.2.1	INSTALACIÓN	40
8.2.2	VERIFICACIÓN DE LA INSTALACIÓN	41
8.2.3	ACTUALIZACIÓN	42
8.2.4	DESINSTALACIÓN	42
<b>9</b>	<b>Microservicio inject-momento-encounter</b>	<b>44</b>
9.1	ENDPOINTS DISPONIBLES	44
9.2	INSTALACIÓN HELM CHART	44
9.2.1	INSTALACIÓN	45
9.2.2	VERIFICACIÓN DE LA INSTALACIÓN	46
9.2.3	ACTUALIZACIÓN	47
9.2.4	DESINSTALACIÓN	47
<b>10</b>	<b>Microservicio practitioner-sync</b>	<b>48</b>

10.1	ENDPOINTS DISPONIBLES .....	48
10.2	INSTALACIÓN HELM CHART .....	50
10.2.1	INSTALACIÓN .....	50
10.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	51
10.2.3	ACTUALIZACIÓN .....	52
10.2.4	DESINSTALACIÓN .....	52
<b>11</b>	<b>Microservicio patient-proxy .....</b>	<b>53</b>
11.1	ENDPOINTS DISPONIBLES .....	53
11.2	INSTALACIÓN HELM CHART .....	57
11.2.1	INSTALACIÓN .....	57
11.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	59
11.2.3	ACTUALIZACIÓN .....	60
11.2.4	DESINSTALACIÓN .....	60
<b>12</b>	<b>Microservicio qr-to-condition .....</b>	<b>61</b>
12.1	ENDPOINTS DISPONIBLES .....	61
12.2	INSTALACIÓN HELM CHART .....	69
12.2.1	INSTALACIÓN .....	69
12.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	70
12.2.3	ACTUALIZACIÓN .....	71
12.2.4	DESINSTALACIÓN .....	71
<b>13</b>	<b>Microservicio procesar-notificaciones-nc .....</b>	<b>73</b>
13.1	ENDPOINTS DISPONIBLES .....	74
13.2	INSTALACIÓN HELM CHART .....	74
13.2.1	INSTALACIÓN .....	74
13.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	76
13.2.3	ACTUALIZACIÓN .....	77
13.2.4	DESINSTALACIÓN .....	77
<b>14</b>	<b>Microservicio enriquecer-url-attachment .....</b>	<b>79</b>
14.1	ENDPOINTS DISPONIBLES .....	79
14.2	INSTALACIÓN HELM CHART .....	79
14.2.1	INSTALACIÓN .....	79
14.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	81
14.2.3	ACTUALIZACIÓN .....	81
14.2.4	DESINSTALACIÓN .....	81
<b>15</b>	<b>Microservicio obtener-info-paciente .....</b>	<b>83</b>
15.1	ENDPOINTS DISPONIBLES .....	83
15.2	INSTALACIÓN HELM CHART .....	84
15.2.1	INSTALACIÓN .....	85
15.2.2	VERIFICACIÓN DE LA INSTALACIÓN .....	86
15.2.3	ACTUALIZACIÓN .....	87
15.2.4	DESINSTALACIÓN .....	87

# 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.

Nos hemos basado en Camel K para crear estas integraciones, ya que nos permite crear flujos de integración contruidos 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 instalar, actualizar o desinstalar cada uno de ellos 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.

A continuación, se procede a explicar cada uno de los microservicios generados y como instalarlos, así como los recursos comunes que se necesitan para la ejecución de las integraciones (*configmaps, secrets...*).

## 2 Recursos comunes para los microservicios

Para la correcta ejecución de los diferentes microservicios es fundamental contar con una serie de recursos en el namespace que hemos definido como comunes, ya que se usan en diversos microservicios. Estos son:

- configmap **na-properties**: este configmap contiene propiedades comunes.
- configmap **na-kafka-properties**: este configmap contiene propiedades relacionadas con Kafka.
- secret **na-secrets**: secret que contiene contraseñas comunes.
- secret **na-kafka-secrets**: secret necesario para Kafka.
- httproute **nc-na-http-route** / virtual service **nc-na-virtualservice**: es el recurso que habilita los endpoints que NA debe exponer a NC. Si el tipo de gateway es k8s se creará el httproute, si es istio se creará el virtual service.
- secret **na-client-cert-nc**: secret con el certificado que se utiliza para la comunicación entre NA y NC.

A continuación, se indica como instalar el Helm que crea estos recursos adaptados al entorno y namespace que corresponda.

### 2.1 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación de los recursos comunes de los microservicios de ÚNICAS, empaquetados en un chart Helm específico.

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

#### 2.1.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de los recursos comunes definidos en `properties-chart` utilizando Helm.

##### 1. Preparación del chart

Verificar que la estructura del directorio `properties-chart` es la siguiente:

```
properties-chart/
├── Chart.yaml
├── files
│   ├── na-kafka-properties.properties.tpl
│   ├── na-kafka-secrets.properties.tpl
│   ├── na-properties.properties.tpl
│   └── na-secrets.properties.tpl
├── README.md
├── templates
│   ├── certsecret.yaml
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   ├── httproute.yaml
│   ├── secret.yaml
│   └── virtualservice.yaml
└── values.yaml
```

En la misma carpeta en la que se encuentre este directorio debe existir otro que contenga el certificado. El nombre de este directorio y el del certificado (en formato .p12) es indiferente para la correcta instalación del Helm. Simplemente tendrá que usarse en el comando de instalación, como se mostrará en el punto 3 de este mismo epígrafe.

## 2. Configuración Previa

Para la instalación, sólo se deberá modificar el archivo **values.yaml**, con los valores que dependen del nodo y el entorno. Estos son:

- `identifierallowedsystems: ["urn:cite:x"]` # sustituye x por el urn de la comunidad autónoma
- `ncbaseurl: #` url del nodo central.
- `nodounicas: #` código del nodo autonómico. Ejemplo: ES-CT (Seguir Anexo I)
- `nodounicasdisplay: #` display asociado al código del nodo autonómico. (Seguir Anexo I)
- `unicasbase: #` dirección ip nodo únicas autonómico, sin protocolo.
- `camelcomponentkafkabrokers: #` broker bootstrap de kafka Ejemplo: ohien-kafka-bootstrap.int.svc.cluster.local:9092
- `kafkassword: #` password de kafka
- `kafkausername: #` user de kafka
- `unicasclientkeystorepassword: #` password del client keystore de UNICAS
- `oauth2clientid: #` client id
- `oauth2clientsecret: #` client secret
- `namespace: #` namespace
- `gateway_type: #` gateway del entorno: `k8s_gateway` o `istio`

## 3. Ejecutar comando de instalación.

A continuación, para instalar el Helm, ejecutamos el comando siguiente:

**helm install <nombre de la release> <ruta del chart> -n <namespace> --set-file certsecret.content=<p12certificado>.**

Por ejemplo, en el namespace "int", situando el certificado "cliente-nc-pre.p12" en un directorio llamado "certificado" junto al chart, ejecutaríamos el siguiente comando para instalar el HELM:

**helm install properties ./properties-chart/ -n int --set-file certsecret.content=certificado/client-nc-pre.p12**

```
sh-4.2$ helm install properties ./properties-chart -n int --set-file certsecret.content=certificado/client-nc-pre.p12
NAME: properties
LAST DEPLOYED: Tue Mar 3 15:36:24 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todos los recursos enumerados al inicio de este epígrafe con los valores indicados en values.yaml.

## 2.1.2 VERIFICACIÓN DE LA INSTALACIÓN

Para confirmar que se han creado correctamente los recursos, se pueden ejecutar los siguientes comandos:

1. Comprobar creación de los configmaps:

Para consultar la existencia de los diferentes configmaps se ejecuta el comando siguiente, indicando el nombre del configmap: **kubectl get cm <nombreconfigmap> -n <namespace>**. Por ejemplo, para "na-properties" aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm na-properties -n int
NAME          DATA      AGE
na-properties  1          4m45s
```

Si además queremos ver su contenido para comprobar el valor de alguna propiedad, podemos utilizar el siguiente comando: **kubectl describe cm <nombreconfigmap> -n <namespace>**. Por ejemplo para este mismo configmap:

```
sh-4.2$ kubectl describe cm na-properties -n int
Name:          na-properties
Namespace:     int
Labels:        app.kubernetes.io/managed-by=Helm
               camel.apache.org/integration=na-all
Annotations:   meta.helm.sh/release-name: properties
               meta.helm.sh/release-namespace: int

Data
====
na-properties.properties:
----
identifier.allowed.systems=["urn:cite:80724000015"]
na-outbound-processor.url=http://na-outbound-processor
hdr.url=http://ohhdr-back:8080
mpi.url=http://ohmpi-back:8080
sso.url=http://ohsso:8080
ont.url=http://ohont-back:8080
aut.url=http://ohaut-back:8080
contexto.aut=/hnaut
contexto.aut.fhir=/hnaut/fhir
contexto.conf=/hnconf
```

2. Comprobar creación de los secrets:

Para consultar la existencia de los diferentes secrets se ejecuta el comando siguiente, indicando el nombre del secret: **kubectl get secret <nombreconfigmap> -n <namespace>**. Por ejemplo, para "na-secrets" aparecerá lo siguiente:

```
sh-4.2$ kubectl get secret na-secrets -n int
NAME          TYPE      DATA      AGE
na-secrets    Opaque    1          7m28s
```

3. Comprobar creación del httproute o virtual service:

Este recurso depende de la tipología de gateway indicada en values. Si se quiere comprobar la existencia del virtual service habrá que ejecutar el siguiente comando: **kubectl get virtualservice nc-na-virtualservice -n <namespace>**.

Por el contrario, si lo que se ha creado es un httproute se hace utilizando el comando: **kubectl get httproute nc-na-http-route -n <namespace>**. Por ejemplo en "int":

```
sh-4.2$ kubectl get httproute nc-na-http-route -n int
NAME                HOSTNAMES  AGE
nc-na-http-route    []         10m
```

### 2.1.3 ACTUALIZACIÓN

En caso de necesitar actualizar el Helm por modificación de uno o varios recursos, es necesario sustituir el directorio "properties-chart" existente por el nuevo. Si solo se quiere actualizar por modificación de alguna variable de values.yaml con modificar este archivo del chart será suficiente. En cualquiera caso, tras el cambio, hay que ejecutar el siguiente comando para que se efectúe la modificación: **helm upgrade properties ./properties-chart/ -n <namespace> --set-file certsecret.content=<p12certificado>**. Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade properties ./properties-chart -n int --set-file certsecret.content=certificado/client-nc-pre.p12
Release "properties" has been upgraded. Happy Helming!
NAME: properties
LAST DEPLOYED: Tue Mar  3 15:35:40 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

Es fundamental indicar en el comando de actualización dónde se encuentra el certificado, aunque no se haya modificado, en caso contrario se sobrescribirá con contenido vacío y la conexión entre NA y NC dejará de funcionar.

### 2.1.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, para por ejemplo eliminar los recursos generados, se ejecuta el siguiente comando: **helm uninstall properties -n <namespace>**.

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

Al ejecutar el comando, Helm elimina los recursos que se había generado:

- ConfigMaps definidos en el Helm.
- Secrets definidos en el Helm.
- Httproute o VirtualService que se hubiera creado con el Helm.

### 3 Microservicio na-health-check

Este microservicio devuelve el estado del Nodo Autónomo de cara al Nodo Central, teniendo en cuenta los componentes principales:

- MPI
- AUT
- HDR
- CONF
- ONT

Solo puede consultarse con método GET, si el componente responde el status será UP, si no responde correctamente será DOWN. Si uno o más componentes responden DOWN, el status general será DOWN también. Solo es UP si todas responden.

De cara a Nodo Central se implementa la llamada al microservicio a través de httproute o virtualservice, depende de la tipología de gateway del nodo autónomo. El recurso que expone el endpoint se genera junto con los recursos comunes del entorno con el Helm Chart properties-chart (nc-na-http-route o nc-na-virtualservice). De cara a NC el endpoint se expone en la ruta /ie/healthcheck. Es el único servicio expuesto a NC que no requiere autenticación con token.

#### 3.1 ENDPOINTS DISPONIBLES

##### GET /

Permite al NC consultar el estado de los componentes principales del NA.

##### Entrada

No aplica.

##### Salida

Estructura del mensaje de salida, con UP o DOWN en status según el estado de la aplicación:

```
{
  "status": "< UP | DOWN >",
  "checks": [
    {
      "name": "MPI",
      "status": "< UP | DOWN >"
    },
    {
      "name": "AUT",
      "status": "< UP | DOWN >"
    },
    {
      "name": "HDR",
      "status": "< UP | DOWN >"
    },
    {
      "name": "CONF",
      "status": "< UP | DOWN >"
    },
    {
      "name": "ONT",
      "status": "< UP | DOWN >"
    }
  ]
}
```

## 3.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **na-health-check**, 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 **na-health-check-chart**.

### 3.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **na-health-check** utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio **na-health-check-chart** es la siguiente:

```
sh-4.2$ tree na-health-check-chart/
na-health-check-chart/
├── Chart.yaml
├── files
│   ├── NAHealthCheck.java
│   └── templatesqute
│       ├── check.qute.json
│       └── response.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 9 files
```

#### 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.

```
integration:
  name: na-health-check
  namespace: # namespace en el que se quiere desplegar la integración
  dependencies:
    - "camel:platform-http"
    - "camel:direct"
  sources:
```

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

### 3. Ejecutar comando de instalación.

A continuación, será necesario ejecutar el comando siguiente: **helm install na-health-check ./na-health-check-chart -n <namespace>**.

En el caso de la integración na-health-check en el namespace int: **helm install na-health-check ./na-health-check-chart -n int**.

```
sh-4.2$ helm install na-health-check ./na-health-check-chart -n int
NAME: na-health-check
LAST DEPLOYED: Tue Mar 3 16:31:01 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration na-health-check
- Configmap na-health-check-qute

### 3.2.2 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 na-health-check -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration na-health-check -n int
NAME          PHASE   READY   RUNTIME PROVIDER  RUNTIME VERSION  CATALOG VERSION  KIT              REPLICAS
na-health-check  Running True    plain-quarkus     3.30.2           3.30.2           kit-d6fikqkrfbb7hvbvpdj0  1
```

En este caso, que también se crea el configmap "<nombre>", se puede comprobar su existencia en el namespace con el siguiente comando: **kubectl get cm na-health-check-qute -n <namespace>**. Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm na-health-check-qute -n int
NAME                DATA  AGE
na-health-check-qute  2      64s
```

2. Consultar los logs del microservicio:

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

```
sh-4.2$ kamel log na-health-check -n int
Integration 'na-health-check' is now running. Showing log ...
[1] Monitoring pod na-health-check-7b87c8fdd6-dsnrx
[1] 2026-03-03 15:37:22,556 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-03 15:37:22,634 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-03 15:37:22,635 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-03 15:37:22,738 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-03 15:37:22,738 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = classpath:routes/NAHealthCheck.java
[1] 2026-03-03 15:37:22,738 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-properties
[1] 2026-03-03 15:37:23,039 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] recipient.list = direct:mpi,direct:aut,direct:hdr,direct:conf,direct:ont
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] unicas.base.url = https://integracio.unicas.salut.intranet.gencat.cat
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.mpi = /ispob
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] healthcheck.subdomain = /healthcheck
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.aut = /hnaut
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.hdr = /hnsrver
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.conf = /hnconf
[1] 2026-03-03 15:37:23,047 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.ont = /hncat
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Routes startup (total:7)
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started Ruta Principal (platform-http:///)
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started Ruta Health Check MPI (direct://mpi)
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started Ruta Health Check AUT (direct://aut)
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started Ruta Health Check HDR (direct://hdr)
[1] 2026-03-03 15:37:23,048 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started Ruta Health Check CONF (direct://conf)
```

### 3.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración por, por ejemplo, un cambio de alcance del microservicio o para añadir nuevos endpoints, es necesario sustituir el directorio "na-health-check-chart" existente por el nuevo. Si solo se quiere modificar algún valor de "values.yaml" basta con editar ese archivo en el chart existente. En cualquier caso, para efectuar la modificación, hay que ejecutar el siguiente comando: **helm upgrade na-health-check ./na-health-check-chart -n <namespace>**. Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade na-health-check ./na-health-check-chart -n int
Release "na-health-check" has been upgraded. Happy Helming!
NAME: na-health-check
LAST DEPLOYED: Tue Mar 3 16:40:07 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
sh-4.2$
```

### 3.2.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo, porque se haya indicado algún valor erróneo en values.yaml y quiera eliminarse la release e instalar de nuevo, se ejecuta el siguiente comando: **helm uninstall na-health-check -n <namespace>**.

```
sh-4.2$ helm uninstall na-health-check -n int
release "na-health-check" 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.
- El o los pods ejecutados por la integración.
- El ConfigMap definido dentro del Helm chart.

## 4 Microservicio validate-patient

Este microservicio valida si el paciente puede o no ser enrolado cuando se le intenta abrir una instancia de proceso P\_UNICAS. Si no cumple las condiciones y es la primera que vez que intenta ser enrolado, se realizará un rollback del paciente en el repositorio. El rollback también puede utilizarse de forma independiente para casos como por ejemplo que no se acepte el consentimiento inicial para entrar en UNICAS.

Para decidir si el paciente puede ser enrolado se valida en el recurso Patient FHIR del paciente:

- El campo Active es true.
- La extensión estado-enrolamiento NO es true, lo que indica que no está enrolado actualmente.
- Tiene menos de 18 años.
- El paciente no está anonimizado en el nodo.

Para decidir si el paciente necesita rollback se comprueba:

- Si el paciente tiene extensión momento (aunque sea de salida) en el recurso, no se realiza rollback. En caso contrario, sí se realiza.

El rollback consiste en realizar DELETE del recurso FHIR Patient tanto en MPI como en HDR.

### 4.1 ENDPOINTS DISPONIBLES

#### GET /validate/{idPacienteUnicas}

Comprueba si el paciente cuyo id FHIR coincide con el "idPacienteUnicas" recibido en la ruta puede ser enrolado en UNICAS.

##### Entrada

Authentication: Bearer token

Accept: application/json

Content-Type: application/json

##### Salida

Estructura mensaje salida:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information/error/exception",
      "code": "informational/conflict/exception",
      "details": {
        "text": "Descripción detallada"
      }
    }
  ]
}
```

Posibles respuestas:

Código	Descripción
200	El paciente cumple las condiciones para ser enrolado.
409	El paciente NO cumple las condiciones para ser enrolado.

Código	Descripción
500	Error procesando solicitud del paciente.

### GET /rollback/{idPacienteUnicas}

Comprueba si el paciente cuyo id FHIR coincide con el "idPacienteUnicas" recibido en la ruta ha sido enrolado previamente en ÚNICAS y, si no lo ha sido, realiza el rollback del paciente del repositorio FHIR.

#### Entrada

Authentication: Bearer token

Accept: application/json

Content-Type: application/json

#### Salida

Estructura mensaje salida:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "information/error/exception",
      "code": "informational/conflict/exception",
      "details": {
        "text": "Descripción detallada"
      }
    }
  ]
}
```

Posibles respuestas:

Código	Descripción
200	OK
204	El paciente no requiere rollback.

## 4.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **validate-patient**, 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 "validate-patient-oute".

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

## 4.2.1 INSTALACIÓN

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

### 1. Preparación del chart

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

```

.
├── Chart.yaml
├── files
│   ├── PatientUNICASValidateService.java
│   └── templatesqute
│       └── response.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 8 files
    
```

### 2. Configuración Previa

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

```

integration:
  name: validate-patient
  namespace: # namespace en el que se quiere desplegar la integración
  dependencies:
    - "camel:platform-http"
    - "camel:jq"
    - "camel:httn"
    
```

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

### 3. Ejecutar comando de instalación.

Para ejecutar la instalación de la integración, se utiliza el comando: **helm install validate-patient ./validate-patient-chart -n <namespace>**.

Sustituyendo "<namespace>" por el namespace en el que se está instalando la integración, por ejemplo, para un namespace llamado "int":

```

sh-4.2$ helm install validate-patient ./validate-patient-chart -n int
NAME: validate-patient
LAST DEPLOYED: Mon Mar 2 15:43:53 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
    
```

Esto creará todo lo necesario para que la integración funcione:

- Integration validate-patient.
- Configmap validate-patient-quete.

## 4.2.2 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 validate-patient -n <namespace>` podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration validate-patient -n int
NAME          PHASE    READY    RUNTIME PROVIDER    RUNTIME VERSION    CATALOG VERSION    KIT                                REPLICAS
validate-patient  Running  True     plain-quarkus       3.28.2             3.28.2             kit-d3npfk33r3c6m2a9gpg         1
sh-4.2$
```

En este caso, que también se debe crear el configmap "validate-patient-quete", se comprueba su existencia en el namespace con el siguiente comando: `kubectl get cm validate-patient-quete -n <namespace>`. Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm validate-patient-quete -n int
NAME          DATA    AGE
validate-patient-quete  1        112s
```

2. Consultar logs del microservicio:

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

```
sh-4.2$ kamel log validate-patient -n int
Integration 'validate-patient' is now running. Showing log ...
[1] Monitoring pod validate-patient-7978945f7c-hgprn
[1] 2026-02-16 07:38:58,340 WARN [io.qua.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-02-16 07:39:06,842 INFO [org.apa.cam.qua.cor.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.27.0 is starting
[1] 2026-02-16 07:39:06,843 INFO [org.apa.cam.mai.MainSupport] (main) Apache Camel (Main) 4.14.0 is starting
[1] 2026-02-16 07:39:17,750 INFO [org.apa.cam.mai.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-02-16 07:39:17,750 INFO [org.apa.cam.mai.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:/etc/camel/sources/**
[1] 2026-02-16 07:39:17,750 INFO [org.apa.cam.mai.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-common-properties
[1] 2026-02-16 07:39:17,750 INFO [org.apa.cam.mai.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-02-16 07:39:17,751 INFO [org.apa.cam.mai.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.context.restConfiguration.component = platform-http
[1] 2026-02-16 07:39:19,946 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Apache Camel 4.14.0 (camel-1) is starting
[1] 2026-02-16 07:39:20,541 INFO [org.apa.cam.mai.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-02-16 07:39:20,541 INFO [org.apa.cam.mai.BaseMainSupport] (main) [OverrideProperties] unicas.base.url = https://integracio.unicas.salut.intranet.unicas.cat
[1] 2026-02-16 07:39:20,542 INFO [org.apa.cam.mai.BaseMainSupport] (main) [OverrideProperties] contexto.hdr.fhir = ehrserver/fhir
[1] 2026-02-16 07:39:20,542 INFO [org.apa.cam.mai.BaseMainSupport] (main) [OverrideProperties] contexto.mpi.fhir = /iapob/fhir
[1] 2026-02-16 07:39:20,542 INFO [org.apa.cam.mai.BaseMainSupport] (main) [OverrideProperties] extension.momento-unicas.url = https://unicas-fhir.sanidad.gob.es/StructureDefinition/MomentosUnicas
[1] 2026-02-16 07:39:20,545 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Routes startup (total:6 rest-dsl:2)
[1] 2026-02-16 07:39:20,547 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Started FilteredPatientSearchRoute (direct://filteredPatientSearch)
[1] 2026-02-16 07:39:20,547 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Started PatientSearchRoute (direct://patientSearch)
[1] 2026-02-16 07:39:20,547 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Started DeletePatientFromMpiRoute (direct://deletePatientFromMpi)
[1] 2026-02-16 07:39:20,547 INFO [org.apa.cam.imp.eng.AbstractCamelContext] (main) Started DeletePatientFromHdrRoute (direct://deletePatientFromHdr)
```

## 4.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración, por ejemplo, por un cambio en los datos que deben comprobarse del paciente para validarlo, es necesario sustituir el directorio "validate-patient-chart" existente por el nuevo. Si se necesita cambiar algún valor de "values.yaml", es suficiente

con cambiar este archivo en el chart existente. En cualquier caso, para efectuar el cambio, es necesario ejecutar el siguiente comando: **helm upgrade validate-patient ./validate-patient-chart -n <namespace>**. Aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade validate-patient ./validate-patient-chart -n int
Release "validate-patient" has been upgraded. Happy Helming!
NAME: validate-patient
LAST DEPLOYED: Mon Mar  2 15:47:56 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

#### 4.2.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo, por un error en "values.yaml" por el que se prefiere empezar una release de cero volviendo a instalar en lugar de actualizar, se puede hacer ejecutando el siguiente comando: **helm uninstall validate-patient -n <namespace>**.

```
sh-4.2$ helm uninstall validate-patient -n int
release "validate-patient" 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.
- El pod o pods asociados a la integración.
- El ConfigMap definido dentro del Helm chart.

## 5 Microservicio salida-unicas

Este microservicio gestiona la salida de pacientes del proceso ÚNICAS. Se utiliza desde el propio proceso ÚNICAS (P\_UNICAS).

Procesa las solicitudes de salida que vienen del proceso, actualiza los datos del paciente en MPI según el motivo de salida y en caso de modificación del paciente envía notificación de modificación core a NC para que informe del cambio al resto de nodos y crea un mensaje en el topic de kafka que informa de las modificaciones core de pacientes a los HIS. Las opciones de salida y cambios asociados a cada una de ellas son:

- Si la salida es por traslado (respuesta en el formulario de salida con code 06), no se realiza ninguna modificación sobre el paciente.
- Si la salida es por revocación del consentimiento (respuesta en el formulario de salida con code 01), se modifica el momento del paciente, el estado de enrolamiento y su etiqueta de seguridad sobre el consentimiento.
- En cualquier otro caso, solo se modifica el momento y el estado de enrolamiento.

La información que necesita la recibe en un recurso Parameters que contiene dos parámetros: id (idPacienteUnicas) y formulario (respuesta del formulario Salida Unicas en formato QuestionnaireResponse).

### 5.1 ENDPOINTS DISPONIBLES

#### POST /salida

Gestiona la salida del paciente enrolado en ÚNICAS en función del motivo.

##### Entrada

Authentication: Bearer token

Accept: application/json

Content-Type: application/json

Cuerpo de la solicitud:

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "id",
      "valueString": "<idPacienteUnicas>"
    },
    {
      "name": "formulario",
      "resource": {
        "resourceType": "QuestionnaireResponse",
        "id": "BPMQRP-3709",
        "meta": {
          "tag": [
            {
              "system": "module",
              "code": "ohbpm"
            },
            {
              "system": "ohbpm.task.formKey",
              "code": "547201120",
              "display": "Salida de paciente ÚNICAS"
            }
          ]
        }
      }
    }
  ]
}
```

```

    },
    "basedOn": [
      {
        "reference": "CarePlan/BPMCP-3683"
      }
    ],
    "subject": {
      "reference": "Patient/AC17605460445944519"
    },
    "authored": "2025-11-18T13:19:25.838Z",
    "author": {
      "reference": "Practitioner/us_admin",
      "display": "Usuario administrador"
    },
    "item": [
      {
        "linkId": "5",
        "definition": "process_end_code",
        "text": "Motivo de la salida",
        "answer": [
          {
            "valueCoding": {
              "code": "06",
              "display": "revocacion prueba"
            }
          }
        ]
      }
    ]
  }
}

```

### Salida

Estructura mensaje salida:

```

{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "<information | error | exception>",
      "code": "<informational | conflict | exception>",
      "details": {
        "text": "Descripción detallada"
      }
    }
  ]
}

```

Posibles respuestas:

Código	Descripción
200	Salida por traslado procesada correctamente.
200	Salida procesada correctamente y reflejada en la información de paciente en UNICAS.
500	Mensaje de excepción.

## 5.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **salida-unicas**, 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 "salida-quete".

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

### 5.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración salida-unicas utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio salida-unicas-chart es la siguiente:

```
salida-unicas-chart/
├── Chart.yaml
├── files
│   ├── SalidaUNICASService.java
│   └── templatesqute
│       └── response.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 8 files
```

#### 2. Configuración Previa

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

```
1 integration:
2   name: salida-unicas
3   namespace: # namespace
4   dependencies:
5     - "camel:platform-http"
6     - "camel:jq"
7     - "camel:http"
8     - "camel:direct"
```

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

### 3. Ejecutar comando de instalación.

Para ejecutar la instalación de la integración, se utiliza el comando: **helm install salida-unicas ./salida-unicas-chart -n <namespace>**.

Sustituyendo "<namespace>" por el namespace en el que se está instalando la integración, por ejemplo, para un namespace llamado "int":

```
sh-4.2$ helm install salida-unicas ./salida-unicas-chart/ -n int
NAME: salida-unicas
LAST DEPLOYED: Mon Mar 2 17:38:47 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration salida-unicas.
- Configmap salida-qute.

## 5.2.2 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 salida-unicas -n <namespace>** podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration salida-unicas -n int
NAME          PHASE  READY  RUNTIME PROVIDER  RUNTIME VERSION  CATALOG VERSION  KIT              REPLICAS
salida-unicas  Running  False  plain-quarkus    3.30.3           3.30.3           kit-d51egipdb11619mitpfg  1
```

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

```
sh-4.2$ kubectl get cm salida-qute -n int
NAME          DATA  AGE
salida-qute   1      73s
sh-4.2$
```

2. Consultar logs del microservicio:

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

```
sh-4.2$ helm log salida-unicas -n int
Integration 'salida-unicas' is now running. Showing log ...
[1] 2026-03-02 17:40:00,198 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-02 17:40:02,600 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-02 17:40:02,601 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-02 17:40:16,808 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-02 17:40:16,809 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:/etc/camel/sources/**
[1] 2026-03-02 17:40:16,809 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-properties
[1] 2026-03-02 17:40:16,809 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-03-02 17:40:16,809 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.context.restConfigurationComponent = platform-http
[1] 2026-03-02 17:40:22,102 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-02 17:40:22,804 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholders summary
[1] 2026-03-02 17:40:22,804 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] extension.momento-unicas.url = https://unicas-fhir.sanidad.gob.es/StructureDefinition/MomentosUnicas
[1] 2026-03-02 17:40:22,806 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] extension.estado-enrolamiento.url = https://unicas-fhir.sanidad.gob.es/StructureDefinition/EstadoEnrolamientoPaciente
[1] 2026-03-02 17:40:22,896 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] meta.security.confidencialidad.co
```

## 5.2.3 ACTUALIZACIÓN

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

```
sh-4.2$ helm upgrade salida-unicas ./salida-unicas-chart -n int
Release "salida-unicas" has been upgraded. Happy Helming!
NAME: salida-unicas
LAST DEPLOYED: Mon Mar 2 17:37:17 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

## 5.2.4 DESINSTALACIÓN

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

```
sh-4.2$ helm uninstall salida-unicas -n int
release "salida-unicas" 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.
- El pod o pods asociados a la integración.
- El ConfigMap definido dentro del Helm chart.

## 6 Microservicio na-outbound-processor

Este microservicio integra el nodo autonómico con el nodo central. Además, crea los mensajes en los topic Kafka que permiten mantener a los HIS informados sobre altas de paciente y modificaciones core de pacientes UNICAS. Los endpoints que se exponen en esta integración solo pueden consumirse desde el resto de microservicios desplegados en el mismo namespace. Este microservicio crea el contexto SSL para que la conexión entre NA y NC pueda darse de forma segura. A continuación, se describen todos los endpoint disponibles en el namespace en el que se despliegue la integración.

### 6.1 ENDPOINTS DISPONIBLES

#### POST /nc/modificacioncore

Informa modificaciones core a NC. Al utilizar esta integración, si todo ha ido bien, se ejecuta también la ruta "notificacionmodcore" para informar a los HIS a través de Kafka. Se precisa solo uno de los siguientes datos de paciente para crear la notificación: CipSns, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo. Se permiten las tres opciones para que el endpoint sea versátil y pueda utilizarse desde diferentes microservicios, independientemente de la información que tengan del paciente.

#### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
        "resourceType": "Patient",
        "id": "ACxxxxxxxxxxx",
        [...]
      }
    }
  ]
}
```

#### Salida

En la salida se devuelve la respuesta de NC.

Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, la llamada a NC no llegará a realizarse y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}
```

## POST /nc/modificaciongenerica

Informa modificaciones genéricas a NC. No se precisa saber exactamente qué dato ha sido el modificado, sabiendo el paciente se enviará la información que necesita el NC para registrar la modificación genérica. Se precisa solo uno de los siguientes datos de paciente para crear la notificación de modificación: CipSns, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo. Es importante que solo se envíe uno de ellos en la entrada.

### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
        "resourceType": "Patient",
        "id": "ACxxxxxxxxxxx",
        [...]
      }
    }
  ]
}
```

### Salida

En la salida se devuelve la respuesta de NC.

Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, la llamada a NC no llegará a realizarse y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}
```

## GET /nc/tsi

Valida el cipsns del paciente contra TSI a través del nodo central. Se precisa solo uno de los siguientes datos de paciente para crear la notificación: CipSns o idPacienteUnicas. El dato necesario para la validación es CIPSNS, por lo que lo óptimo es enviar este dato si se tiene. Si no se tiene y se envía el idPacienteUnicas, la integración buscará el CIPSNS en el repositorio.

### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

### Salida

En la salida se devuelve la respuesta de NC.

Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, la llamada a NC no llegará a realizarse y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}
```

## POST /nc/altapaciente

Informa al NC en caso de alta de paciente. Una vez informado NC, se envía la notificación al topic de kafka destinado a informar a los HIS de este alta de paciente haciendo uso de la ruta asociada al endpoint "his/altapaciente". Se precisa solo uno de los siguientes datos de paciente para ejecutar la integración: CIPSNS, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo.

### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
        "resourceType": "Patient",
        "id": "ACxxxxxxxxxx",
        [...]
      }
    }
  ]
}
```

### Salida

En la salida se devuelve la respuesta de NC.

Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, la llamada a NC no llegará a realizarse y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}
```

## POST /nc/presenciapaciente

Informa al NC sobre índice de presencia del paciente. Se precisa solo uno de los siguientes datos, aunque lo más óptimo es mandar directamente el CIPSNS: CIPSNS, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo.

### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
        "resourceType": "Patient",
        "id": "ACxxxxxxxxxxx",
        [...]
      }
    }
  ]
}
```

### Salida

En la salida se devuelve la respuesta de NC.

Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, la llamada a NC no llegará a realizarse y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```
{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}
```

## GET /nc/getallnotifications

Solicita a NC todas las notificaciones pendientes de ser leídas hasta ese momento. No se precisan datos específicos de entrada, solicita todas las notificaciones para ese nodo en general.

### Entrada

No aplica.

### Salida

En la salida se devuelve la respuesta de NC.

## GET /nc/getnotification/{idNotificacion}

Solicita a NC el detalle de una notificación a partir de su id. El identificador de la notificación se indica en la ruta de la solicitud, sustituyendo "{idNotificacion}".

## Entrada

No aplica.

## Salida

En la salida se devuelve la respuesta de NC.

## POST /nc/getinfopatient

Solicita información completa de paciente a NC, que va a consultarla a todos los NA o a los que se indiquen en header "nodos", tal y como ha definido NC en su endpoint. En esta búsqueda es indispensable tener desde el inicio el id de transacción, por lo que se delega su creación a la integración que utiliza este servicio, de forma que pueda controlar la trazabilidad de esta llamada y las siguientes. En el cuerpo de la solicitud se indican los parámetros de búsqueda en un Bundle de tipo batch, tal y como lo precisa NC para hacer la búsqueda.

### Entrada

id\_transaccion: <uuid>

X-NA-TARGET: hdr

Estructura cuerpo:

```
{
  "resourceType": "Bundle",
  "type": "batch",
  "entry": [
    {
      "resource": {
        "resourceType": "Parameters",
        "parameter": [
          {
            "name": "patient",
            "valueString": "<idPacienteUnicas>"
          }
        ]
      }
    }
  ],
  "request": {
    "method": "POST",
    "url": "/Patient/$everything"
  }
}
```

### Salida

En la salida se devuelve la respuesta de NC.

## POST /nc/nextinfopatient

Este servicio complementa a la llamada "nc/getinfopatient". Solicita al NC que continúe con la búsqueda de recursos asociados al paciente por los distintos nodos. Permite "gestionar" la paginación utilizando el id\_transacción. Es muy importante que el id\_transaccion sea el mismo en todas las consultas asociadas a una misma búsqueda, por lo que como en el caso anterior este header se debe recibir como entrada del endpoint, se delega su gestión a la integración solicitante. El header "id\_transaccion" se mandará a NC tal y como se reciba. El resto de headers sí se generarán (id\_peticion, code\_agent).

### Entrada

id\_transaccion: <uuid>

## Salida

En la salida se devuelve la respuesta de NC.

### GET /nc/getfhirinfo?{parametrosdebusqueda}

Solicita información FHIR a NC. Los parámetros de la búsqueda se indican en la misma URL, siguiendo la estructura de la búsqueda en FHIR. En este caso también se pueden indicar los nodos que se quieren consultar: en el header "nodos" se puede indicar si se quiere consultar solo en un nodo concreto (ej: ES-CT), en una lista de nodos (ej: ES-CT, ES-AN) o en todos (ALL).

#### Entrada

nodos: <nodos> (opcional).

#### Salida

En la salida se devuelve la respuesta de NC.

### GET /nc/getalldocs

Solicita a NC todos los documentos que el nodo puede consultar. No se precisan datos específicos de entrada.

#### Entrada

No aplica.

#### Salida

En la salida se devuelve la respuesta de NC.

### GET /nc/getdoc/{idDocumento}

Solicita a NC el detalle de un documento a partir de su id. El identificador del documento se indica en la ruta de la solicitud, sustituyendo "{idDocumento}".

#### Entrada

No aplica.

#### Salida

En la salida se devuelve la respuesta de NC.

### POST /his/notificacionmodcore

Envía paciente core al topic kafka "modificacion-core-paciente-unicas". Cuando se informa al NC una modificación core se llama directamente a esta ruta internamente. Para ejecutarla desde fuera se puede hacer un POST al endpoint indicado. Se precisa uno de los siguientes datos: CIPSNS, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo.

#### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```
{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
```

```

    "resourceType": "Patient",
    "id": "ACxxxxxxxxxx",
    [...]
  }
}
]
}

```

### Salida

En la salida se devuelve el recurso enviado al topic. Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, el recurso no llega a insertarse en la cola kafka y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```

{
  "resourceType": "OperationOutcome",
  "issue": [
    {
      "severity": "error",
      "code": "exception",
      "diagnostics": "<Descripción del error>"
    }
  ]
}

```

### POST /his/altapaciente

Envía paciente core al topic kafka "alta-paciente-unicas". Alta paciente llama directamente a esta ruta internamente cuando se ejecuta. Para ejecutarla desde fuera se puede hacer un POST al endpoint indicado. Se precisa uno de los siguientes datos: CIPSNS, idPacienteUnicas o body con recurso Parameters con un parámetro llamado "Patient" que incluya el resource Patient FHIR completo.

#### Entrada

cipsns: <CipSns del paciente> | idPacienteUnicas: <id Paciente Unicas>

Estructura cuerpo (solo si no se ha informado alguno de los headers anteriores):

```

{
  "resourceType": "Parameters",
  "parameter": [
    {
      "name": "Patient",
      "resource": {
        "resourceType": "Patient",
        "id": "ACxxxxxxxxxx",
        [...]
      }
    }
  ]
}

```

#### Salida

En la salida se devuelve el recurso enviado al topic. Si falta información del paciente, no se encuentra el paciente en MPI o se ha enviado más de un elemento de entrada de los definidos, el recurso no llega a insertarse en la cola kafka y se recibirá un mensaje de excepción con código 500 y la siguiente estructura:

```

{
  "resourceType": "OperationOutcome",
  "issue": [

```

```
{
  "severity": "error",
  "code": "exception",
  "diagnostics": "<Descripción del error>"
}
```

## 6.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **na-outbound-processor**, 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 `na-outbound-processor-queue`, configmap con las plantillas `queue` que se utilizan en la integración.

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

### 6.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración `na-outbound-processor` utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio `na-outbound-processor-chart` es la siguiente:

```

├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── naOutboundProcessor.java
│   ├── SSLAuthentication.java
│   └── templatesqute
│       ├── alta-paciente.qute.json
│       ├── indice-presencia.qute.json
│       ├── mod-core.qute.json
│       ├── mod-generica.qute.json
│       ├── paciente-core.qute.json
│       └── response.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 15 files
    
```

## 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.

```

integration:
  name: na-outbound-processor
  namespace: # namespace
  dependencies:
    - "camel:core"
    - "camel:platform-http"
    
```

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

## 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install na-outbound-processor ./<ruta del chart> -n <namespace>**.

En el caso de la integración na-outbound-processor en el namespace int → **helm install na-outbound-processor ./na-outbound-processor-chart -n int**.

```

sh-4.2$ helm install na-outbound-processor ./na-outbound-processor-chart -n int
NAME: na-outbound-processor
LAST DEPLOYED: Tue Mar  3 14:44:18 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
    
```

Esto creará todo lo necesario para que la integración funcione:

- Integration na-outbound-processor.
- ConfigMap na-outbound-processor-quete.

## 6.2.2 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 na-outbound-processor -n <namespace>` podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration na-outbound-processor -n int
NAME                                PHASE    READY    RUNTIME PROVIDER    RUNTIME VERSION    CATALOG VERSION    KIT                                REPLICAS
na-outbound-processor              Running  True     plain-quarkus       3.30.2             3.30.2             kit-d4tb7t9db11619mitoeg        1
```

En este caso, que también se crea el configmap "na-outbound-processor-quete", se puede comprobar su existencia en el namespace con el siguiente comando: `kubectl get cm na-outbound-processor-quete -n <namespace>`. Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm na-outbound-processor-quete -n int
NAME                                DATA    AGE
na-outbound-processor-quete         6        7d21h
```

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: `kamel logs na-outbound-processor -n <namespace>`. Ejemplo de esta integración en "int":

```
sh-4.2$ kamel log na-outbound-processor -n int
Integration 'na-outbound-processor' is now running. Showing log ...
[1] 2026-03-03 14:45:30,517 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-03 14:45:32,617 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-03 14:45:32,619 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-03 14:45:53,625 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-03 14:45:53,713 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern=file:/etc/camel/sources/**
[1] 2026-03-03 14:45:53,715 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation=/etc/camel/conf.d/_configmaps/na-properties,/etc/camel/conf.d/_configmaps/na-kafka-properties,/etc/camel/conf.d/_secrets/na-secrets,/etc/camel/conf.d/_secrets/na-kafka-secrets
[1] 2026-03-03 14:45:53,716 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled=true
[1] 2026-03-03 14:45:53,716 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.context.restConfiguration.component = platform-http
[1] 2026-03-03 14:46:09,413 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-03 14:46:11,020 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[1] 2026-03-03 14:46:13,019 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-03 14:46:13,020 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] nodo.unicas = ES-CT
[1] 2026-03-03 14:46:13,116 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] mpi.url = http://ohmpi-back:8080
[1] 2026-03-03 14:46:13,116 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.mpi.fhir = /ispob/fhir
[1] 2026-03-03 14:46:13,119 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] system.identifier.cipans = urn:ci:16.724.4.40
[1] 2026-03-03 14:46:13,120 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] meta.security.confidentiality.co
```

## 6.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración por un cambio en el diseño de la misma o una modificación en el alcance del microservicio, es necesario sustituir el directorio "na-outbound-processor-chart" existente por el nuevo y ejecutar el siguiente comando: `helm upgrade na-outbound-processor ./na-outbound-processor-chart -n <namespace>`. Nos aparecerá un mensaje como el siguiente al ejecutarlo:

```
sh-4.2$ helm upgrade na-outbound-processor ./na-outbound-processor-chart -n int
Release "na-outbound-processor" has been upgraded. Happy Helming!
NAME: na-outbound-processor
LAST DEPLOYED: Tue Mar 3 14:39:20 2026
NAMESPACE: int
STATUS: deployed
REVISION: 4
TEST SUITE: None
```

## 6.2.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, por ejemplo, porque la integración haya dejado de ser necesaria o se haya errado modificando los parámetros, se ejecuta el siguiente comando: **helm uninstall na-outbound-processor -n <namespace>**.

```
sh-4.2$ helm uninstall na-outbound-processor -n int
release "na-outbound-processor" 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.
- El pod o pods ejecutados por la integración.
- El ConfigMap definido dentro del Helm chart.

## 7 Microservicio indice-presencia-nc

Este microservicio consume los mensajes que llegan al topic de Kafka "nc-indice-presencia-topic". Esto tiene el objetivo de detectar la presencia de pacientes en nodos autonómicos en los que no había estado previamente, ya que el recurso FHIR Subscription asociado al topic detecta los Encounters de pacientes anonimizados en el nodo. En la integración se toma el idPacienteUnicas del mensaje recibido al consumir el mensaje del topic. Con este id se recupera el Patient actual en el nodo autonómico, obtenemos su nodo de alta y procedemos a obtener el paciente completo llamando a NC, de forma que nos devuelva el recurso Patient concretamente de su nodo de alta.

Una vez obtenido el Patient completo, la integración actualiza el paciente en MPI para que en el nodo autonómico en cuestión se cuente con toda la información del paciente. Por ello, este microservicio incluye la clase Authentication.java, ya que se necesita incluir Authorization Bearer **token** en la llamada a MPI. Una vez actualizado el recurso Patient en el nodo autonómico, este microservicio informa a NC sobre la presencia del paciente en el NA.

### 7.1 ENDPOINTS DISPONIBLES

No aplica.

### 7.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **indice-presencia-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. En este caso debe crearse el configmap "indice-presencia-nc".

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

#### 7.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración indice-presencia-nc utilizando Helm.

##### 1. Preparación del chart

Verificar que la estructura del directorio indice-presencia-nc-chart es la siguiente:

```

indice-presencia-nc-chart/
├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── IndicePresenciaNc.java
│   └── templatesqute
│       └── bundle-searchset.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 9 files
    
```

## 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.

```

integration:
  name: indice-presencia-nc
  namespace: # namespace
  dependencies:
    - "camel:jq"
    - "camel:direct"
    - "camel:rest"
    
```

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

## 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install indice-presencia-nc ./indice-presencia-nc-chart -n <namespace> .**

En el caso de la integración indice-presencia-nc en el namespace int → **helm install indice-presencia-nc ./indice-presencia-nc-chart -n int .**

```

sh-4.2$ helm install indice-presencia-nc ./indice-presencia-nc-chart -n int
NAME: indice-presencia-nc
LAST DEPLOYED: Tue Mar 3 10:03:31 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
    
```

Esto creará todo lo necesario para que la integración funcione:

- Integration indice-presencia-nc.
- ConfigMap indice-presencia-qute.

## 7.2.2 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 indice-presencia-nc -n <namespace>` podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

```
sh-4.2$ kubectl get integration indice-presencia-nc -n int
NAME                PHASE    READY    RUNTIME PROVIDER    RUNTIME VERSION    CATALOG VERSION    KIT                REPLICAS
indice-presencia-nc Running  True     plain-quarkus       3.30.2             3.30.2             kit-d65iia971qc4tp1jvj20  1
```

En este caso, que también se crea el configmap "indice-presencia-quete", se puede comprobar su existencia en el namespace con el siguiente comando: `kubectl get cm indice-presencia-quete -n <namespace>`. Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm indice-presencia-quete -n int
NAME                DATA    AGE
indice-presencia-quete  1        89s
```

2. Consultar los logs del microservicio:

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

```
sh-4.2$ kamel log indice-presencia-nc -n int
Integration 'indice-presencia-nc' is now running. Showing log ...
[1] Monitoring pod indice-presencia-nc-569fd4cf-wcqr
[1] 2026-03-03 10:04:41,073 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-03 10:04:43,070 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-03 10:04:43,070 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-03 10:04:59,578 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-03 10:04:59,578 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:/etc/camel/sources/**
[1] 2026-03-03 10:04:59,580 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-properties,/etc/camel/conf.d/_configmaps/na-kafka-properties,/etc/camel/conf.d/_secrets/na-secrets,/etc/camel/conf.d/_secrets/na-kafka-secrets
[1] 2026-03-03 10:04:59,580 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-03-03 10:05:06,874 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-03 10:05:07,485 INFO [org.apache.camel.component.kafka.RafkaConsumer] (main) Starting Kafka consumer on topic: nc-indice-presencia-topi with breakOnFirstError: false
[1] 2026-03-03 10:05:07,686 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-03 10:05:07,686 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] sso.url = http://ohsso:8080
[1] 2026-03-03 10:05:07,686 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.sso = /auth
[1] 2026-03-03 10:05:07,686 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.request.body = grant_type=client_credentials&client_id=na-cliente-admin&client_secret=zJbpEVL6FfuzXMvwXplaxWifVPEtd5Rys&scope=offline_access
[1] 2026-03-03 10:05:07,687 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] na-outbound-processor.url = http://na-outbound-processor
[1] 2026-03-03 10:05:07,687 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] mpi.url = http://ohmpi-back:8080
[1] 2026-03-03 10:05:07,687 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.mpi.fhir = /isopb/fhir
[1] 2026-03-03 10:05:07,687 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.clientId = xxxxxx
[1] 2026-03-03 10:05:07,687 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Using 2 instances of same component class: org.apache.c...
```

## 7.2.3 ACTUALIZACIÓN

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

```
sh-4.2$ helm upgrade indice-presencia-nc ./indice-presencia-nc-chart -n int
Release "indice-presencia-nc" has been upgraded. Happy Helming!
NAME: indice-presencia-nc
LAST DEPLOYED: Tue Mar 3 10:02:49 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

## 7.2.4 DESINSTALACIÓN

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

```
sh-4.2$ helm uninstall indice-presencia-nc -n int
release "indice-presencia-nc" 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.
- El pod o pods ejecutados por la integración.
- El ConfigMap definido dentro del Helm chart.

## 8 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 Autenticación 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.

### 8.1 ENDPOINTS DISPONIBLES

No aplica.

## 8.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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.

### 8.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **careteam-mpi-creator** utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio **careteam-mpi-creator-chart** es la siguiente:

```

.
├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── CareTeamMPICreator.java
│   ├── na-creator-careteam.properties
│   └── templatesqute
│       ├── careteam.qute.json
│       └── response.qute.json
├── README.md
├── templates
│   ├── configmap-na-creator-careteam.yaml
│   ├── configmap-templates-qute.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 12 files
    
```

#### 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.

```
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 instale.

### 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install careteam-mpi-creator ./careteam-mpi-creator-chart -n <namespace>**.

Sustituyendo "<namespace>" por el namespace en el que se está instalando la integración, por ejemplo para un namespace llamado "int":

```
sh-4.2$ helm install careteam-mpi-creator ./careteam-mpi-creator-chart -n int
NAME: careteam-mpi-creator
LAST DEPLOYED: Tue Mar  3 07:41:10 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration careteam-mpi-creator.
- Configmap careteam-mpi-creator-queue.

## 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 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-queue", se comprueba su existencia en el namespace con el siguiente comando: **kubectl get cm careteam-mpi-creator-queue -n <namespace>**.

Aparecerá lo siguiente:

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

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":

```
sh-4.2$ kamel log careteam-mpi-creator -n int
Integration 'careteam-mpi-creator' is now running. Showing log ...
[1] Monitoring pod careteam-mpi-creator-df7785b9c-fqnrz
[1] 2026-03-03 07:42:19,305 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify
[1] 2026-03-03 07:42:21,514 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-03 07:42:21,598 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-03 07:42:37,708 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-03 07:42:37,708 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:etc/
[1] 2026-03-03 07:42:37,709 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/
[1] 2026-03-03 07:42:37,710 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-03-03 07:42:45,501 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-03 07:42:46,911 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[1] 2026-03-03 07:42:49,806 INFO [org.apache.camel.component.kafka.KafkaConsumer] (main) Starting Kafka consumer on topic: inject-patient-careteam-topic with
[1] 2026-03-03 07:42:50,102 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-03 07:42:50,102 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] equipounicas.system = http://unicas-fhir.s
[1] 2026-03-03 07:42:50,103 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] equipounicas.code = EC
[1] 2026-03-03 07:42:50,103 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] equipounicas.display = Equipo Clínico
[1] 2026-03-03 07:42:50,104 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] mpi.url = http://olmpi-back:8080
[1] 2026-03-03 07:42:50,104 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.mpi.fhir = /lspcb/fhir
[1] 2026-03-03 07:42:50,104 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] sso.url = http://ohsso:8080
[1] 2026-03-03 07:42:50,109 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.sso = /auth
[1] 2026-03-03 07:42:50,110 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.request.body = grant_type=client_cr
[1] 2026-03-03 07:42:50,110 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.clientId = xxxxxx
[1] 2026-03-03 07:42:50,110 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Using 2 instances of same component class: org.apache.camel.compo
[1] 2026-03-03 07:42:50,111 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Routes startup (total:7)
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started kafka-careteam-entrada-route (kafka://inject-patient-
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started careteam-mpi-creator-route (direct://careteam-mpi-cre
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started obtener-patient-name-display-route (direct://obtene
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started comprobar-existencia-careteam-route (direct://comprob
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started put-careteam-route (direct://put-careteam)
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started post-careteam-route (direct://post-careteam)
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started autenticacion (direct://autenticacion)
[1] 2026-03-03 07:42:50,112 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) started in 4s610ms (build:0ms init:
[1] 2026-03-03 07:42:50,199 INFO [org.apache.camel.component.kafka.KafkaPatchRecords] (Camel (camel-1) thread #1 - KafkaConsumer[inject-patient-careteam-topi
[1] 2026-03-03 07:42:52,501 INFO [org.apache.camel.component.kafka.KafkaConsumer.support.classic.AssignmentAdapterHelper] (Camel (camel-1) thread #1 - KafkaConsum
[1] 2026-03-03 07:42:52,502 INFO [org.apache.camel.component.kafka.KafkaPatchRecorder] (Camel (camel-1) thread #1 - KafkaConsumer[inject-patient-careteam-topi
```

### 8.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración por un cambio en el diseño de la misma o una modificación en el alcance del microservicio, es necesario sustituir el directorio "careteam-mpi-creator-chart" existente por el nuevo y 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
```

### 8.2.4 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.
- El ConfigMap definido dentro del Helm chart.

## 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 Autenticación 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 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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 creación del PATCH del Encounter.

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

## 9.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración inject-momento-encounter utilizando Helm.

### 1. Preparación del chart

Verificar que la estructura del directorio inject-momento-encounter-chart es la siguiente:

```

.
├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── InjectMomentoEncounter.java
│   └── templatesqute
│       ├── encounter-patch.qute.xml
│       └── operation-outcome.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 10 files
    
```

### 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.

```

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 instale.

### 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install inject-momento-encounter ./inject-momento-encounter-chart -n <namespace>**.

Sustituyendo "<namespace>" por el namespace en el que se está instalando la integración, por ejemplo, para un namespace llamado "int":

```
sh-4.2$ helm install inject-momento-encounter ./inject-momento-encounter-chart -n int
NAME: inject-momento-encounter
LAST DEPLOYED: Tue Mar 3 08:53:26 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration inject-momento-encounter.
- Configmap inject-momento-encounter-quete.

## 9.2.2 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 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-d4ql5e9db11619mitnkg  1
sh-4.2$
```

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

Aparecerá lo siguiente:

```
sh-4.2$ kubectl get cm inject-momento-encounter-quete -n int
NAME                DATA    AGE
inject-momento-encounter-quete  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":

```
sh-4.2$ kubectl log inject-momento-encounter -n int
Integration "inject-momento-encounter" is now running. Showing log ...
[1] Monitoring pod inject-momento-encounter-67978845d-56dxc
[2] 2026-03-03 08:54:35,637 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency exte
[3] 2026-03-03 08:54:37,742 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[4] 2026-03-03 08:54:37,743 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[5] 2026-03-03 08:54:54,433 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[6] 2026-03-03 08:54:54,434 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:/etc/camel/sources/**
[7] 2026-03-03 08:54:54,435 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/configmaps/a
[8] 2026-03-03 08:54:54,435 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/configmaps/a
[9] 2026-03-03 08:54:54,435 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[10] 2026-03-03 08:55:02,736 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[11] 2026-03-03 08:55:03,745 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[12] 2026-03-03 08:55:06,445 INFO [org.apache.camel.component.kafka.KafkaConsumer] (main) Starting Kafka consumer on topic: inject-momento-encounter-topic with breakOnFirstError: false
[13] 2026-03-03 08:55:06,645 INFO [org.apache.camel.component.kafka.KafkaFetchRecords] (Camel (camel-1) thread #1 - KafkaConsumer[inject-momento-encounter-topic]) Connecting Kafka cons
[14] 2026-03-03 08:55:06,741 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[15] 2026-03-03 08:55:06,742 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[16] 2026-03-03 08:55:06,837 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[17] 2026-03-03 08:55:06,838 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[18] 2026-03-03 08:55:06,838 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[19] 2026-03-03 08:55:06,838 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[20] 2026-03-03 08:55:06,840 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[21] 2026-03-03 08:55:06,841 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties]
[22] 2026-03-03 08:55:06,842 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Using 2 instances of same component class: org.apache.camel.component.http.HttpComponent wi
[23] 2026-03-03 08:55:06,933 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Routes startup (total:5)
[24] 2026-03-03 08:55:06,934 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started InjectMomentEncounter-Kafka-Route (kafka://inject-momento-encounter-topic)
[25] 2026-03-03 08:55:06,934 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started ProcessorEncounterRoute (direct://processorEncounter)
[26] 2026-03-03 08:55:06,934 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started ObtenerPacienteRoute (direct://obtenerPaciente)
[27] 2026-03-03 08:55:06,934 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started InyectarTagRoute (direct://inyectarTag)
[28] 2026-03-03 08:55:06,935 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started autenticacion (direct://autenticacion)
[29] 2026-03-03 08:55:06,043 INFO [org.apache.camel.component.kafka.consumer.support.classic.AssignmentAdapterHelper] (Camel (camel-1) thread #1 - KafkaConsumer[inject-momento-encounte
[30] 2026-03-03 08:55:09,043 INFO [org.apache.camel.component.kafka.KafkaFetchRecords] (Camel (camel-1) thread #1 - KafkaConsumer[inject-momento-encounter-topic]) Searching for a custe
```

### 9.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración por un cambio en el diseño de la misma o una modificación en el alcance del microservicio, es necesario sustituir el directorio "inject-momento-encounter-chart" existente por el nuevo y 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
```

### 9.2.4 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.
- El ConfigMap definido dentro del Helm chart.

## 10 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.
- 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 Autenticacion para poder incluir Authorization Bearer **token** en las llamadas a AUT y HDR.

### 10.1 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:

Para UNIDAD

```
{
  "practitioner_document": "123456789Z",
  "role_code": "ACCESO_UNICAS_UNIDAD_PACIENTE",
  "center_code": "012345",
  "unit_code": "012345NEU",
  "careline_code": "CEX",
  "name": "Nombre Apellido1 Apellido2",
  "given_name": "Nombre",
  "family_name": "Apellido1 Apellido2",
  "email": "prueba@ejemplo.com",
  "profession_code": "DOC"
}
```

Para SISTEMA

```
{
  "practitioner_document": "123456789A",
  "role_code": "ADMIN_UNICAS",
  "center_code": "",
  "unit_code": "",
  "careline_code": "",
  "name": "Nombre Apellido1 Apellido2",
  "given_name": "Nombre",
  "family_name": "Apellido1 Apellido2",
  "email": "pruebaAdv@ejemplo.com",
  "profession_code": "ADV"
}
```

### 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 <code>source</code> .

## 10.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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-quete"

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

### 10.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración practitioner-sync utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio practitioner-sync-chart es la siguiente:

```

├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── PractitionerSync.java
│   └── templatesquete
│       ├── practitioner.quete.json
│       └── practitioner-role.quete.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.yaml
└── values.yaml

3 directories, 10 files
    
```

#### 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.

```

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 instale.

### 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install practitioner-sync ./practitioner-sync-chart -n <namespace>**.

Sustituyendo "<namespace>" por el namespace en el que se está instalando la integración, por ejemplo para un namespace llamado "int":

```
sh-4.2$ helm install practitioner-sync ./practitioner-sync-chart -n int
NAME: practitioner-sync
LAST DEPLOYED: Tue Mar 3 10:02:45 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration practitioner-sync.
- Configmap practitioner-sync-qute.

## 10.2.2 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 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-d62akb97lqc4tpljvid0        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":

```
sh-4.2$ kubectl logs practitioner-sync -n int
Integration 'practitioner-sync' is now running. Showing log ...
[1] Monitoring pod practitioner-sync-7649c6457-g24d
[1] 2026-02-19 14:20:56.059 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the d
[1] 2026-02-19 14:20:58.159 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-02-19 14:20:58.160 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-02-19 14:21:12.857 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-02-19 14:21:12.858 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routesIncludePattern = file:/etc/camel/sour
[1] 2026-02-19 14:21:12.859 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d
[1] 2026-02-19 14:21:12.860 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-02-19 14:21:12.860 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.context.restConfiguration.component = platform-h
[1] 2026-02-19 14:21:20.153 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] aut.url = http://ohaut-back:8080
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.aut.fhir = /hnaut/fhir
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] system.identifier.dni = urn:oid:1.3.6.1.4.1.19126.3
[1] 2026-02-19 14:21:20.861 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] hdr.url = http://ohdr-back:8080
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.hdr.fhir = /shseerver/fhir
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] sso.url = http://ohsso:8080
[1] 2026-02-19 14:21:20.862 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.sso = /auth
[1] 2026-02-19 14:21:20.863 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.request.body = grant_type=client_credentials&c
[1] 2026-02-19 14:21:20.863 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] oauth2.clientId = xxxxxx
[1] 2026-02-19 14:21:20.953 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Using 2 instances of same component class: org.apache.camel.component.http.HT
[1] 2026-02-19 14:21:20.955 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Routes startup (total:7 rest-dsl:1)
[1] 2026-02-19 14:21:20.955 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started BuscarPractitionerRoute (direct://buscar-practitioner)
[1] 2026-02-19 14:21:20.956 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started ComprobarUnidadRoute (direct://comprobar-unidad)
[1] 2026-02-19 14:21:20.956 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started BuscarPractitionerRoleRoute (direct://buscar-practitioner-role)
[1] 2026-02-19 14:21:20.956 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started CrearPractitionerRoleRoute (direct://crear-practitioner-role)
[1] 2026-02-19 14:21:20.957 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started CrearPractitionerRoute (direct://crear-practitioner)
[1] 2026-02-19 14:21:20.957 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started MainRoute (rest://post://practitionerSync)
[1] 2026-02-19 14:21:20.957 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started autenticacion (direct://Autenticacion)
[1] 2026-02-19 14:21:20.958 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) started in 804ms (build:0ms init:0ms start:804m
[1] 2026-02-19 14:21:22.753 INFO [io.quarkus] (main) camel-k-integration 2.7.0 on JVM (powered by Quarkus 3.30.6) started in 37.196s. Listening on: http://0.0.0.0:8080
[1] 2026-02-19 14:21:22.754 INFO [io.quarkus] (main) Profile prod activated.
```

### 10.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración por un cambio en el diseño de la misma o una modificación en el alcance del microservicio, es necesario sustituir el directorio "practitioner-sync-chart" existente por el nuevo y 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
```

### 10.2.4 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.

## 11 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.

### 11.1 ENDPOINTS DISPONIBLES

#### PUT /ie/ProxyMPI

Envío del patient para el alta o modificación de datos.

##### Entrada

Authentication: Bearer token

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

Content-Type: application/json

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

##### Salida

La estructura mensaje salida en caso de resultado 200 OK es un recurso Patient formateado debidamente para la aplicación 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/UNICASConfidentiality",
        "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": "BBBBBBBBBHH000848"
  },
  {
    "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/lastname-fathers-family",
            "valueString": "PEREZ"
          },
          {
            "url":
"http://hl7.org/fhir/StructureDefinition/lastname-mothers-family",
            "valueString": "CERCADO"
          }
        ],
        "use": "nickname",
        "text": "JOAQUIN PEREZ CERCADO",
        "family": "PEREZ CERCADO",
        "_family": {
          "extension": [
            {
              "url":
"http://hl7.org/fhir/StructureDefinition/lastname-fathers-family",
              "valueString": "PEREZ"
            },
            {
              "url":
"http://hl7.org/fhir/StructureDefinition/lastname-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"

## 11.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **<nombre microservicio>**, 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).
- Configuración del gateway.

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

### 11.2.1 INSTALACIÓN

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:

```
patient-proxy-chart/
├── Chart.yaml
├── files
│   ├── KeyCloakPolicy.java
│   └── PatientProxy.java
├── README.md
├── templates
│   ├── _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.

```
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.

A continuación, deberemos ejecutar el comando siguiente: **helm install patient-proxy ./patient-proxy-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm install patient-proxy ./patient-proxy-chart/ -n int
NAME: patient-proxy
LAST DEPLOYED: Tue Mar 3 16:05:38 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration patient-proxy.
- Configmap patient-proxy.

- Httproute patient-proxy-gateway.

## 11.2.2 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 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":

```
sh-4.2$ kamel logs patient-proxy -n int
Integration 'patient-proxy' is now running. Showing log ...
[!] Monitoring pod patient-proxy-460b647c-qjanz
[!] 2024-03-03 16:07:13,393 INFO [org.apache.camel] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[!] 2024-03-03 16:06:55,242 INFO [org.apache.camel] (main) Apache Camel Quarkus 4.16.0 is starting
[!] 2024-03-03 16:06:55,308 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[!] 2024-03-03 16:07:13,030 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[!] 2024-03-03 16:07:13,081 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] = camel.main.routesIncludePattern = file:/etc/camel/sources/**
[!] 2024-03-03 16:07:13,092 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] = camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/ha-properties,/etc/camel/conf.d/_configmaps/ha-kafka-properties,/etc/camel/conf.d/_secrets/ha-secrets,/etc/camel/conf.d/_secrets/ha-kafka-secrets
[!] 2024-03-03 16:07:13,093 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] = camel.main.sourceLocationEnabled = true
[!] 2024-03-03 16:07:13,093 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] = camel.context.restConfiguration.component = platform-http
[!] 2024-03-03 16:07:24,196 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (cm-1-1) is starting
[!] 2024-03-03 16:07:24,242 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[!] 2024-03-03 16:07:30,234 INFO [org.apache.camel.main.BaseMainSupport] (main) PropertyPlaceholder summary
[!] 2024-03-03 16:07:30,235 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = identifier.allowed.systems = ["*:*:*:*:8972400015*"]
[!] 2024-03-03 16:07:30,235 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.mds-ata.url = https://unicae-ehi1.sanidad.gob.es/StructureDefinition/ModaMala
[!] 2024-03-03 16:07:30,235 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.mds-impl.url = https://unicae-ehi1.sanidad.gob.es/StructureDefinition/ModaInscripcion
[!] 2024-03-03 16:07:30,236 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.mds-qa.url = https://unicae-ehi1.sanidad.gob.es/StructureDefinition/ModaMala
[!] 2024-03-03 16:07:30,236 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.mds-origen.system = unicae.origen
[!] 2024-03-03 16:07:30,236 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.mds-impl.system.url = http://unicae-ehi1.sanidad.gob.es/CodeSystem/CodigoMdsAutonominas
[!] 2024-03-03 16:07:30,237 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.identifier-guardian.url = http://ha.indra.es/ehi/StructureDefinition/identifier-guardian
[!] 2024-03-03 16:07:30,237 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.telecom-identifier.url = http://ha.indra.es/ehi/StructureDefinition/telecom-identifier
[!] 2024-03-03 16:07:30,237 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.address-kyosha.url = http://ha.indra.es/ehi/StructureDefinition/address-kyosha
[!] 2024-03-03 16:07:30,238 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.processor.url = http://ha.indra.es/ehi/StructureDefinition/processor
[!] 2024-03-03 16:07:30,238 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = extension.url = http://dmgi-locati0898
[!] 2024-03-03 16:07:30,238 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = context.api.file = /jpsb/ehi
[!] 2024-03-03 16:07:30,238 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = context.url = http://ehint-bank:8089
[!] 2024-03-03 16:07:30,239 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = context.port = /hobot
[!] 2024-03-03 16:07:30,240 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] = camel.identifier.catalog = UnicaeEhi1672400015
```

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$
```

### 11.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración, es necesario sustituir el directorio "patient-proxy-chart" existente por el nuevo y ejecutar el siguiente comando: **helm upgrade patient-proxy ./patient-proxy-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
```

### 11.2.4 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.

## 12 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 de que ya lo estuviera le modificaremos los datos pertinentes con su correspondiente notificación a NC tanto en el caso de alta como de modificación Core.

La validación de la autenticación es por KeyCloakPolicy y el rol necesario es "OHBPBPM\_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 tenía momento Únicas:
    - Generar Condition.
    - Notificar la modificación core a NC.
    - Si hay error en la notificación 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 modificación Core a NC.
  - Si hay error en la notificación se llama a la integración rollbackPatient.

### 12.1 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": "dddd"
      }
    ]
  },
  {
    "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 resaltado 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": "BBBBBBBBBHH000849"
}
},
"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"

## 12.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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 (incluidas plantillas Qute, schemas,... si los hubiera).

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

### 12.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **qr-to-condition** utilizando Helm.

#### 1. Preparación del chart

Verificar que la estructura del directorio **qr-to-condition-chart** es la siguiente:

```
qr-to-condition-chart/
├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── KeyCloakPolicy.java
│   ├── NCUtils.java
│   ├── QrToCondition.java
│   └── templatesqute
│       ├── condition-transform.qute.json
│       ├── modificacion-core.qute.json
│       └── nc-alta-paciente.qute.json
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.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.

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

## 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install qr-to-condition ./qr-to-condition-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm install qr-to-condition ./qr-to-condition-chart/ -n int
NAME: qr-to-condition
LAST DEPLOYED: Tue Mar 3 17:32:41 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration qr-to-condition.
- Configmap qr-to-condition-quete.

## 12.2.2 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 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-gute -n int
NAME          DATA      AGE
qr-to-condition-gute 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":

```
sh-4.2$ kamel logs qr-to-condition -n int
Integration "qr-to-condition" is now running. Showing log ...
v1) Monitoring pod qr-to-condition-5fd8e567d-rdzhn
[1] 2026-03-03 17:33:54,115 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-03 17:33:56,416 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-03 17:33:56,518 INFO [org.apache.camel.main.BaseMainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-03 17:34:18,521 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-03 17:34:18,521 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] c
camel.main.routes.includePattern = file:/etc/camel/sources/**
[1] 2026-03-03 17:34:18,521 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] c
camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-properties/etc/camel/conf.d/_configmaps/na-kafk
a-properties/etc/camel/conf.d/_secrets/na-secrets/etc/camel/conf.d/_secrets/na-kafka-secrets
[1] 2026-03-03 17:34:18,521 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] c
camel.main.sourceLocationEnabled = true
[1] 2026-03-03 17:34:18,521 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] c
camel.context.restConfiguration.component = platform-http
[1] 2026-03-03 17:34:32,615 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (cam
el-1) is starting
[1] 2026-03-03 17:34:33,720 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully
logged in.
[1] 2026-03-03 17:34:37,218 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-03 17:34:37,218 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] h
de.url = http://ohhri-bank18090
[1] 2026-03-03 17:34:37,219 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] c
contexto.hdr = /ehrsrver
[1] 2026-03-03 17:34:37,219 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] c
contexto.hdr.fhir = /ehrsrver/fhir
[1] 2026-03-03 17:34:37,219 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] a
system.identifier.dni = urn:oid:1.3.6.1.4.1.19126.3
[1] 2026-03-03 17:34:37,220 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] c
de.url = http://ohhri-bank18090
[1] 2026-03-03 17:34:37,220 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] a
contexto.mpi.fhir = /iapob/fhir
[1] 2026-03-03 17:34:37,220 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] a
system.identifier.ciomsa = urn:oid:2.16.724.4.40
[1] 2026-03-03 17:34:37,220 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] a
extension.momento-unicae.url = https://unicae-fhir.salud.gob.es/StructureDefinition/MomentoUnicae
[1] 2026-03-03 17:34:37,221 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] c
extensio.momento-unicae.url = https://unicae-fhir.salud.gob.es/StructureDefinition/MomentoUnicae
```

## 12.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración, es necesario sustituir el directorio "qr-to-condition-chart" existente por el nuevo y ejecutar el siguiente comando: **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
```

## 12.2.4 DESINSTALACIÓN

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

```
sh-4.2$ helm uninstall qr-to-condition -n int
release "qr-to-condition" 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 ConfigMaps definidos dentro del Helm chart.

## 13 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:
  - Desde el endpoint antes indicado recibimos un aviso de 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 para nuestro 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 para 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, obtendremos el paciente completo desde 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 notificación 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.

## 13.1 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 envío
  "observaciones": ""
}
```

#### Salida

El peticionario va a recibir un 200 OK si el nodo autonómico 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

## 13.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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).
- Configuración del gateway.

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

### 13.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación 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
├── Chart.yaml
├── files
│   ├── aggregation
│   │   └── DatosCoreAggregationStrategy.java
│   ├── Authentication.java
│   ├── ConsultaNotificacionesNC.java
│   ├── KeycloakPolicy.java
│   ├── NotificacionCriticaNC.java
│   ├── ProcesadorNotificacionesKafka.java
│   ├── templatesqute
│   │   ├── bundle-searchset.qute.json
│   │   └── paciente-core.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

Para la instalación, sólo se deberá 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.

A continuación, deberemos ejecutar el comando siguiente: **helm install procesar-notificaciones-nc ./procesar-notificaciones-nc-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm install procesar-notificaciones-nc ./procesar-notificaciones-nc-chart -n int
NAME: procesar-notificaciones-nc
LAST DEPLOYED: Wed Mar  4 09:51:36 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
sh-4.2$
```

Esto creará todo lo necesario para que la integración funcione:

- Integration procesar-notificaciones-nc.
- Configmap procesar-notificaciones-nc-queue.
- Configmap procesar-notificaciones-nc-schemas.
- Httproute procesar-notificaciones-nc-gateway.

## 13.2.2 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 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-notificaciones-nc -n int
NAME                                PHASE    READY
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":

```
sh-4.2$ kubectl logs procesar-notificaciones-nc -n int
Integration 'procesar-notificaciones-nc' is now running. Showing log ...
[1] Monitoring pod procesar-notificaciones-nc-c7dd76fff-cktjg
[1] 2026-03-04 09:52:44,514 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-04 09:52:46,512 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-04 09:52:46,513 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-04 09:53:06,514 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-04 09:53:06,514 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.routes.includePattern = file:/etc/camel/sources/**
[1] 2026-03-04 09:53:06,515 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudProperties.location = /etc/camel/conf.d/_configmaps/na-properties,/etc/camel/conf.d/_configmaps/na-kafka-properties,/etc/camel/conf.d/_secrets/na-secrets,/etc/camel/conf.d/_secrets/na-kafka-secrets
[1] 2026-03-04 09:53:06,516 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-03-04 09:53:06,516 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.context.restConfiguration.component = platform-http
[1] 2026-03-04 09:53:07,521 INFO [io.quarkus.quartz.runtime.QuartzSchedulerImpl] (main) No scheduled business methods found - Quartz scheduler will not be started
[1] 2026-03-04 09:53:18,710 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-04 09:53:18,714 INFO [org.apache.camel.component.quartz.QuartzComponent] (main) Setting org.quartz.scheduler.jmx.export=true to ensure QuartzScheduler(s) will be enlisted in JMX
[1] 2026-03-04 09:53:20,607 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[1] 2026-03-04 09:53:25,308 INFO [org.apache.camel.component.quartz.QuartzEndpoint] (main) Job Camel_camel-1.consultanotificacionesNC (cron=0 * * * * ?, triggerType=CronTriggerImpl, jobClass=CamelJob) is scheduled. Next fire date is 2026-03-04T10:00:00.000+0000
[1] 2026-03-04 09:53:25,615 INFO [org.apache.camel.component.kafka.KafkaConsumer] (main) Starting Kafka consumer on topic: notif_alta_paciente,notif_baja_paciente,notif_na_inactivo,notif_manual,notif_modificacion_core,notif_modificacion_generica with breakOnFirstError: false
[1] 2026-03-04 09:53:25,915 INFO [org.apache.camel.component.kafka.KafkaFetchRecords] (Camel (camel-1) thread #1 - KafkaConsumer[notif_alta_paciente,notif_baja_paciente,notif_na_inactivo,notif_manual,notif_modificacion_core,notif_mod
```

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":

```
NAME                                HOSTNAMES    AGE
procesar-notificaciones-nc-gateway  []           5m2s
sh-4.2$
```

### 13.2.3 ACTUALIZACIÓN

En caso de necesitar actualizar la integración, 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$
```

### 13.2.4 DESINSTALACIÓN

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

```
sh-4.2$ helm uninstall procesar-notificaciones-nc -n int  
release "procesar-notificaciones-nc" 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 (quite y schemas).
- El gateway generado para la exposición de los endpoints.

## 14 Microservicio enriquecer-url-attachment

Existe una suscripción sobre ENCOUNTER bajo el criterio "Encounter?status=finished&patient:missing=true&\_tag=mymed.app.target|ohzeus". Si esto se cumple el DocumentReference vinculado a ese Encounter se mete en el payload y se envía el evento al topic "enrich\_url" del kafka.

Este servicio consume de este topic y procesa el DocumentReference metiendo la URL base a la url del binary del DocumentReference en caso de que esta sea relativa y no absoluta (http\*), y lo actualiza en nuestro HDR.

Si se produce cualquier error con el recurso tratado, este se enviará al topic "enrich-url-error".

### 14.1 ENDPOINTS DISPONIBLES

No aplica.

### 14.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación y desinstalación del microservicio **enriquecer-url-attachment**, 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).

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

#### 14.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **enriquecer-url-attachment** utilizando Helm.

##### 1. Preparación del chart

Verificar que la estructura del directorio **enriquecer-url-attachment-chart** es la siguiente:

```
enriquecer-url-attachment-chart/
├── Chart.yaml
├── files
│   ├── Authentication.java
│   ├── EnriquecerUrlAttachement.java
│   └── templatesqute
│       └── patch-template.qute.xml
├── README.md
├── templates
│   ├── configmap.yaml
│   ├── _helpers.tpl
│   └── integration.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.

```
integration:
  name: enriquecer-url-attachment
  namespace: #namespace en el que se quiere desplegar la integración
  runtime:
    provider: plain-quarkus
    version: 3.30.6
```

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: enriquecer-url-attachment-gateway
  unicasbase: #url base del servidor. Por ejemplo: integracio.unicas.salut.intranet.gencat.cat
```

## 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install enriquecer-url-attachment ./enriquecer-url-attachment-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm install enriquecer-url-attachment ./enriquecer-url-attachment-chart/ -n int
NAME: enriquecer-url-attachment
LAST DEPLOYED: Wed Mar 4 10:10:21 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration enriquecer-url-attachment.
- Configmap enriquecer-url-attachment-qute.

## 14.2.2 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 enriquecer-url-attachment -n <namespace>` podemos comprobar si la integración está ejecutándose (Running). Ejemplo para esta integración en namespace "int":

NAME	PHASE	READY
enriquecer-url-attachment	Running	True
hvbvdpd10	1	

2. Consultar los logs del microservicio:

Para comprobar los logs, se puede usar el siguiente comando Kamel: `kamel logs enriquecer-url-attachment -n <namespace>`. Ejemplo de esta integración en "int":

```
sh-4.2$ kamel logs enriquecer-url-attachment -n int
Integration 'enriquecer-url-attachment' is now running. Showing log ...
[1] Monitoring pod enriquecer-url-attachment-58496f94b-xlbt4
[1] 2026-03-04 10:11:32,793 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.log.console.json" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
[1] 2026-03-04 10:11:34,000 INFO [org.apache.camel.quarkus.core.CamelBootstrapRecorder] (main) Apache Camel Quarkus 3.30.0 is starting
[1] 2026-03-04 10:11:34,991 INFO [org.apache.camel.main.MainSupport] (main) Apache Camel (Main) 4.16.0 is starting
[1] 2026-03-04 10:11:52,997 INFO [org.apache.camel.main.BaseMainSupport] (main) Auto-configuration summary
[1] 2026-03-04 10:11:53,002 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.router.includePatterns = file:/etc/camel/sources/**
[1] 2026-03-04 10:11:53,090 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.cloudPropertiesLocation = /etc/camel/conf.d/_configmaps/na-properties,/etc/camel/conf.d/_configmaps/na-kafka-properties,/etc/camel/conf.d/_secrets/na-secrets,/etc/camel/conf.d/_secrets/na-kafka-secrets
[1] 2026-03-04 10:11:53,091 INFO [org.apache.camel.main.BaseMainSupport] (main) [MicroProfilePropertiesSource] camel.main.sourceLocationEnabled = true
[1] 2026-03-04 10:11:58,282 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Apache Camel 4.16.0 (camel-1) is starting
[1] 2026-03-04 10:12:00,199 INFO [org.apache.kafka.common.security.authenticator.AbstractLogin] (main) Successfully logged in.
[1] 2026-03-04 10:12:02,101 INFO [org.apache.camel.component.kafka.KafkaConsumer] (main) Starting Kafka consumer on topic: enrich-url with bootstrapServers: false
[1] 2026-03-04 10:12:02,392 INFO [org.apache.camel.main.BaseMainSupport] (main) Property-placeholder summary
[1] 2026-03-04 10:12:02,392 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] auth2.request.body.grant.type=client_credentials&client_id=na-cliente-admin&client_secret=2JpEvi6FfuzMvwXplRwIFyRzKD7y5escope=offline_access
[1] 2026-03-04 10:12:02,394 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] auth2.tokenEndpoint = http://ohasso:8080/auth/realms/oh-base/protocol/openid-connect/token
[1] 2026-03-04 10:12:02,394 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] http.url = http://ohasso:8080
[1] 2026-03-04 10:12:02,395 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] contexto.hdr.fhnr = /ehsserver/fhnr
[1] 2026-03-04 10:12:02,395 INFO [org.apache.camel.main.BaseMainSupport] (main) [OverrideProperties] auth2.clientId = xxxxxx
[1] 2026-03-04 10:12:02,397 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Routes startup (total:5)
[1] 2026-03-04 10:12:02,397 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started autenticacion (direct://autenticacion)
[1] 2026-03-04 10:12:02,397 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started PRINCIPAL-KAFKA (kafka://enrich-url)
[1] 2026-03-04 10:12:02,398 INFO [org.apache.camel.impl.engine.AbstractCamelContext] (main) Started enriquecerUrlAttachment (direct://enriquecer-url-attachment)
```

## 14.2.3 ACTUALIZACIÓN

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

```
sh-4.2$ helm upgrade enriquecer-url-attachment ./enriquecer-url-attachment-chart/ -n int
Release 'enriquecer-url-attachment' has been upgraded. Happy Helming!
NAME: enriquecer-url-attachment
LAST DEPLOYED: Wed Mar 4 10:10:47 2026
NAMESPACE: int
STATUS: deployed
REVISION: 2
TEST SUITE: None
```

## 14.2.4 DESINSTALACIÓN

Si se necesita desinstalar la integración, se ejecuta el siguiente comando: `helm uninstall enriquecer-url-attachment -n <namespace>`.

```
sh-4.2$ helm uninstall enriquecer-url-attachment -n int  
release "enriquecer-url-attachment" uninstalled  
sh-4.2$
```

Al ejecutar el comando, Helm elimina:helm uninstall enri:

- 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.

## 15 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 no se vuelvan a pedir los que ya se han obtenido con anterioridad.

### Comportamiento detallado:

- Obtenemos el CipSNS del Paciente proporcionado.
- Obtener fecha de la última 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 recopilada 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" está 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 algun nodo este inactivo en la última "paginación" se devuelve una lista de dichos nodos.

### 15.1 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 repositorio, el array de entry's vacío (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:

```

{
  "resourceType": "Bundle",
  "type": "searchset",
  "total": 0,
  "entry": [],
  "issues": {}
}
    
```

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

```

{
  "resourceType": "Bundle",
  "type": "searchset",
  "total": 0,
  "entry": [],
  "issues": Se han encontrado nodos con indice de presencia inactivos: ES-C1, ES-C2
}
    
```

En caso de error se devolverá el mismo bundle searchset pero con la información 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):

```

{
  "resourceType": "Bundle",
  "type": "searchset",
  "total": 0,
  "entry": [],
  "issues": Error: class org.apache.camel.CamelAuthorizationException - Content-type: - Message: Access token not found in exchange. Exchange[]
}
    
```

## 15.2 INSTALACIÓN HELM CHART

Este apartado describe el procedimiento para la instalación 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).
- Configuración del gateway.

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

## 15.2.1 INSTALACIÓN

A continuación, se describe el procedimiento para la correcta instalación de la integración **obtener-info-paciente** utilizando Helm.

### 1. Preparación del chart

Verificar que la estructura del directorio `obtener-info-paciente-chart` es la siguiente:

```
obtener-info-paciente-chart/
├── Chart.yaml
├── files
│   ├── aggregation
│   │   └── ListAggregationStrategy.java
│   ├── AuditEventUtils.java
│   ├── KeyCloakPolicy.java
│   ├── ObtenerExtras.java
│   ├── ObtenerInfoPaciente.java
│   ├── PatientUtils.java
│   └── templatesqute
│       ├── audit-event.qute.json
│       ├── bundle-batch-extras.qute.json
│       ├── bundle-batch.qute.json
│       └── bundle-searchset.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.

```
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
```

### 3. Ejecutar comando de instalación.

A continuación, deberemos ejecutar el comando siguiente: **helm install obtener-info-paciente ./obtener-info-paciente-chart -n <namespace>**.

Ejemplo para el namespace "int":

```
sh-4.2$ helm install obtener-info-paciente ./obtener-info-paciente-chart/ -n int
NAME: obtener-info-paciente
LAST DEPLOYED: Wed Mar  4 10:51:28 2026
NAMESPACE: int
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Esto creará todo lo necesario para que la integración funcione:

- Integration obtener-info-paciente.
- Configmap obtener-info-paciente-quete.
- Httproute obtener-info-paciente-gateway.

## 15.2.2 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 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-info-paciente -n int
NAME                                PHASE    READY
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":



```
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.

## Anexo I

<b>CODIGO</b>	<b>Descripción</b>	<b>Código [CA]</b>
ES-AN	Andalucía	Andalucia
ES-AR	Aragón	Aragon
ES-AS	Principado de Asturias	Asturias
ES-CB	Cantabria	Cantabria
ES-CM	Castilla-La Mancha	CastillaLaMancha
ES-CL	Castilla y León	CastillaLeon
ES-CT	Cataluña	Catalunya
ES-EX	Extremadura	Extremadura
ES-GA	Galicia	Galicia
ES-IB	Illes Balears	Baleares
ES-MD	Comunidad de Madrid	Madrid
ES-MC	Región de Murcia	Murcia
ES-NC	Comunidad Foral de Navarra	Navarra
ES-PV	País Vasco	PaisVasco
ES-RI	La Rioja	LaRioja
ES-CN	Canarias	Canarias
ES-CV	Comunidad Valenciana	Valencia
ES-01	Ingesa	Ingesa
ES-C1	Ministerio sanidad A	MinisterioSanidadA
ES-C2	Ministerio sanidad B	MinisterioSanidadB
ES-C3	Ministerio sanidad C	MinisterioSanidadC