

Artículo

[Bernardo Linarez](#) · 22 nov, 2021 Lectura de 4 min

## Streams en trazas de interoperabilidad

Hace algún tiempo, empecé a recibir alertas de consumo excesivo de espacio en el sistema de archivos (filesystem) de un cliente, cuya solución utiliza la capa de interoperabilidad (IRIS / Ensemble) de manera masiva.

Me percaté que las bases de datos que crecían eran las dedicadas a la interoperabilidad, mas no la base de datos operacional de la solución, por tanto había que revisar los mensajes de las diferentes integraciones presentes.

También verifiqué los espacios libres de cada base de datos por si compactar y truncar podrían liberar algún espacio importante, pero no era el caso.

(imagen referencial)

<https://docs.intersystems.com/irisforhealth2021/csp/docbook/DocBook.UI.Page.cls?KEY=GSAmange#GSAmangedatabasesinfo>

# Databases

Integrity CheckIntegrity Log

Last update: 2021-11-22 20:37:55.795

To view details and to manage a local database, click the database name link from the table below:

Filter:  Page size:  Max rows:  Results: 25 Page:  of 1 ☐ General view ☒ Free space view

| Name                     | Directory                          | Max Size  | Size « | Expansion Size | Available | % Free |
|--------------------------|------------------------------------|-----------|--------|----------------|-----------|--------|
| <a href="#">IRISLIB</a>  | c:\intersystems\iris\mgr\irislib\  | Unlimited | 384MB  | System Default | 0.34MB    | .08    |
| <a href="#">ENSLIB</a>   | c:\intersystems\iris\mgr\enslib\   | Unlimited | 163MB  | System Default | 12MB      | 7.36   |
| <a href="#">IRISSYS</a>  | c:\intersystems\iris\mgr\          | Unlimited | 127MB  | System Default | 58MB      | 45.66  |
| <a href="#">MEGALABS</a> | c:\intersystems\iris\mgr\megalabs\ | Unlimited | 31MB   | System Default | 6.8MB     | 21.93  |

Procedí a revisar si la tarea (task) core de limpieza de mensajes se encontraba configurada y si se estaba ejecutando. Efectivamente la task se ejecutaba con normalidad, por tanto las integraciones estaba generando muchos mensajes y eventualmente pesados.

# Task Details

[Edit](#)[History](#)

## GENERAL INFORMATION

|                |                             |
|----------------|-----------------------------|
| Task Name:     | IRIS Purge Interop Messages |
| Description:   | IRIS Purge Interop Messages |
| Namespace:     | REPO                        |
| Task Class:    | Ens.Util.Tasks.Purge        |
| Task Priority: | Normal                      |
| Is Batch Mode: | No                          |
| Type:          | User                        |
| Suspended:     |                             |
| Last Status:   | 1                           |

Dado que no tenía claridad de la naturaleza de todos los mensajes que circulaban por esa producción, decidí ir por el lado de investigar las globales que ocupaban mas espacio, utilizando el comando ^%GZISE, el cual se ejecuta desde consola del namespace que tenga mapeada la base de datos que estamos analizando.

```
REPO>Do ^%GZISE
```

| Page: 1         |          | GLOBAL SIZE    |         |           |
|-----------------|----------|----------------|---------|-----------|
| Global          | Blocks   | Bytes Used     | Packing | Contig.   |
| BLT             | 61497    | 365,829,824    | 73 %    | 10,573    |
| CacheStream     | 11673237 | 90,941,838,777 | 95 %    | 7,388,787 |
| DeepSee.Cubes   | 1        | 280            | 3 %     | 0         |
| DeepSee.FolderD | 1        | 68             | 1 %     | 0         |

(imagen referencial)

Esto me dio una idea directa de que la global ^CacheStream estaba ocupando demasiado espacio comparándola con el tamaño total de la base de datos.

Entonces procedí a mirar algunas definiciones de las clases de mensajería, encontrando ejemplos como estos

```
Class Test.GetEntity.Response Extends Ens.Response
{
    Property Return As %GlobalCharacterStream;
```

```
Property StatusCode As %String(MAXLEN = "");
```

En varios casos había una propiedad de tipo %GlobalCharacterStream. Sin embargo las clases con este tipo de propiedades tienen una global particular definida en su persistencia, por lo que estos stream (cadenas de texto grandes) deberían estar guardados en su global particular.

```
Storage Default
{
....
<StreamLocation>^Test.GetEntity.ResponseS</StreamLocation>
...
}
```

Sin embargo, encontré a través de una inspección de la global ^CacheStream, cadenas de texto que deberían estar en las globales dedicadas a stream de sus respectivas clases.

|                                                               |                                  |                                                                                        |                                       |
|---------------------------------------------------------------|----------------------------------|----------------------------------------------------------------------------------------|---------------------------------------|
| Global Search Mask: <input type="text" value="^CacheStream"/> |                                  | <input type="button" value="Display"/>                                                 | <input type="button" value="Cancel"/> |
| Search History: <input type="text" value="^CacheStream"/>     | <input type="button" value="x"/> | Maximum Rows: <input type="text" value="100"/>                                         | <input type="checkbox"/> Allow Edit   |
| 1:                                                            | ^CacheStream                     | = 5635078                                                                              |                                       |
| 2:                                                            | ^CacheStream(1400014)            | = 1                                                                                    |                                       |
| 3:                                                            | ^CacheStream(1400014,0)          | = 826                                                                                  |                                       |
| 4:                                                            | ^CacheStream(1400014,1)          | = "{ ""Nombre"": ""DAMARIS MACARENA"", ""ApellidoPaterno"": ""ABASTO"", ""ApellidoMat" |                                       |
| 5:                                                            | ^CacheStream(1400015)            | = 1                                                                                    |                                       |
| 6:                                                            | ^CacheStream(1400015,0)          | = 1005                                                                                 |                                       |
| 7:                                                            | ^CacheStream(1400015,1)          | = "{ ""Nombre"": ""DAMARIS MACARENA"", ""ApellidoPaterno"": ""ABASTO"", ""ApellidoMat" |                                       |
| 8:                                                            | ^CacheStream(1400018)            | = 1                                                                                    |                                       |
| 9:                                                            | ^CacheStream(1400018,0)          | = 752                                                                                  |                                       |
| 10:                                                           | ^CacheStream(1400018,1)          | = "[{ ""IDEXAMEN"": ""MDMwNjA4MjAxICAgLTA1ICAtMjEwNjE5NDk0NS0zMjA1MjQ4Nw==", ""NUI"    |                                       |
| 11:                                                           | ^CacheStream(1400022)            | = 1                                                                                    |                                       |

(imagen referencial)

Buscando en la comunidad encontré el siguiente post

<https://community.intersystems.com/post/streamlocation-doesnt-seem-affec...>

Donde una de las respuestas comenta que cierta sintaxis de asignación, afecta directamente en el comportamiento de persistencia del stream.



Robert Cemper · Mar 30, 2020

Hi Jenna,

I took some time to verify my suspicion. (Caché 18.1)

You depend on the type of stream that is used in your object

#A) %Library.GlobalBinaryStream or %Library.GlobalCharacterStream

- if you have no stream yet you run `do obj.MyStream.Write("whatever")` then your stream will land in `^SSA.DocumentCacheS` as expected
- but if you get an already existing external stream and set `obj.MyStream = myStreamOref` then just the oref / OID of the stream is set including `^CacheStream`.

Which seems to be your case 🙄🙄🙄

it's not a big surprise as ENSEMBLE may still use the old style.

#B) using %Stream.GlobalBinary, ...

- Both cases ended as expected with the stream in `^SSA.DocumentCacheS` It seems to me that a `CopyFromStream` happens during the assignment. I'd name it expected behavior. 😊

En vista de esto, empecé a mirar en el código de los diferentes host respecto de como se interactuaba con los stream, encontrando sintaxis de este estilo:

```
set pResponse.ResponseJSON = fichaClinica
```

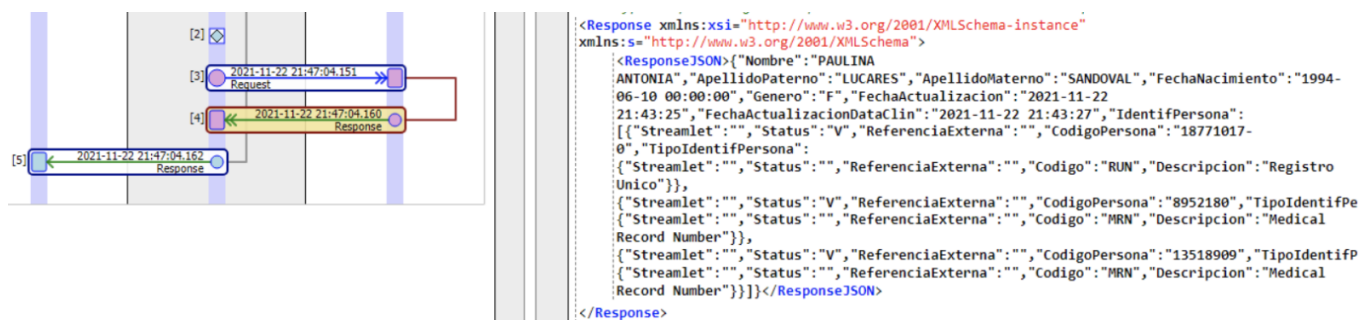
En donde la variable `fichaClinica` es un Stream. Entonces procedí a cambiar esta sintaxis por:

```
Set tSC = pResponse.ResponseJSON.CopyFrom(fichaClinica)
Quit:$System.Status.IsError(tSC)
```

De esta manera, se dejó de acumular tanta información en el `^CacheStream` y se empezó a distribuir a sus respectivas globales, y se pudo limpiar dicha global.

Sin embargo, posterior a este análisis, se volvió a notar de un aumento en el consumo de espacio en disco. En efecto, la solución estaba recibiendo muchísimas más peticiones (request) que en el momento anterior (casi el cuádruple). Entonces, directamente fui a investigar las globales particulares, comprobando que habían crecido a un ritmo más acelerado.

Entonces decidí mirar las trazas que generaba la solución, percatándome de una en particular:



(imagen referencial)

En este caso de uso, la propiedad `ResponseJSON` era un stream que representaba el resumen clínico de un paciente (en formato JSON). Era el stream mas largo de entre los diferentes casos de uso. Los eventos [4] y [5] contenían exactamente el mismo JSON, dado que el Business Process (BP) no realizaba ninguna acción sobre

dicho stream, sino que simplemente traspasaba el contenido entre el response del BP hacia el response del Business Operation (BO). Además este JSON se guarda en una tabla de bitácora propia de la solución.

En otras palabras, ¡este stream se guardaba en 3 tablas diferentes!. Este JSON persistía en la tabla del response del BO, luego en el response del BP y finalmente en la bitácora de la propia solución.

Entonces se aplicó un cambio a la lógica de esta parte de la integración. En lugar de traspasar el propio stream, se traspasa un objeto con un puntero lógico a ese stream, para que sea obtenido, sólo en el momento que sea necesario:

```
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <Entidades>
    <ResponseDetail>
      <IdEntidad>2361155</IdEntidad>
      <NombreClaseEntidad>Repo.Models.Identificacion.Ic
    </ResponseDetail>
  </Entidades>
  <IdPaciente>769716</IdPaciente>
</Response>
```

Esta lógica se implementó a través de una clase que contuviera el %id y la clase que guarda la stream. De esta manera la información ya no se guarda varias veces, y además se evita mostrar datos clínicos en las trazas de integración.

El cambio permitió un consumo menor de espacio en las trazas.

Espero que esta experiencia les ayude en sus implementaciones

[#Ensemble](#) [#InterSystems](#) [IRIS](#)

---

URL de fuente: <https://es.community.intersystems.com/post/streams-en-trazas-de-interoperabilidad>