
Artículo

[Ricardo Paiva](#) · 7 jul, 2022 Lectura de 17 min

[Open Exchange](#)

Creando un Visualizador de Mensajes alternativo en IRIS

Si tuvieras la oportunidad de cambiar algo en el Visualizador de Mensajes de Interoperabilidad en IRIS, ¿qué harías?

Después de publicar el artículo [Panel de Control "IRIS History Monitor"](#), recibí algunos comentarios muy interesantes y varias peticiones. Una de ellas fue un Visualizador de Mensajes mejorado.

Si aún no lo has hecho, echa un vistazo al proyecto: merece la pena que le dediques un rato, y además ganó el 3er premio (Bronce) a [Los mejores desarrolladores y aplicaciones de InterSystems Open Exchange en 2019](#).

Empecé a pensar algunas ideas sobre lo que me gustaría incluir en el "nuevo" Visualizador de Mensajes pero ¿cómo podría mostrar estos recursos de la forma más rápida y sencilla?

Bueno, antes de nada. Por lo general, se empieza configurando una producción de interoperabilidad, y después se exporta y se implementa en el sistema de destino, como se indica en la [documentación](#). Este es un proceso que realmente no me gusta. En realidad, no es que haya algo malo en él. Solo que he idealizado hacer todo mediante el código.

Espero que cada vez que alguien ejecute este tipo de proyectos, empiece de esta manera:

```
$ docker-compose build
```

```
$ docker-compose up -d
```

```
;;;Y eso es todo!!!
```

Con estos sencillos pasos en mente, comencé a buscar en la comunidad de InterSystems y encontré algunos consejos. En una de las publicaciones surgió la pregunta que me estaba haciendo: [¿Cómo crear producciones desde una rutina?](#)

En esa publicación, [@Eduard Lebedyuk](#) respondió, mostrando cómo crear una producción mediante el uso de un código.

"Para crear la clase de producción de forma automática es necesario:

1. Crear el objeto %Dictionary.ClassDefinition para tu producción de prueba
2. Crear el objeto Ens.Config.Production
3. Crear %Dictionary.XDataDefinition
4. Serializar (2) en (3)
5. Insertar XData (3) en (1)
6. Guardar y compilar (1)"

También encontré un comentario de [@Jenny Ames](#):

"Una de las prácticas que recomendamos con frecuencia es crear hacia atrás. Crea las business operations primero, después los business processes, después los business services..."

Así que, ¡hagámoslo!

Solicitudes, Business operations y Business services

La clase **diashenrique.messageviewer.util.InstallerProduction.cls** es, como su nombre lo indica, la clase que se encarga de instalar nuestra producción. El manifiesto del instalador invoca el ClassMethod ****Install**** desde esa clase:

```
/// Helper to install a production to display capabilities of the enhanced viewer
```

```
ClassMethod Install() As %Status
```

```
{
    Set sc = $$$OK
    Try {
        Set sc = $$$ADDSC(sc,..InstallProduction()) quit:$$$ISERR(sc)
        Set sc = $$$ADDSC(sc,..GenerateMessages()) quit:$$$ISERR(sc)
        Set sc = $$$ADDSC(sc,..GenerateUsingEnsDirector()) quit:$$$ISERR(sc)
    }
    Catch (err) {
        Set sc = $$$ADDSC(sc,err.AsStatus())
    }
    Return sc
}
```

El ClassMethod **InstallProduction** reúne la estructura principal que permite crear una producción mediante la creación de:

- una solicitud
- una *business operation*
- un *business service*
- una producción de interoperabilidad

Dado que la idea es crear una producción de interoperabilidad mediante código, vamos al modo de codificación completa para crear todas las clases para la solicitud, la *business operation* y los *business services*. Al hacer eso, haremos un amplio uso de algunos paquetes de librerías de InterSystems:

- %Dictionary.ClassDefinition
- %Dictionary.PropertyDefinition
- %Dictionary.XDataDefinition
- %Dictionary.MethodDefinition
- %Dictionary.ParameterDefinition

El ClassMethod **InstallProduction** crea dos clases que se extienden desde **Ens.Request**, por medio de las siguientes líneas:

```
Set sc = $$$ADDSC(sc,..CreateRequest("diashenrique.messageviewer.Message.SimpleRequest","Message")) quit:$$$ISERR(sc)
```

```
Set sc = $$$ADDSC(sc,..CreateRequest("diashenrique.messageviewer.Message.AnotherRequest","Something")) quit:$$$ISERR(sc)
```

```
ClassMethod CreateRequest(classname As %String, prop As %String) As %Status [ Private  
]
```

```
{  
  
    New $Namespace  
  
    Set $Namespace = ..#NAMESPACE  
  
    Set sc = $$$OK  
  
    Try {  
  
        Set class = ##class(%Dictionary.ClassDefinition).%New(classname)  
  
        Set class.GeneratedBy = $ClassName()  
  
        Set class.Super = "Ens.Request"  
  
        Set class.ProcedureBlock = 1  
  
        Set class.Inheritance = "left"  
  
        Set sc = $$$ADDSC(sc,class.%Save())  
  
        #; create adapter  
  
        Set property = ##class(%Dictionary.PropertyDefinition).%New(classname)  
  
        Set property.Name = prop  
  
        Set property.Type = "%String"  
  
        Set sc = $$$ADDSC(sc,property.%Save())  
  
        Set sc = $$$ADDSC(sc,$System.OBJ.Compile(classname,"fck-dv"))  
  
    }  
  
    Catch (err) {  
  
        Set sc = $$$ADDSC(sc,err.AsStatus())  
  
    }  
  
    Return sc  
  
}
```

Ahora vamos a crear la clase para una *business operation* que se extiende desde **Ens.BusinessOperation**:

```
Set sc = $$$ADDSC(sc,..CreateOperation()) quit:$$$ISERR(sc)
```

Además de crear la clase, creamos MessageMap y el método Consume:

```
ClassMethod CreateOperation() As %Status [ Private ]
{
    New $Namespace
    Set $Namespace = ..#NAMESPACE
    Set sc = $$$OK
    Try {
        Set classname = "diashenrique.messageviewer.Operation.Consumer"
        Set class = ##class(%Dictionary.ClassDefinition).%New(classname)
        Set class.GeneratedBy = $ClassName()
        Set class.Super = "Ens.BusinessOperation"
        Set class.ProcedureBlock = 1
        Set class.Inheritance = "left"

        Set xdata = ##class(%Dictionary.XDataDefinition).%New()
        Set xdata.Name = "MessageMap"
        Set xdata.XMLNamespace = "http://www.intersystems.com/urlmap"
        Do xdata.Data.WriteLine("<MapItems>")
        Do xdata.Data.WriteLine("<MapItem MessageType=\"\"diashenrique.messageviewer.Me
ssage.SimpleRequest\"\">")
        Do xdata.Data.WriteLine("<Method>Consume</Method>")
        Do xdata.Data.WriteLine("</MapItem>")
        Do xdata.Data.WriteLine("<MapItem MessageType=\"\"diashenrique.messageviewer.Me
ssage.AnotherRequest\"\">")
        Do xdata.Data.WriteLine("<Method>Consume</Method>")
        Do xdata.Data.WriteLine("</MapItem>")
        Do xdata.Data.WriteLine("</MapItems>")
        Do class.XDatas.Insert(xdata)
        Set sc = $$$ADDSC(sc,class.%Save())

        Set method = ##class(%Dictionary.MethodDefinition).%New(classname)
        Set method.Name = "Consume"
        Set method.ClassMethod = 0
        Set method.ReturnType = "%Status"
        Set method.FormalSpec = "input:diashenrique.messageviewer.Message.SimpleReque
st,&output:Ens.Response"
        Set stream = ##class(%Stream.TmpCharacter).%New()
        Do stream.WriteLine("    set sc = $$$OK")
        Do stream.WriteLine("    $$$TRACE(input.Message)")
        Do stream.WriteLine("    return sc")
        Set method.Implementation = stream
        Set sc = $$$ADDSC(sc,method.%Save())

        Set sc = $$$ADDSC(sc,$System.OBJ.Compile(classname,"fck-dv"))
    }
    Catch (err) {
        Set sc = $$$ADDSC(sc,err.AsStatus())
    }
    Return sc
}
```

En el último paso antes de realmente crear la producción de interoperabilidad, vamos a crear la clase responsable del *business service*:

```
Set sc = $$$ADDSC(sc,..CreateRESTService()) quit:$$$ISERR(sc)
```

Esta clase tiene UrlMap y Routes para recibir solicitudes Http.

```
ClassMethod CreateRESTService() As %Status [ Private ]
{
    New $Namespace
    Set $Namespace = ..#NAMESPACE
    Set sc = $$$OK
    Try {
        Set classname = "diashenrique.messageviewer.Service.REST"
        Set class = ##class(%Dictionary.ClassDefinition).%New(classname)
        Set class.GeneratedBy = $ClassName()
        Set class.Super = "EnsLib.REST.Service, Ens.BusinessService"
        Set class.ProcedureBlock = 1
        Set class.Inheritance = "left"

        Set xdata = ##class(%Dictionary.XDataDefinition).%New()
        Set xdata.Name = "UrlMap"
        Set xdata.XMLNamespace = "http://www.intersystems.com/urlmap"
        Do xdata.Data.WriteLine("<Routes>")
        Do xdata.Data.WriteLine("<Route Url=''/send/message'' Method='''POST'' Call='''
SendMessage'''/>")
        Do xdata.Data.WriteLine("<Route Url=''/send/something'' Method='''POST'' Call=
''SendSomething'''/>")
        Do xdata.Data.WriteLine("</Routes>")
        Do class.XDatas.Insert(xdata)
        Set sc = $$$ADDSC(sc,class.%Save())

        #; create adapter
        Set adapter = ##class(%Dictionary.ParameterDefinition).%New(classname)
        Set class.GeneratedBy = $ClassName()
        Set adapter.Name = "ADAPTER"
        Set adapter.SequenceNumber = 1
        Set adapter.Default = "EnsLib.HTTP.InboundAdapter"
        Set sc = $$$ADDSC(sc,adapter.%Save())

        #; add prefix
        Set prefix = ##class(%Dictionary.ParameterDefinition).%New(classname)
        Set prefix.Name = "EnsServicePrefix"
        Set prefix.SequenceNumber = 2
        Set prefix.Default = "|demoiris"
        Set sc = $$$ADDSC(sc,prefix.%Save())

        Set method = ##class(%Dictionary.MethodDefinition).%New(classname)
        Set method.Name = "SendMessage"
        Set method.ClassMethod = 0
        Set method.ReturnType = "%Status"
        Set method.FormalSpec = "input:%Library.AbstractStream,&output:%Stream.Object
"

        Set stream = ##class(%Stream.TmpCharacter).%New()
        Do stream.WriteLine("    set sc = $$$OK")
        Do stream.WriteLine("    set request = ##class(diashenrique.messageviewer.Mess
age.SimpleRequest).%New()")
        Do stream.WriteLine("    set data = {}.%FromJSON(input)")
        Do stream.WriteLine("    set request.Message = data.Message")
        Do stream.WriteLine("    set sc = $$$ADDSC(sc,..SendRequestSync(''diashenrique
.messageviewer.Operation.Consumer'',request,.response))")
    }
}
```

```
Do stream.WriteLine("    return sc")
Set method.Implementation = stream
Set sc = $$$ADDSC(sc,method.%Save())

Set method = ##class(%Dictionary.MethodDefinition).%New(classname)
Set method.Name = "SendSomething"
Set method.ClassMethod = 0
Set method.ReturnType = "%Status"
Set method.FormalSpec = "input:%Library.AbstractStream,&output:%Stream.Object
"

Set stream = ##class(%Stream.TmpCharacter).%New()
Do stream.WriteLine("    set sc = $$$OK")
Do stream.WriteLine("    set request = ##class(diashenrique.messageviewer.Mess
age.AnotherRequest).%New()")
Do stream.WriteLine("    set data = {}.%FromJSON(input)")
Do stream.WriteLine("    set request.Something = data.Something")
Do stream.WriteLine("    set sc = $$$ADDSC(sc,..SendRequestSync(\"\"diashenrique
.messageviewer.Operation.Consumer\"\",request,.response))")
Do stream.WriteLine("    return sc")
Set method.Implementation = stream
Set sc = $$$ADDSC(sc,method.%Save())

Set sc = $$$ADDSC(sc,$System.OBJ.Compile(classname,"fck-dv"))
}
Catch (err) {
    Set sc = $$$ADDSC(sc,err.AsStatus())
}
Return sc
}
```

Usando Visual Studio Code

Crear las clases mediante el paquete %Dictionary puede ser difícil, y difícil de leer también, pero es práctico. Para mostrar un procedimiento un poco más sencillo con una mejor comprensión del código, también crearé nuevas clases de solicitud, *business service* y *business operations* por medio de Visual Studio Code:

- diashenrique.messageviewer.Message.SimpleMessage.cls
- diashenrique.messageviewer.Operation.ConsumeMessageClass.cls
- diashenrique.messageviewer.Service.SendMessage.cls

```
Class diashenrique.messageviewer.Message.SimpleMessage Extends Ens.Request [ Inheritance = left, ProcedureBlock ]
{
Property ClassMessage As %String;
}
```

```
Class diashenrique.messageviewer.Operation.ConsumeMessageClass Extends Ens.BusinessOperation [ Inheritance = left, ProcedureBlock ]
{
Method Consume(input As diashenrique.messageviewer.Message.SimpleMessage, ByRef output As Ens.Response) As %Status
{
    Set sc = $$$OK
    $$$TRACE(pRequest.ClassMessage)
    Return sc
}
```

```
}
XData MessageMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
  <MapItems>
    <MapItem MessageType="diashenrique.messageviewer.Message.SimpleMessage">
      <Method>Consume</Method>
    </MapItem>
  </MapItems>
}
}
```

```
Class diashenrique.messageviewer.Service.SendMessage Extends Ens.BusinessService [ ProcedureBlock ]
{
Method OnProcessInput(input As %Library.AbstractStream, ByRef output As %Stream.Object) As %Status
{
  Set tSC = $$$OK
  // Create the request message
  Set request = ##class(diashenrique.messageviewer.Message.SimpleMessage).%New()
  // Place a value in the request message property
  Set request.ClassMessage = input
  // Make a synchronous call to the business process and use the response message as our response
  Set tSC = ..SendRequestSync("diashenrique.messageviewer.Operation.ConsumeMessageClass",request,.output)
  Quit tSC
}
}
```

Desde el punto de vista de la legibilidad del código, la diferencia es enorme.

Creando la Producción de Interoperabilidad

Vamos a terminar la creación de nuestra producción de interoperabilidad. Para ello, crearemos una clase de producción, y después la asociaremos con las *Business Operation* y *Business Services*.

```
Set sc = $$$ADDSC(sc,..CreateProduction()) quit:$$$ISERR(sc)

ClassMethod CreateProduction(purge As %Boolean = 0) As %Status [ Private ]
{
  New $Namespace
  Set $Namespace = ..#NAMESPACE
  Set sc = $$$OK
  Try {
    #; create new production
    Set class = ##class(%Dictionary.ClassDefinition).%New(..#PRODUCTION)
    Set class.ProcedureBlock = 1
    Set class.Super = "Ens.Production"
    Set class.GeneratedBy = $ClassName()
    Set xdata = ##class(%Dictionary.XDataDefinition).%New()
    Set xdata.Name = "ProductionDefinition"
    Do xdata.Data.Write("<Production Name=""_..#PRODUCTION_"" LogGeneralTraceEvents=""true""></Production>")
    Do class.XDatas.Insert(xdata)
    Set sc = $$$ADDSC(sc,class.%Save())
    Set sc = $$$ADDSC(sc,$System.OBJ.Compile(..#PRODUCTION,"fck-dv"))
  }
}
```

```
Set production = ##class(Ens.Config.Production).%OpenId(..#PRODUCTION)
Set item = ##class(Ens.Config.Item).%New()
Set item.ClassName = "diashenrique.messageviewer.Service.REST"
Do production.Items.Insert(item)
Set sc = $$$ADDSC(sc,production.%Save())
Set item = ##class(Ens.Config.Item).%New()
Set item.ClassName = "diashenrique.messageviewer.Operation.Consumer"
Do production.Items.Insert(item)
Set sc = $$$ADDSC(sc,production.%Save())
Set item = ##class(Ens.Config.Item).%New()
Set item.ClassName = "diashenrique.messageviewer.Service.SendMessage"
Do production.Items.Insert(item)
Set sc = $$$ADDSC(sc,production.%Save())
Set item = ##class(Ens.Config.Item).%New()
Set item.ClassName = "diashenrique.messageviewer.Operation.ConsumeMessageClas
s"
Do production.Items.Insert(item)
Set sc = $$$ADDSC(sc,production.%Save())
}
Catch (err) {
    Set sc = $$$ADDSC(sc,err.AsStatus())
}
Return sc
}
```

Utilizamos la clase **Ens.Config.Item** para asociar la clase de producción con las clases *Business Operation* y *Business Service*. Puedes hacer esto tanto si creaste tu clase usando el paquete %Dictionary como con VS Code, Studio o Atelier.

En cualquier caso, ¡lo conseguimos! Creamos una producción de interoperabilidad por medio de código.

Pero recuerda el propósito original de todo este código: crear una producción y mensajes para mostrar las funcionalidades del Visualizador de Mensajes mejorado. Usando los métodos de clase siguientes, ejecutaremos los dos *business services* y generaremos los mensajes.

Generando mensajes con %Net.HttpRequest:

```
ClassMethod GenerateMessages() As %Status [ Private ]
{
    New $Namespace
    Set $Namespace = ..#NAMESPACE
    Set sc = $$$OK
    Try {
        Set action(0) = "/demoiris/send/message"
        Set action(1) = "/demoiris/send/something"
        For i=1:1:..#LIMIT {
            Set content = { }
            Set content.Message = "Hi, I'm just a random message named "_$Random(3000
0)
            Set content.Something = "Hi, I'm just a random something named "_$Random(
30000)

            Set httprequest = ##class(%Net.HttpRequest).%New()
            Set httprequest.SSLCheckServerIdentity = 0
            Set httprequest.SSLConfiguration = ""
            Set httprequest.Https = 0
            Set httprequest.Server = "localhost"
            Set httprequest.Port = 9980
            Set serverUrl = action($Random(2))
```



```
        Do httpRequest.EntityBody.Write(content.%ToJSON())
        Set sc = httpRequest.Post(serverUrl)
        Quit:$$$ISERR(sc)
    }
}
Catch (err) {
    Set sc = $$$ADDSC(sc,err.AsStatus())
}
Return sc
}
```

Generando mensajes con EnsDirector:

```
ClassMethod GenerateUsingEnsDirector() As %Status [ Private ]
{
    New $Namespace
    Set $Namespace = ..#NAMESPACE
    Set sc = $$$OK
    Try {
        For i=1:1:..#LIMIT {
            Set tSC = ##class(Ens.Director).CreateBusinessService("diashenrique.messageviewer.Service.SendMessage",.tService)
            Set message = "Message Generated By CreateBusinessService "_$Random(1000)
            Set tSC = tService.ProcessInput(message,.output)
            Quit:$$$ISERR(sc)
        }
    }
    Catch (err) {
        Set sc = $$$ADDSC(sc,err.AsStatus())
    }
    Return sc
}
}
```

Eso todo lo que se necesita para el código. Encontrarás el proyecto completo en <https://github.com/diashenrique/iris-message-viewer>.

Ejecutando el proyecto

Ahora veamos el proyecto en marcha.

Primero, git clone o git pull el repositorio en cualquier directorio local:

```
git clone https://github.com/diashenrique/iris-message-viewer.git
```

Después, abre el terminal en este directorio y ejecuta:

```
docker-compose build
```

Finalmente, ejecuta el contenedor IRIS con tu proyecto:

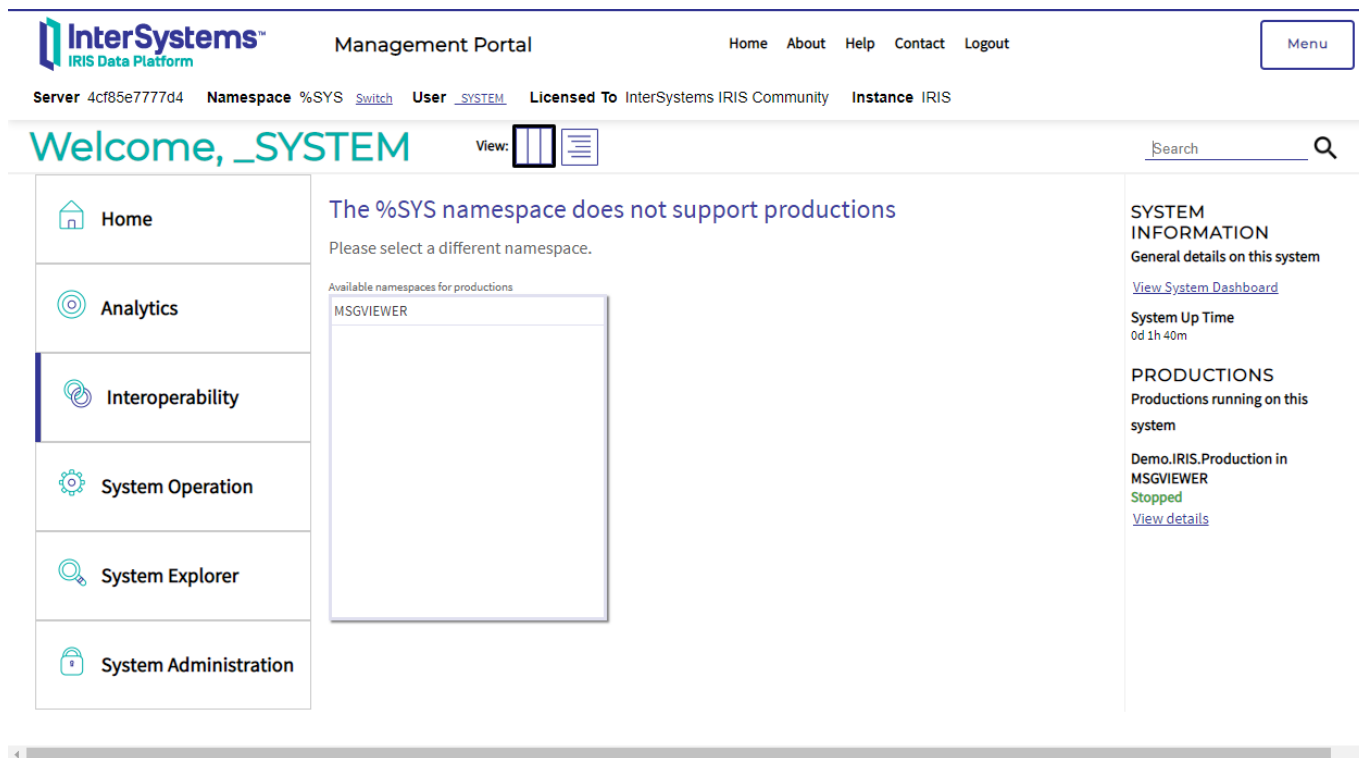
```
docker-compose up -d
```

Ahora accederemos al Portal de Administración por medio de la página <http://localhost:52773/csp/sys/UtilHome.csp>. Deberás ver nuestro *namespace* de interoperabilidad MSGVIEWER,

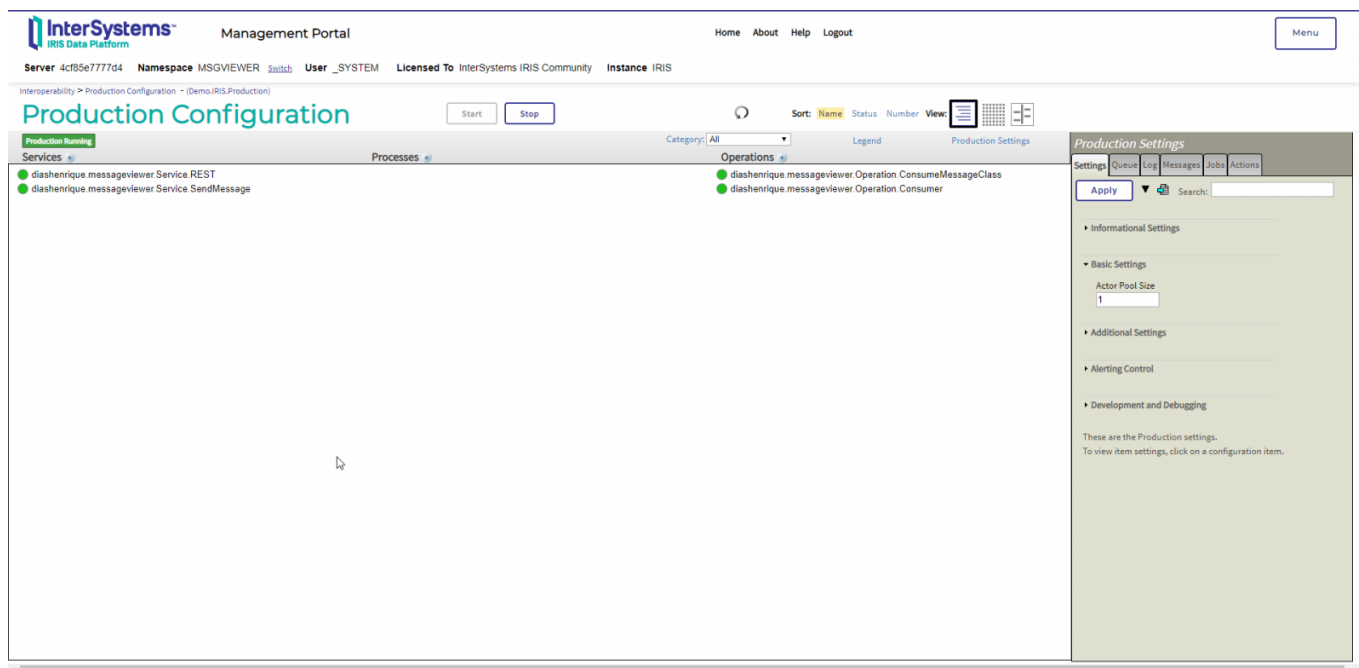
Creando un Visualizador de Mensajes alternativo en IRIS

Published on InterSystems Developer Community (<https://community.intersystems.com>)

como en esta imagen:



Esta es nuestra pequeña producción, con dos *business services* y dos *business operations*:



Tenemos muchos mensajes:

Creando un Visualizador de Mensajes alternativo en IRIS

Published on InterSystems Developer Community (<https://community.intersystems.com>)

InterSystems Management Portal

Server 4cf85e777d4 Namespace MSGVIEWER User _SYSTEM Licensed To InterSystems IRIS Community Instance IRIS

Interoperability > Message Viewer

Search Cancel Reset Resend Previous Next Export

Sort Order: Newest First Page Size: 100 Time Format: Complete Page: 1

Basic Criteria

Status: All Type: Session Start Start Time: Start ID: End Time: End ID: Source: Target:

Extended Criteria

Saved Searches

#	ID	Time Created	Session	Status	Error	Source	Target
1	1202	2020-01-30 15:16:25.522	1202	Completed	OK	Ens.ScheduleService	Ens.ScheduleHandler
2	1200	2020-01-29 19:54:42.766	1200	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
3	1198	2020-01-29 19:54:42.765	1198	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
4	1196	2020-01-29 19:54:42.763	1196	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
5	1194	2020-01-29 19:54:42.762	1194	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
6	1192	2020-01-29 19:54:42.761	1192	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
7	1190	2020-01-29 19:54:42.759	1190	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
8	1188	2020-01-29 19:54:42.758	1188	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
9	1186	2020-01-29 19:54:42.757	1186	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
10	1184	2020-01-29 19:54:42.756	1184	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
11	1182	2020-01-29 19:54:42.755	1182	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
12	1180	2020-01-29 19:54:42.754	1180	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
13	1178	2020-01-29 19:54:42.753	1178	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
14	1176	2020-01-29 19:54:42.752	1176	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
15	1174	2020-01-29 19:54:42.751	1174	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
16	1172	2020-01-29 19:54:42.750	1172	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
17	1170	2020-01-29 19:54:42.749	1170	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
18	1168	2020-01-29 19:54:42.748	1168	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
19	1166	2020-01-29 19:54:42.747	1166	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
20	1164	2020-01-29 19:54:42.746	1164	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
21	1162	2020-01-29 19:54:42.745	1162	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
22	1160	2020-01-29 19:54:42.743	1160	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
23	1158	2020-01-29 19:54:42.742	1158	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
24	1156	2020-01-29 19:54:42.741	1156	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
25	1154	2020-01-29 19:54:42.740	1154	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
26	1152	2020-01-29 19:54:42.739	1152	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
27	1150	2020-01-29 19:54:42.738	1150	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage
28	1148	2020-01-29 19:54:42.737	1148	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessage

Header Body Contents Trace

<ObjectID> 1200
<TargetQueueName> diashenrique.messageviewer.Operation.ConsumeMessage
<Type> Request
<Invocation> Queue
<CorrespondingMessageId> 1201
<SessionId> 1200
<SourceConfigName> diashenrique.messageviewer.Service.SendMessage
<TargetConfigName> diashenrique.messageviewer.Operation.ConsumeMessage
<SourceBusinessType> BusinessService
<TargetBusinessType> BusinessOperation
<BusinessProcessId> SyncCall365
<ReturnQueueName> diashenrique.messageviewer.Message.SimpleMessage
<MessageBodyClassName> diashenrique.messageviewer.Message.SimpleMessage
<MessageBodyId> 600
<Description>
<SuperSession>
<Resent>
<Priority> Sync
<TimeCreated> 2020-01-29 19:54:42.766
<TimeProcessed> 2020-01-29 19:54:42.766
<Status> Completed
<IsError?> 0
<ErrorStatus> OK
<Banded> 0

Con todo en marcha en nuestro Visualizador de Mensajes personalizado, vamos a echar un vistazo a algunas de sus funcionalidades.

El Visualizador de Mensajes mejorado

Ten en cuenta que solo se mostrarán los *namespaces* que estén habilitados para las producciones de interoperabilidad.

<http://localhost:52773/csp/msgviewer/messageviewer.csp>

InterSystems Management Portal

Interoperability > Message Viewer

Server 4cf85e777d4 | Instance IRIS

Message Viewer

Choose Your Namespace

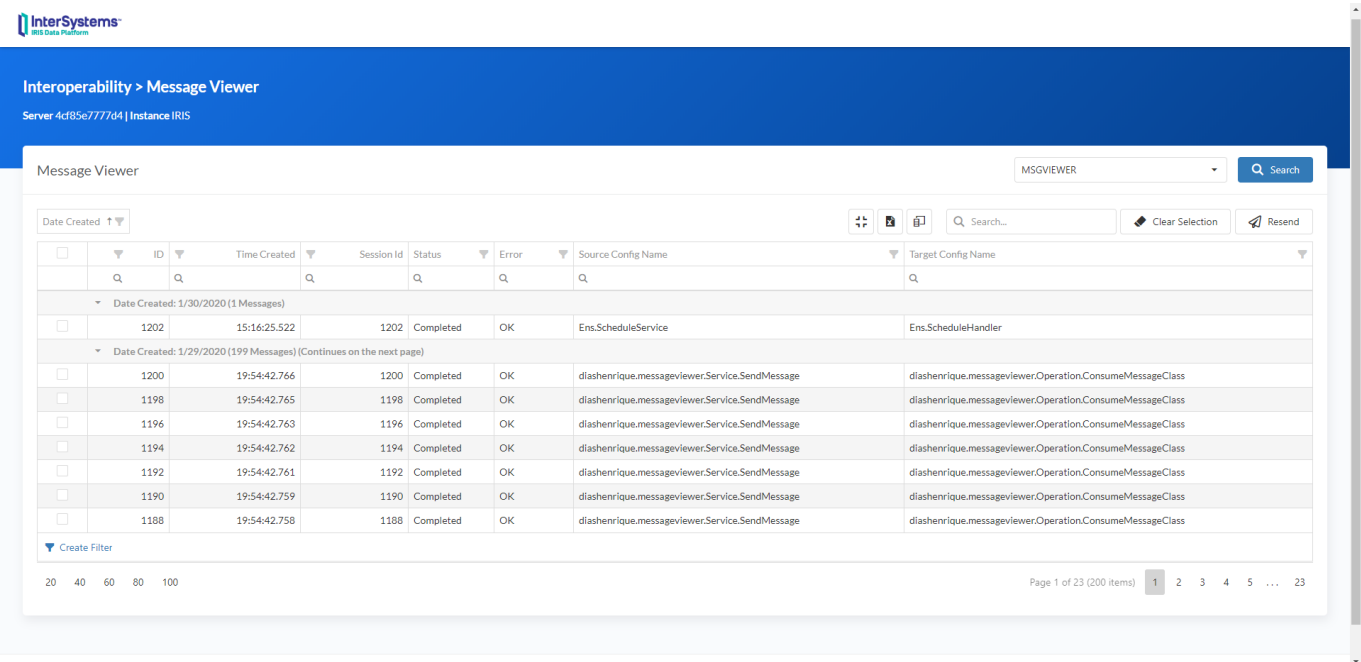
MSGVIEWER

Search

Management Portal

2020 © by Henrique Dias

El Visualizador de Mensajes mejorado incorpora funcionalidades y la flexibilidad que permite crear diferentes filtros, agrupar las columnas en n niveles, exportar a Excel y muchas cosas más.



Se pueden utilizar diferentes filtros para conseguir los resultados que se necesitan. También se pueden ordenar las columnas de forma múltiple pulsando la tecla Mayúsculas y haciendo clic en el encabezado de las columnas. ¡Incluso se puede exportar la tabla de datos a Excel!

ID	Date Created	Time Created	Session Id
1202	1/30/2020	15:16:25.522	1202
1200	1/29/2020	19:54:42.766	1200
1198	1/29/2020	19:54:42.765	1198
1196	1/29/2020	19:54:42.763	1196
1194	1/29/2020	19:54:42.762	1194
1192	1/29/2020	19:54:42.761	1192
1190	1/29/2020	19:54:42.759	1190
1188	1/29/2020	19:54:42.758	1188

Source Config Name ↑▼

☐	▼ ID	Date Created	▼	▼	Time Created	▼	Session Id
	🔍	🔍	📅	🔍		🔍	
▼ Source Config Name: Ens.ScheduleService (1 Messages)							
☐	1202	1/30/2020			15:16:25.522		1202
▼ Source Config Name: diashenrique.messageviewer.service.SendMessage (199 Messages)							
☐	1200	1/29/2020			19:54:42.766		1200
☐	1198	1/29/2020			19:54:42.765		1198
☐	1196	1/29/2020			19:54:42.763		1196
☐	1194	1/29/2020			19:54:42.762		1194
☐	1192	1/29/2020			19:54:42.761		1192
☐	1190	1/29/2020			19:54:42.759		1190
☐	1188	1/29/2020			19:54:42.758		1188

Además, se pueden crear filtros complejos con la opción Generador de filtros.

Se pueden agrupar los datos con cualquier columna disponible, agrupando la información mediante los n niveles que se desee. De forma predeterminada, el grupo se establece a partir del campo Fecha de creación (Date Created).

Creando un Visualizador de Mensajes alternativo en IRIS

Published on InterSystems Developer Community (<https://community.intersystems.com>)



Interoperability > Message Viewer

Server 4cf85e777d4 | Instance IRIS

Message Viewer

MSGVIEWER

Search

Date Created ↑

Search...

Clear Selection

Resend

☐	ID	Time Created	Session Id	Status	Error	Source Config Name	Target Config Name
	Q	Q	Q	Q	Q	Q	Q
Date Created: 1/30/2020 (1 Messages)							
☐	1202	15:16:25.522	1202	Completed	OK	Ens.ScheduleService	Ens.ScheduleHandler
Date Created: 1/29/2020 (199 Messages) (Continues on the next page)							
☐	1200	19:54:42.766	1200	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1198	19:54:42.765	1198	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1196	19:54:42.763	1196	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1194	19:54:42.762	1194	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1192	19:54:42.761	1192	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1190	19:54:42.759	1190	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC
☐	1188	19:54:42.758	1188	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageC

20 40 60 80 100

Page 1 of 23 (200 items) 1 2 3 4 5 ... 23

Y hay una funcionalidad que permite seleccionar columnas. La siguiente página tiene todas las columnas de Ens.MessageHeader, y muestra solo las columnas predeterminadas en la visualización inicial. Pero se pueden elegir las otras columnas mediante el botón "Seleccionador de columnas" (Column Chooser).



Interoperability > Message Viewer

Server 4cf85e777d4 | Instance IRIS

Message Viewer

MSGVIEWER

Search

Date Created ↑

Search...

Clear Selection

Resend

☐	ID	Time Created	Session Id	Status	Error	Source Config Name	Target Config Name
	Q	Q	Q	Q	Q	Q	Q
Date Created: 1/30/2020 (1 Messages)							
☐	1202	15:16:25.522	1202	Completed	OK	Ens.ScheduleService	Ens.ScheduleHandler
Date Created: 1/29/2020 (199 Messages) (Continues on the next page)							
☐	1200	19:54:42.766	1200	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1198	19:54:42.765	1198	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1196	19:54:42.763	1196	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1194	19:54:42.762	1194	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1192	19:54:42.761	1192	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1190	19:54:42.759	1190	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass
☐	1188	19:54:42.758	1188	Completed	OK	diashenrique.messageviewer.Service.SendMessage	diashenrique.messageviewer.Operation.ConsumeMessageClass

20 40 60 80 100

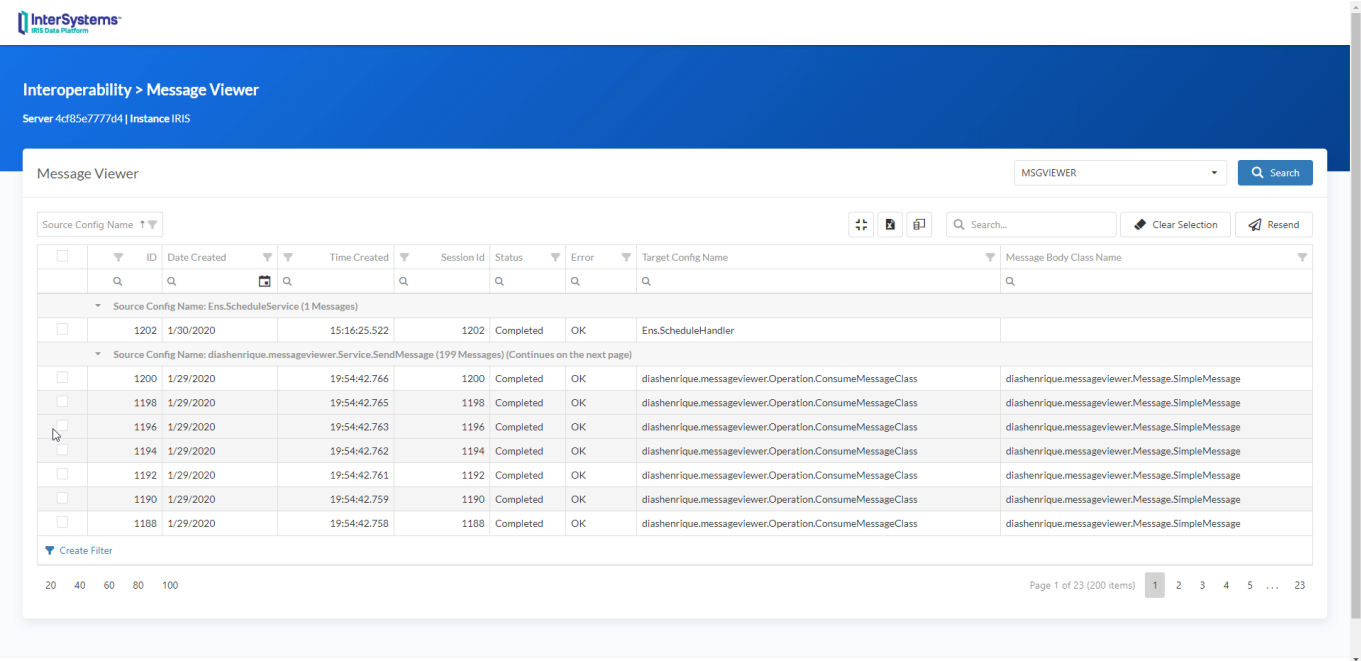
Page 1 of 23 (200 items) 1 2 3 4 5 ... 23

Se pueden contraer o expandir todos los grupos con un solo clic.

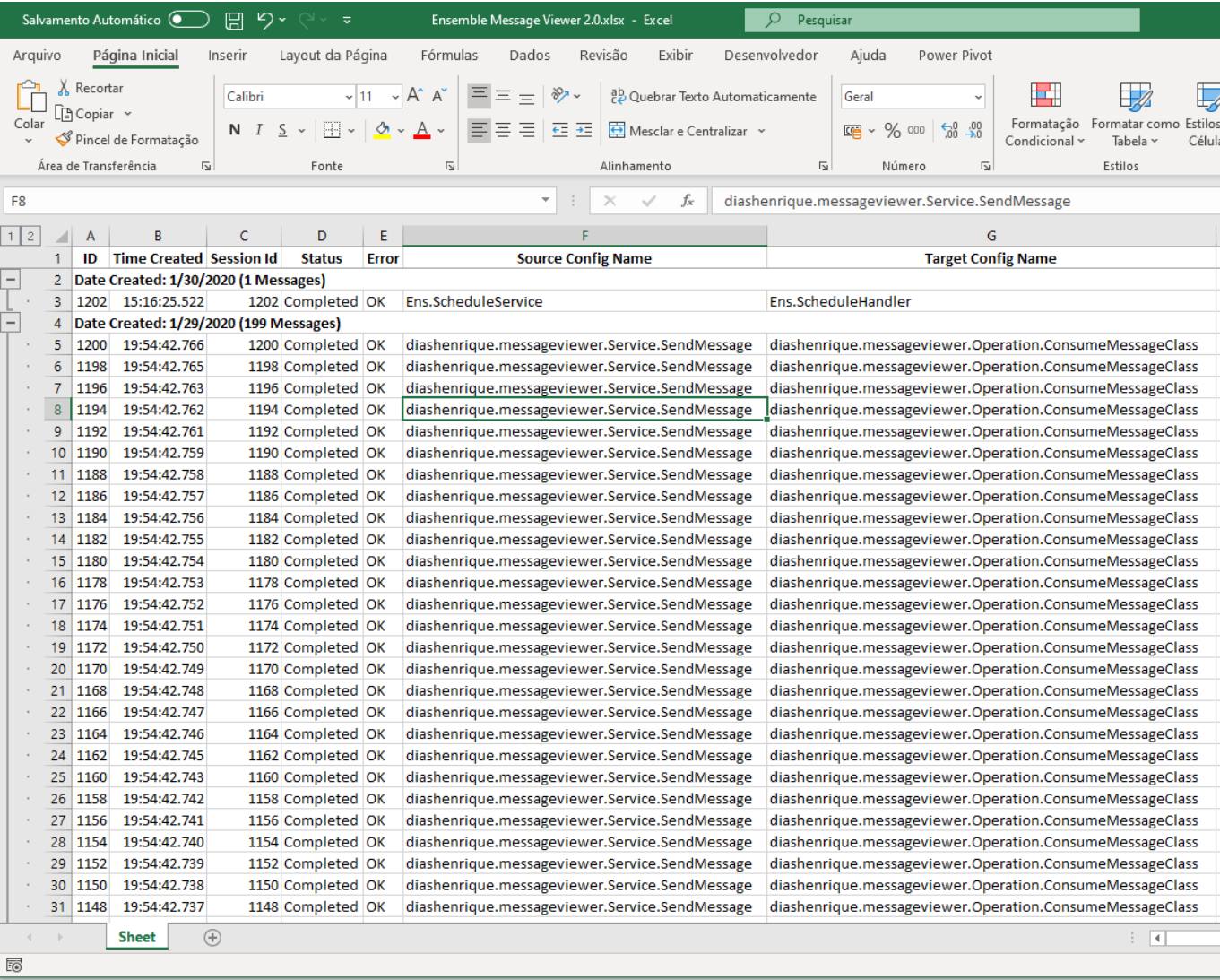
La información del campo SessionID incluye un enlace a la función Visual Trace.

Se pueden reenviar mensajes si se necesita. Tan solo hay que seleccionar los mensajes que se necesitan y hacer clic para volver a enviarlos. Esta función utiliza el siguiente método de clase:

```
##class(Ens.MessageHeader).ResendDuplicatedMessage(id)
```

Por último, como se mencionó, se puede exportar la tabla de datos a Excel:



El resultado en Excel mostrará el mismo formato, contenido y grupo definidos en las páginas del servidor de caché (CSP).

Creando un Visualizador de Mensajes alternativo en IRIS

Published on InterSystems Developer Community (<https://community.intersystems.com>)

P.D.: Quiero dar un agradecimiento especial a [@Renan Lourenco](#), que me ayudó mucho en este viaje.

[#Búsqueda de mensajes](#) [#Docker](#) [#Interoperabilidad](#) [#ObjectScript](#) [#InterSystems Package Manager \(IPM\)](#)
[#Operación empresarial](#) [#Servicio empresarial](#) [#Ensemble](#) [#InterSystems IRIS](#)
[Ir a la aplicación en InterSystems Open Exchange](#)

URL de
fuentes: <https://es.community.intersystems.com/post/creando-un-visualizador-de-mensajes-alternativo-en-iris>