Article
Luis Angel Pére... · Mar 2, 2023 5m read

Asynchronous Socket in IRIS and connection from JavaScript client

I've been working for some days in the connectivity between NodeJS client applications and IRIS as server using web sockets.

You can get all the information in relation to the web socket connections using IRIS as a client or as a server from this URL: https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls...

For this example we are going to configure an asynchronous server, that would be really usefull to create a subscription manager for our productions.

First of all, we have created a really simple persistent object in which we are going to store the session id that we are going to use to identify our connection by socket. This id is assigned by the client automatically.

```
Class User.WebSocketSession Extends %Persistent
{
 Property SessionID As %String;
}
```

As you can see it's really simple. We are going to create it into USER namespace. The main purpose of this object is to get all the opened sockets and send notifications to their clients. In the case that we want to manage different subscription types we can add new properties and make this object as complex as it's required.

Once we have coded our object to manage the socket connections we can start our socket class:

```
Class User.WebSocketServer Extends %CSP.WebSocket
{

Method OnPreServer() As %Status
{
        set webSocketSession = ##class(WebSocketSession).%New()
        set webSocketSession.SessionID = ..WebSocketID
        do webSocketSession.%Save()

        Set ..SharedConnection = 1
        Quit $$$OK
}

Method Server() As %Status
{

   Quit $$$OK
}

Method OnPostServer() As %Status
{
   Quit $$$OK
}
}
```

Easy peasy! Our class extends %CSP.WebSocket and we just have to implement for our example OnPreServer() method, this method will be executed when a connection request is received from our client, a step before to open the connection. In the code of OnPreServer method we are going to create an object from our class WebSocketSession and we'll store the web socket id (WebSocketID). We can add some kind of authentication if it's necessary, in that case we have to implement the Server() method.

You can see in our code that we are setting to 1 a property named SharedConnection, this parameter is used to configure our web socket like an asynchronous socket, it means that we can send messages to the client using our socket connection from any component of our productions.

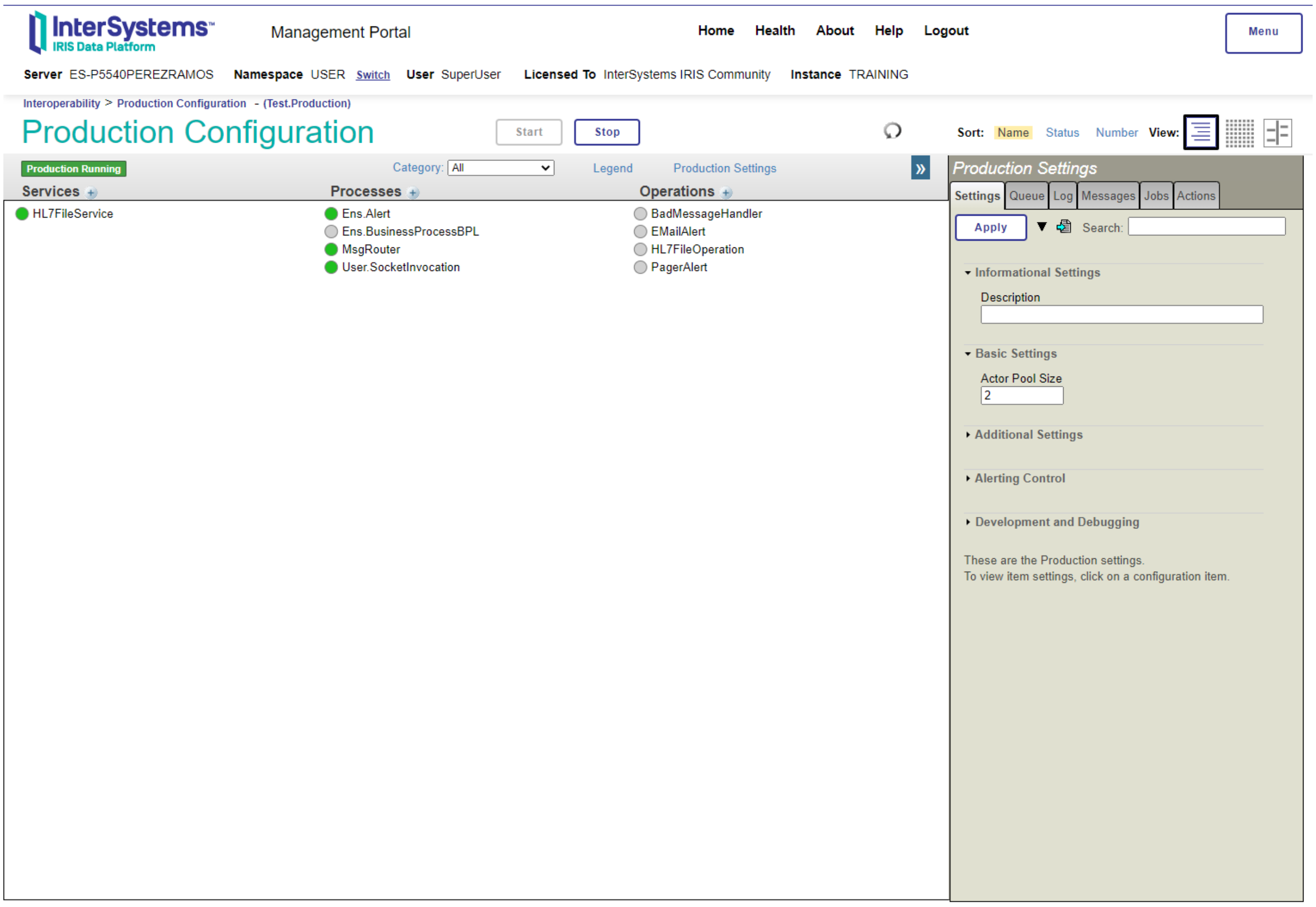
To connect to our socket we are going to use the following URL:
ws(s)://{IRIS_IP}:{IRIS_PORT}/csp/{NAMESPACE}/{SOCKET_CLASS}

In our case the URL is ws://localhost:52774/csp/user/User.WebSocketServer.cls and the calls from our client application will something be like this

```
function socketConnect() {
    socket = new WebSocket("ws://localhost:52774/csp/user/User.WebSocketServer.cls");

    socket.onmessage = function(msg){
        console.log(msg.data);
    };

    var auth = { "User": "_SYSTEM", "Password": "SYS"};

 // we need to wait before connection is established
 setTimeout(function() {
  socket.send(JSON.stringify(auth));
 },1000);
}

function socketClose() {
    socket.close();
}
```

We have added and small message to send to the server an user and a password, just in case that we need to validate the access to the server, in our example we are not going deeper, but I recomend you to send a test message in the connection to be sure that we are properly connected.

No mistery. We have our socker in a class of ObjectScript and this provide to us of an URL to be call from our client directly. The next step will be to start a simple production in IRIS (USER namespace) to check the behaviour of the notifications.


InterSystems IRIS Data Platform
Management Portal
Home Health About Help Logout
Menu
Server ES-P5540PEREZRAMOS Namespace USER Switch User SuperUser Licensed To InterSystems IRIS Community Instance TRAINING
Interoperability > Production Configuration - (Test.Production)
Production Configuration
Start
Stop
Sort: Name Status Number View:
Production Running
Category: All
Legend
Production Settings

Services
- ● HL7FileService

Processes
- ● Ens.Alert
- ● Ens.BusinessProcessBPL
- ● MsgRouter
- ● User.SocketInvocation

Operations
- ● BadMessageHandler
- ● EMailAlert
- ● HL7FileOperation
- ● PagerAlert

Production Settings

Settings | Queue | Log | Messages | Jobs | Actions

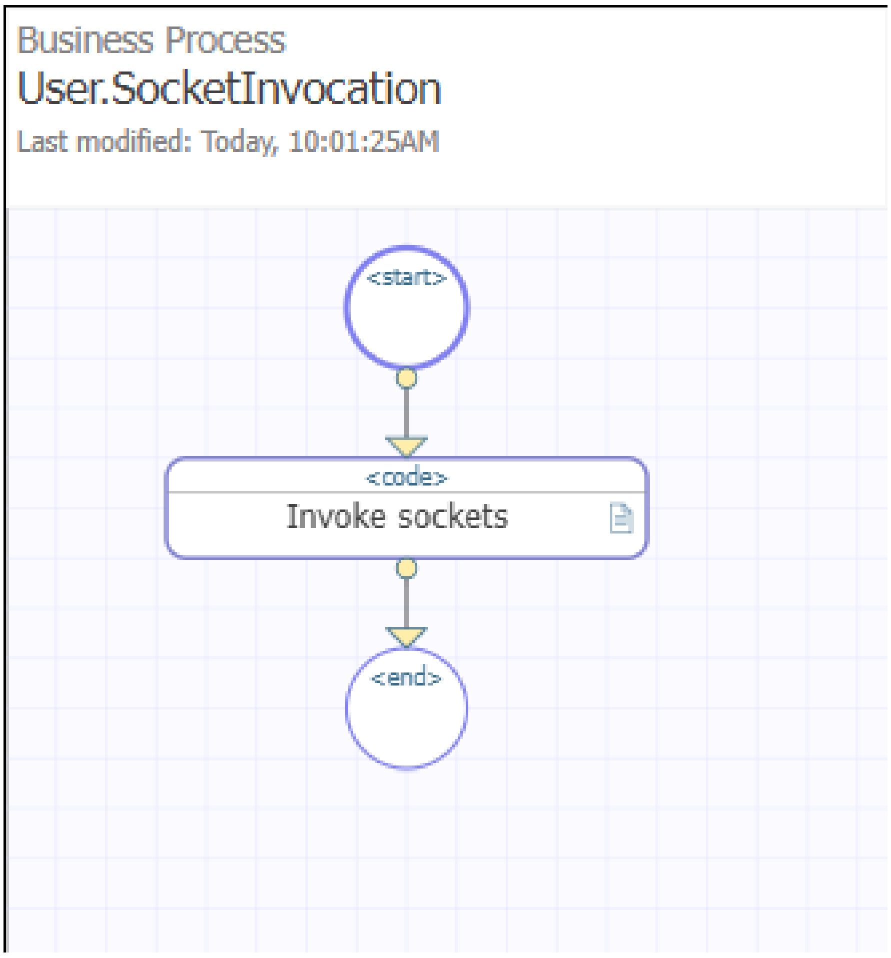
Apply · Search:

- Informational Settings
  - Description:
- Basic Settings
  - Actor Pool Size: 2
- Additional Settings
- Alerting Control
- Development and Debugging

These are the Production settings.
To view item settings, click on a configuration item.

Here is our basic production generated by default, we have added a new Business Process named User.SocketInvocation, this BP will receive an HL7 message as soon as our Business Service HL7FileService detects a new file in a specific folder of our computer.

If we open the BPL definition User.SocketInvocation we are going to see that it's just a Code element.



We can review the code of this element:

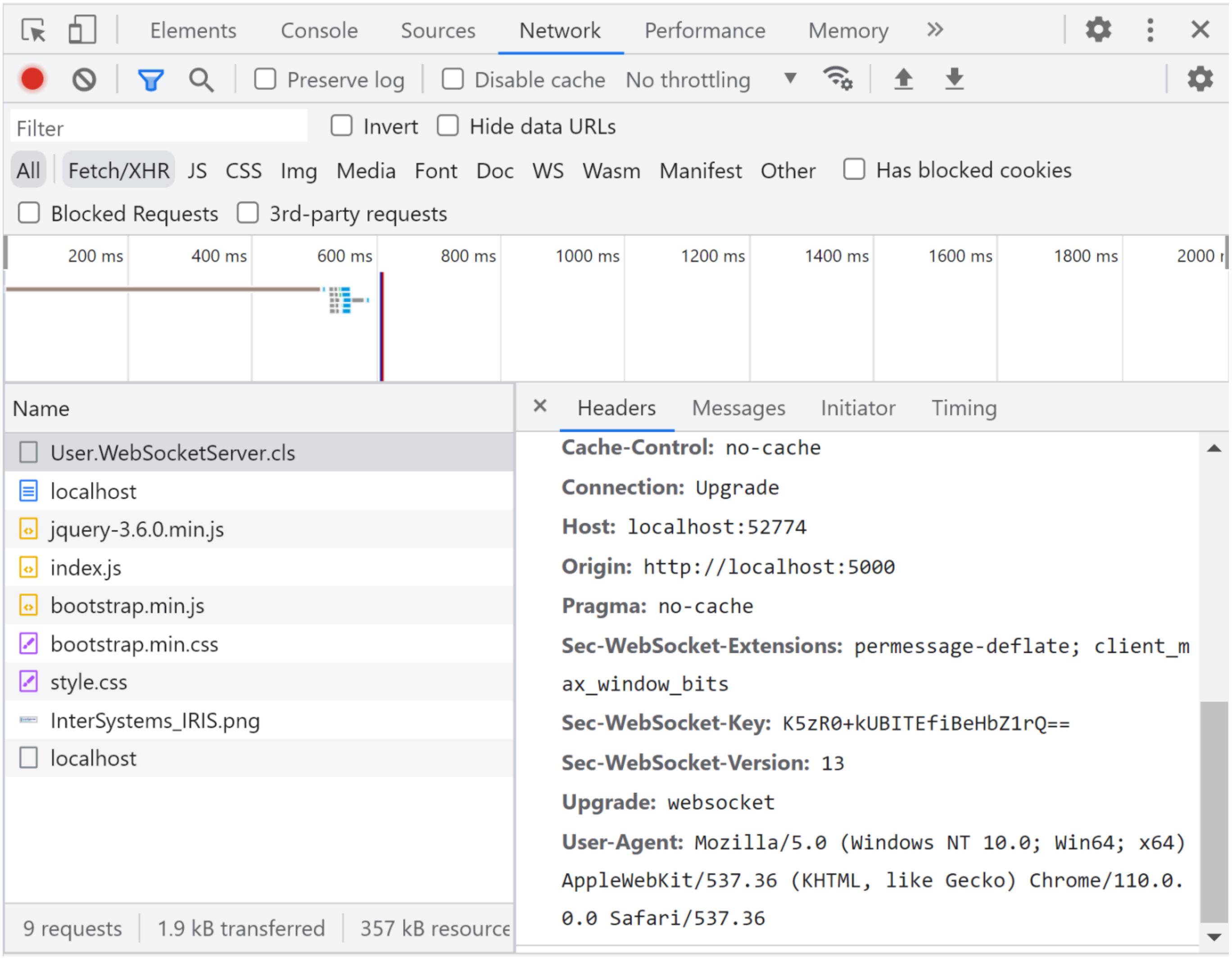
```
// Query to get the open sockets
Set result=##class(%ResultSet).%New("%DynamicQuery:SQL")
Do result.Prepare("SELECT %ID,SessionID FROM WebSocketSession")
Do result.Execute()
// Loop to send notification to each opened socket
while(result.Next()) {
   try {
       Set ws=##class(%CSP.WebSocket).%New()
       Set tSC = ws.OpenServer(result.Data("SessionID"))
       // testing if socket is opened
       Set data= ws.Read(, .status, )
       If $$$ISERR(status) {
          If $$$GETERRORCODE(status) = $$$CSPWebSocketClosed {
            $$$LOGINFO("The socket is closed")
          }
          If $$$GETERRORCODE(status) = $$$CSPWebSocketTimeout {
       $$$LOGINFO("The socket is in timeout")
          }
          // if socket is closed, delete it from the database
          set sqltext = "DELETE FROM WebSocketSession WHERE SessionID = ?"
         set tStatement = ##class(%SQL.Statement).%New()
         set qStatus = tStatement.%Prepare(sqltext)
        if qStatus'=1 {
        $$$LOGINFO("Error in sql for deleting info")
       }
         set rtn = tStatement.%Execute(result.Data("SessionID"))
```

```
           if rtn.%SQLCODE=0 {
             $$$LOGINFO("Socket deleted succesfully")
            }
           else {
            $$$LOGINFO("Error deleting socket session")
           }
         }
         else {
            //if socket is opened, send a message
            Set tSC = ws.Write("Something has change!")
         }
     } catch err {
         $$$LOGINFO(err.Name)
     }
  }
```

In this piece of code we are doing the following operations:

1. Getting the list of open sockets.
2. Checking the status of each socket stored.
    1. If the socket is already closed we delete it from our WebSocketSession table.
    2. If the socket is open we just send a message to our client.

Now we are going to review how it works opening a connection to our socket from the client application:

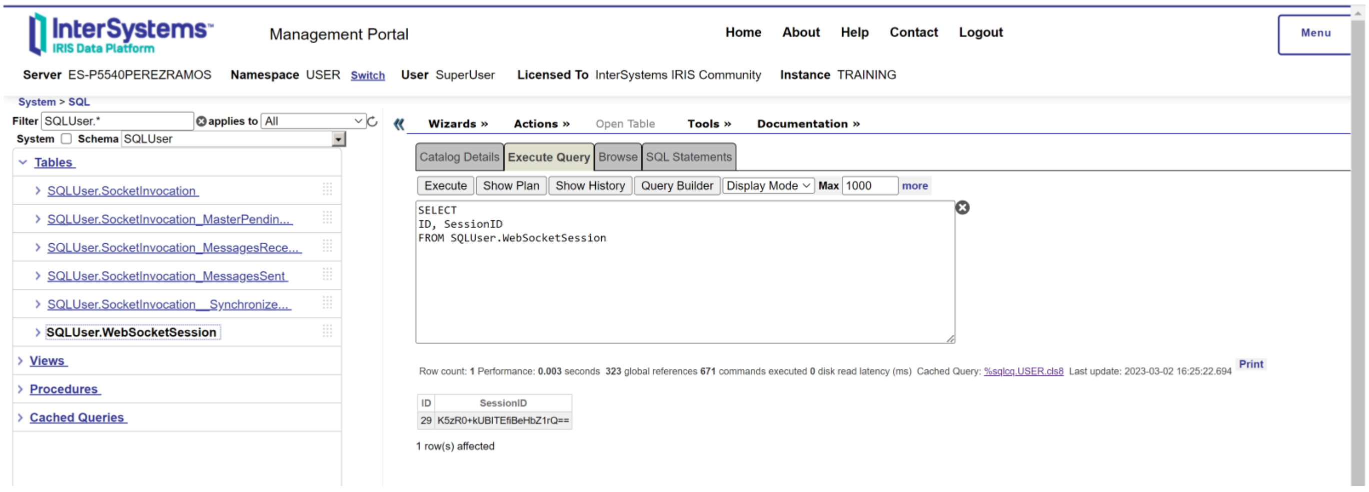


User.WebSocketServer.cls
localhost
jquery-3.6.0.min.js
index.js
bootstrap.min.js
bootstrap.min.css
style.css
InterSystems_IRIS.png
localhost

9 requests | 1.9 kB transferred | 357 kB resource

Cache-Control: no-cache
Connection: Upgrade
Host: localhost:52774
Origin: http://localhost:5000
Pragma: no-cache
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Key: K5zR0+kUBITEfiBeHbZ1rQ==
Sec-WebSocket-Version: 13
Upgrade: websocket
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36

If it works right in our socket we could see in our WebSocketServer table a new record with the socket identifier K5zR0+kUBITEfiBeHbZ1rQ== (Sec-WebSocket-Key field).



```
SELECT
ID, SessionID
FROM SQLUser.WebSocketSession
```

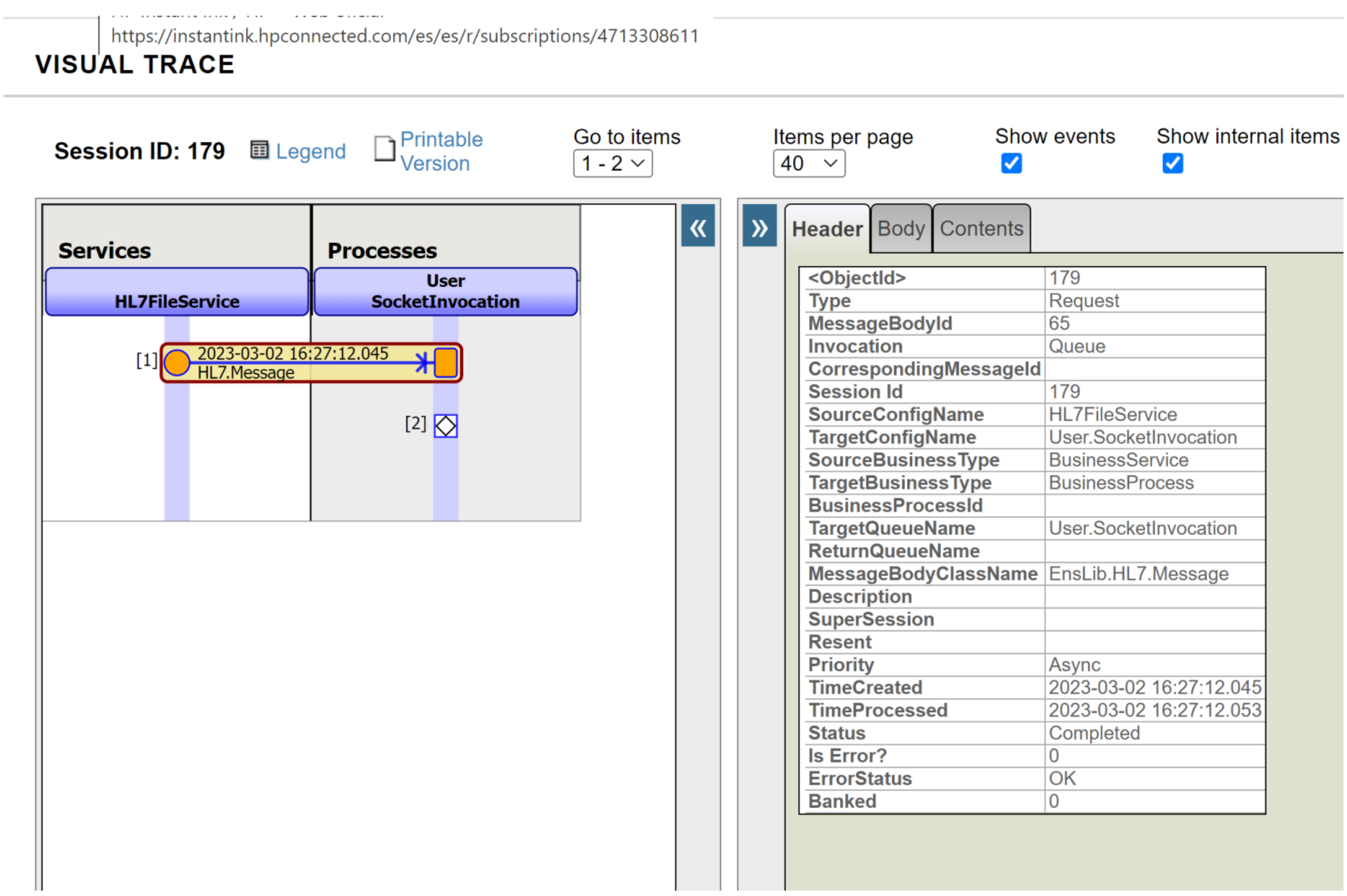
Row count: 1 Performance: 0.003 seconds 323 global references 671 commands executed 0 disk read latency (ms) Cached Query: %sqlcq.USER.cls8 Last update: 2023-03-02 16:25:22.694

ID	SessionID
29	K5zR0+kUBITEfiBeHbZ1rQ==

1 row(s) affected

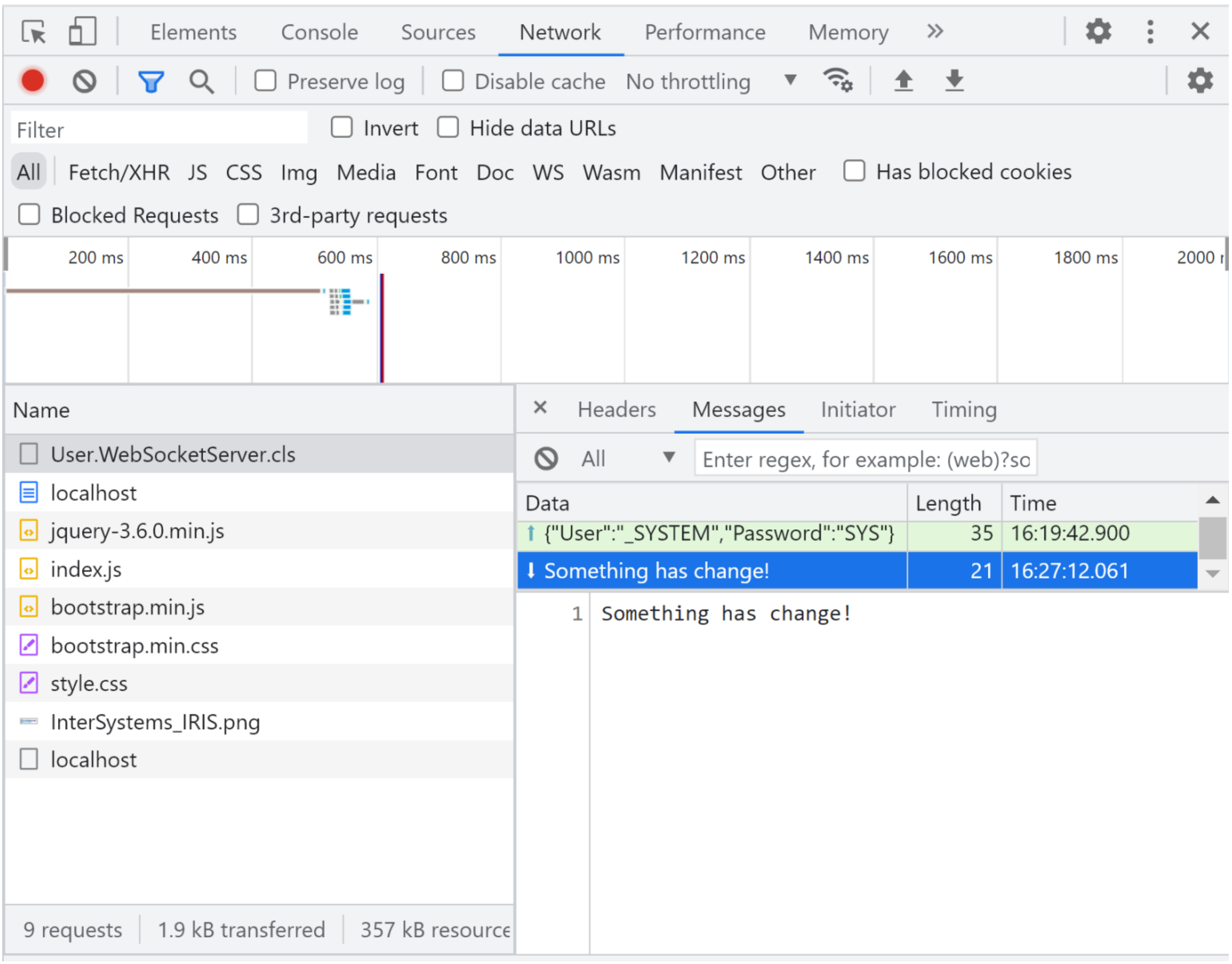
Bingo! Here is our socket id stored, now we can feed our Business Service with any file and check if we receive any notifications into our client.

This is the message sent from the Business Service into our Business Operation:



<ObjectId>	179
Type	Request
MessageBodyId	65
Invocation	Queue
CorrespondingMessageId	
Session Id	179
SourceConfigName	HL7FileService
TargetConfigName	User.SocketInvocation
SourceBusinessType	BusinessService
TargetBusinessType	BusinessProcess
BusinessProcessId	
TargetQueueName	User.SocketInvocation
ReturnQueueName	
MessageBodyClassName	EnsLib.HL7.Message
Description	
SuperSession	
Resent	
Priority	Async
TimeCreated	2023-03-02 16:27:12.045
TimeProcessed	2023-03-02 16:27:12.053
Status	Completed
Is Error?	0
ErrorStatus	OK
Banked	0

And we can check our client application:



Name
User.WebSocketServer.cls
localhost
jquery-3.6.0.min.js
index.js
bootstrap.min.js
bootstrap.min.css
style.css
InterSystems_IRIS.png
localhost

9 requests | 1.9 kB transferred | 357 kB resource

Data	Length	Time
↑ {"User":"_SYSTEM","Password":"SYS"}	35	16:19:42.900
↓ Something has change!	21	16:27:12.061

```
Something has change!
```

That is our notification!

Well, that's all. We have configured a simple web socket which receives a connection request from a web application, we have managed a list of open sockets with their identifiers and we have send a notification to those open asynchronous sockets.

If you have any question or suggestion don't hesitate to send a comment!

#Tips & Tricks #Tutorial #InterSystems IRIS #InterSystems IRIS for Health

Source URL:https://community.intersystems.com/post/asynchronous-socket-iris-and-connection-javascript-client