Søren Hansen

September 9, 2018 7 minute read

Gateway OData Service – troubleshooting and detailed tracing in ABAP

Hi Experts,

I have recently come across a situation, where bad performance of one of my custom developed OData service caused a lot of issues, and where I was forced to think out-of-the-box to come up with ways to perform detailed tracing. And now I want to share these with you guys.

In this blog post I will show some tools for logging and tracing OData calls in the Gateway as well as in the SAP Backend. Furthermore, I will show how OData calls can be captured in the ABAP Runtime Analysis tool (SAT) in the SAP backend for very detailed analysis of all steps being carried out in the backend.

The Runtime Analysis Tool is known by most ABAP developers, but I will show how a capture can be scheduled, so an OData call from a browser can trigger the creation of a measurement for further analysis.

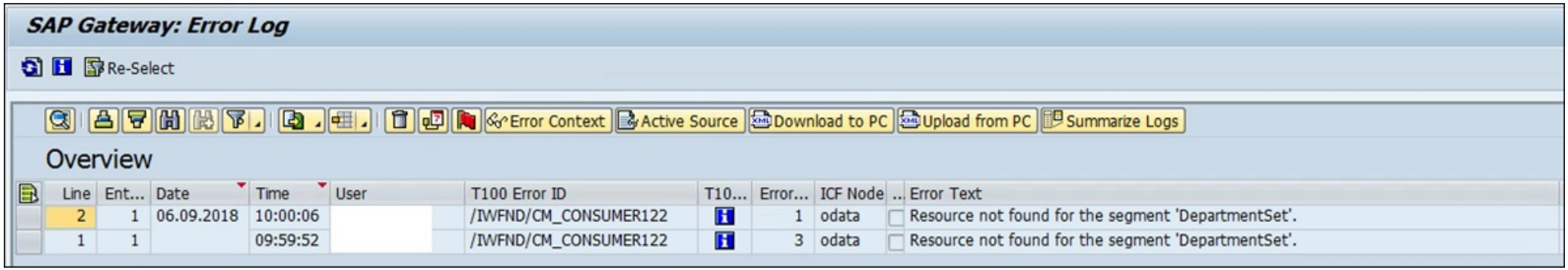
In SAT it is possible to view things like: Which DB tables are being updated, which coding blocks are being called, and how many times, as well as runtime in microseconds for each call, and much more.

SAP Gateway Tracing

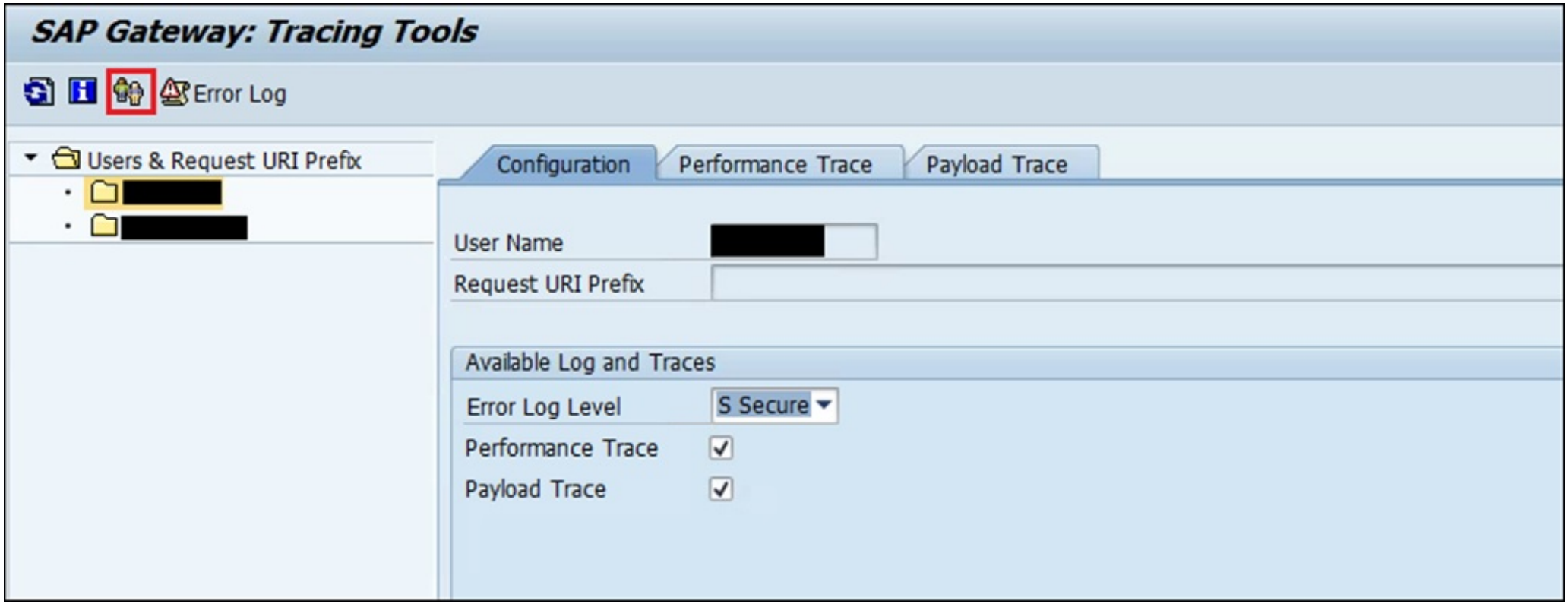
When working with SAP gateway OData services, most people come across the two most commonly used transactions for logging and tracing in SAP Gateway.

- /IWFND/ERROR_LOG: Displays erroneous OData calls to the Gateway
- /IWFND/TRACES: Display Performance and Payload traces

In environments where the SAP Gateway and the SAP Backend system are in two distinct systems (using a Gateway Hub), these utilities are used on the SAP Gateway Hub (not the backend).



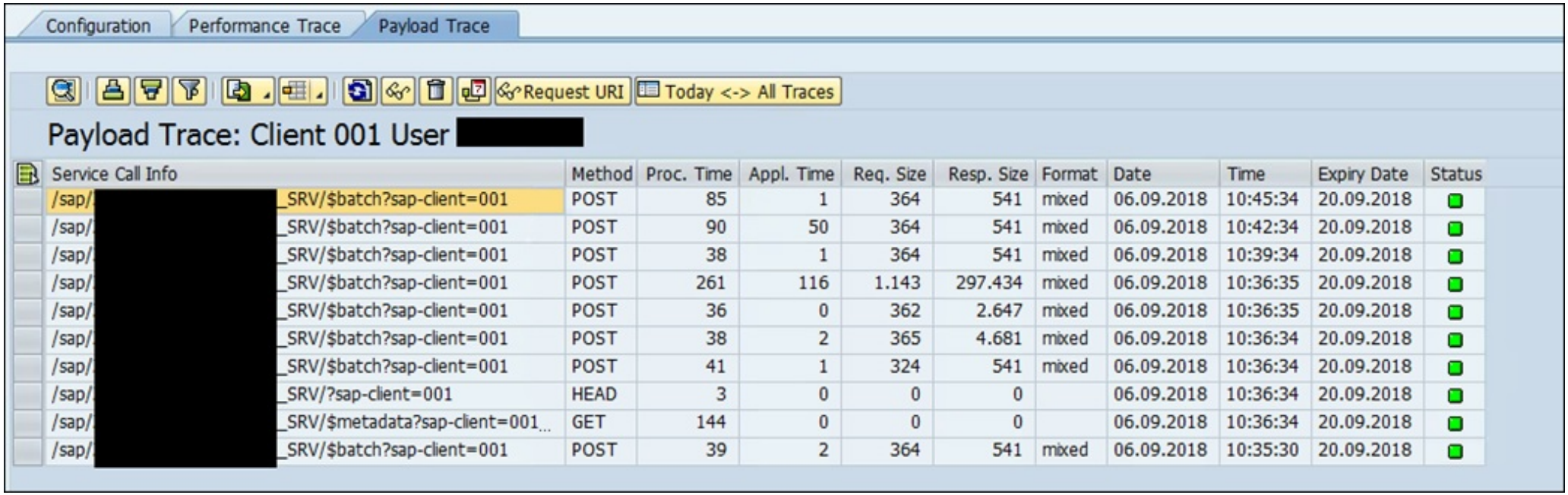
The Error Log is a continuous log, that will display framework errors.



Traces, which are user specific, must be enabled before they can be displayed.

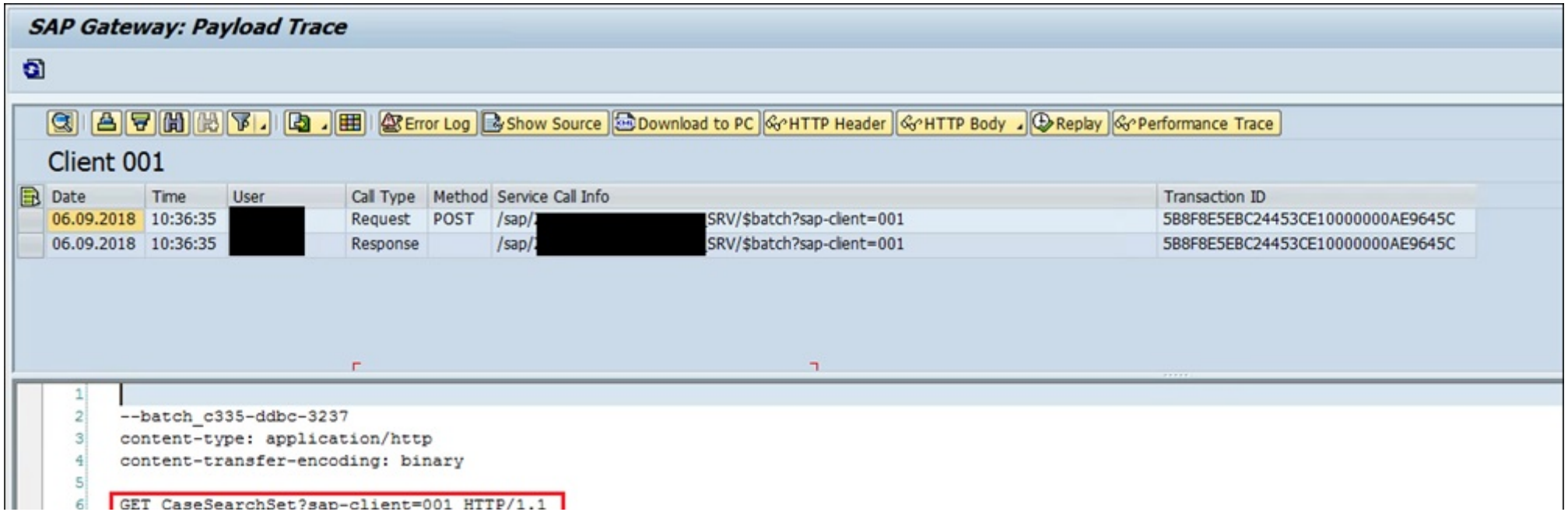
The login user will be selected by default, but other users may be added by using the marked button.

Traces are active for a period of two hours, after which they will automatically be inactivated.



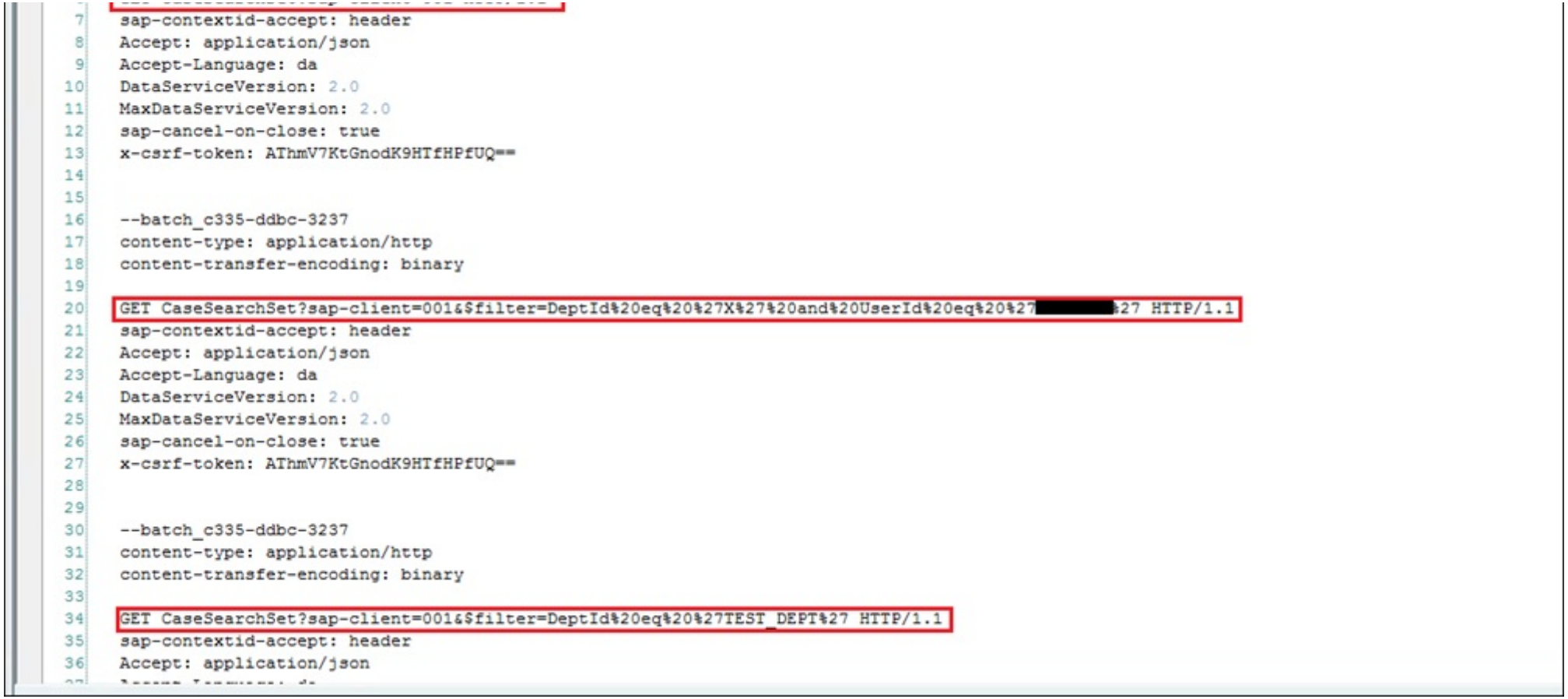
Service Call Info	Method	Proc. Time	Appl. Time	Req. Size	Resp. Size	Format	Date	Time	Expiry Date	Status
/sap/[redacted]_SRV/$batch?sap-client=001	POST	85	1	364	541	mixed	06.09.2018	10:45:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	90	50	364	541	mixed	06.09.2018	10:42:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	38	1	364	541	mixed	06.09.2018	10:39:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	261	116	1.143	297.434	mixed	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	36	0	362	2.647	mixed	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	38	2	365	4.681	mixed	06.09.2018	10:36:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	41	1	324	541	mixed	06.09.2018	10:36:34	20.09.2018	
/sap/[redacted]_SRV/?sap-client=001	HEAD	3	0	0	0		06.09.2018	10:36:34	20.09.2018	
/sap/[redacted]_SRV/$metadata?sap-client=001...	GET	144	0	0	0		06.09.2018	10:36:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	39	2	364	541	mixed	06.09.2018	10:35:30	20.09.2018	

Payload Trace is an excellent tool for displaying the raw OData requests (and responses) being performed on the system.



Date	Time	User	Call Type	Method	Service Call Info	Transaction ID
06.09.2018	10:36:35	[redacted]	Request	POST	/sap/[redacted]SRV/$batch?sap-client=001	5B8F8E5EBC24453CE10000000AE9645C
06.09.2018	10:36:35	[redacted]	Response		/sap/[redacted]SRV/$batch?sap-client=001	5B8F8E5EBC24453CE10000000AE9645C

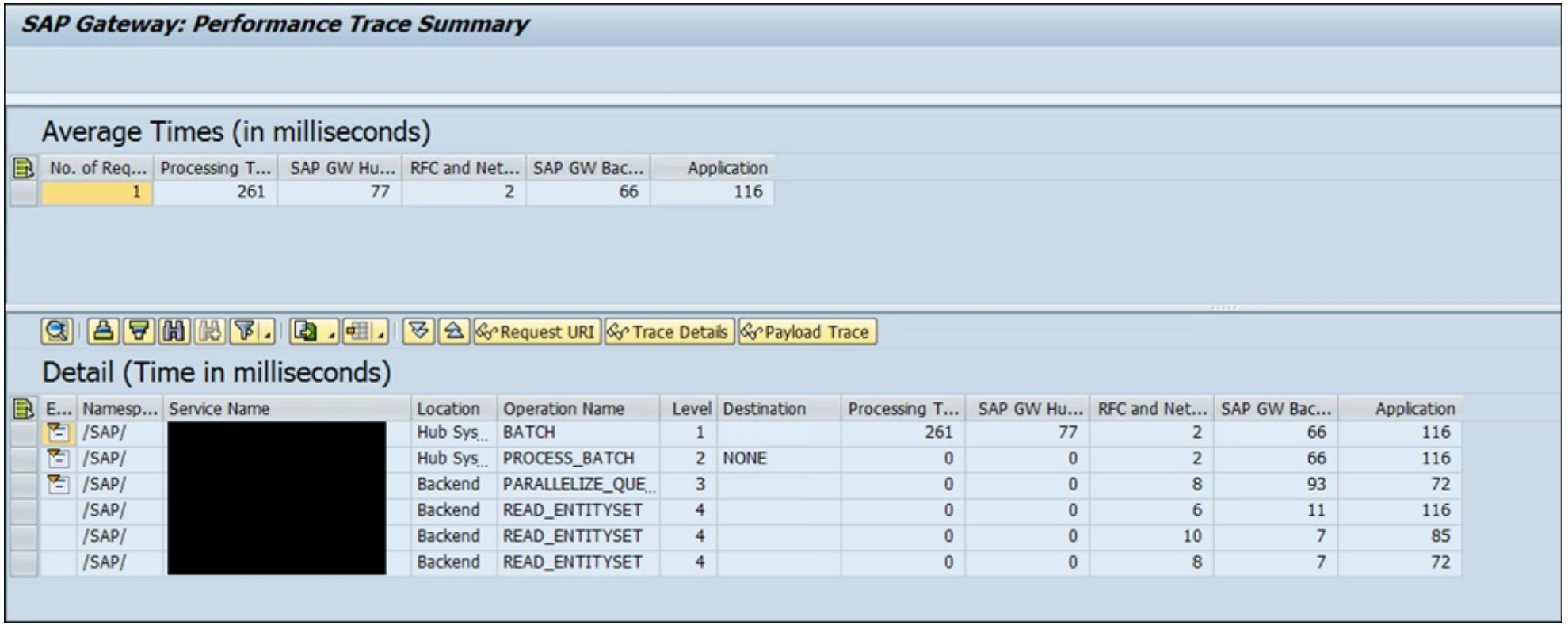
```
1
2  --batch_c335-ddbc-3237
3  content-type: application/http
4  content-transfer-encoding: binary
5
6  GET CaseSearchSet?sap-client=001 HTTP/1.1
```



In case of $batch calls being used one http request may contain multiple OData requests. This makes it harder, in error situations, to pinpoint where the problem is.

In the above call 3 separate GET calls are being carried in one request.

In this view we see the raw data being passed from (request) and to (response) the client.



The corresponding Performance Trace display the 3 GetEntitySet calls, but with no details from the backend. Only the total processing time for each call

SAP Backend

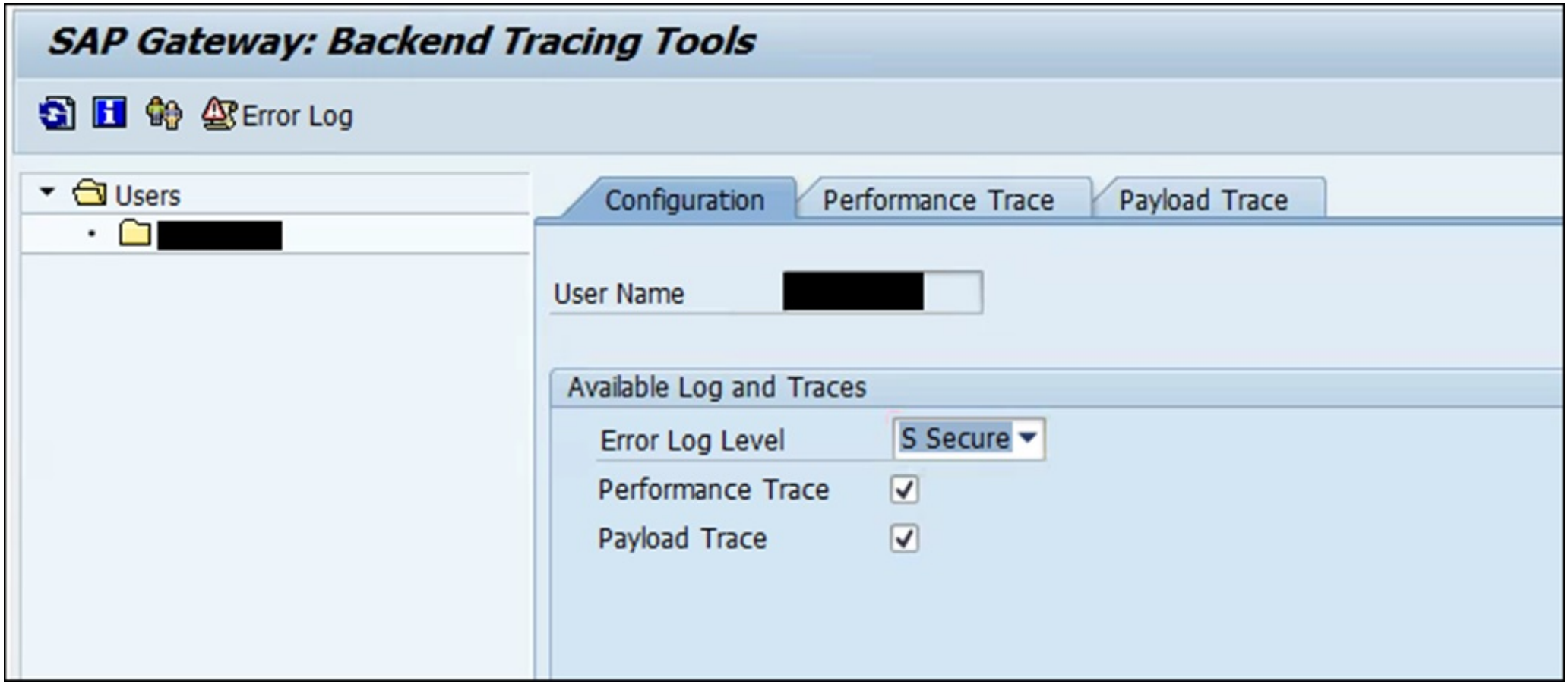
In the backend system there are 2 similar utilities for logging and tracing.

- /IWBEP/ERROR_LOG: Displays erroneous OData calls to the Gateway
- /IWBEP/TRACES: Display Performance and Payload traces

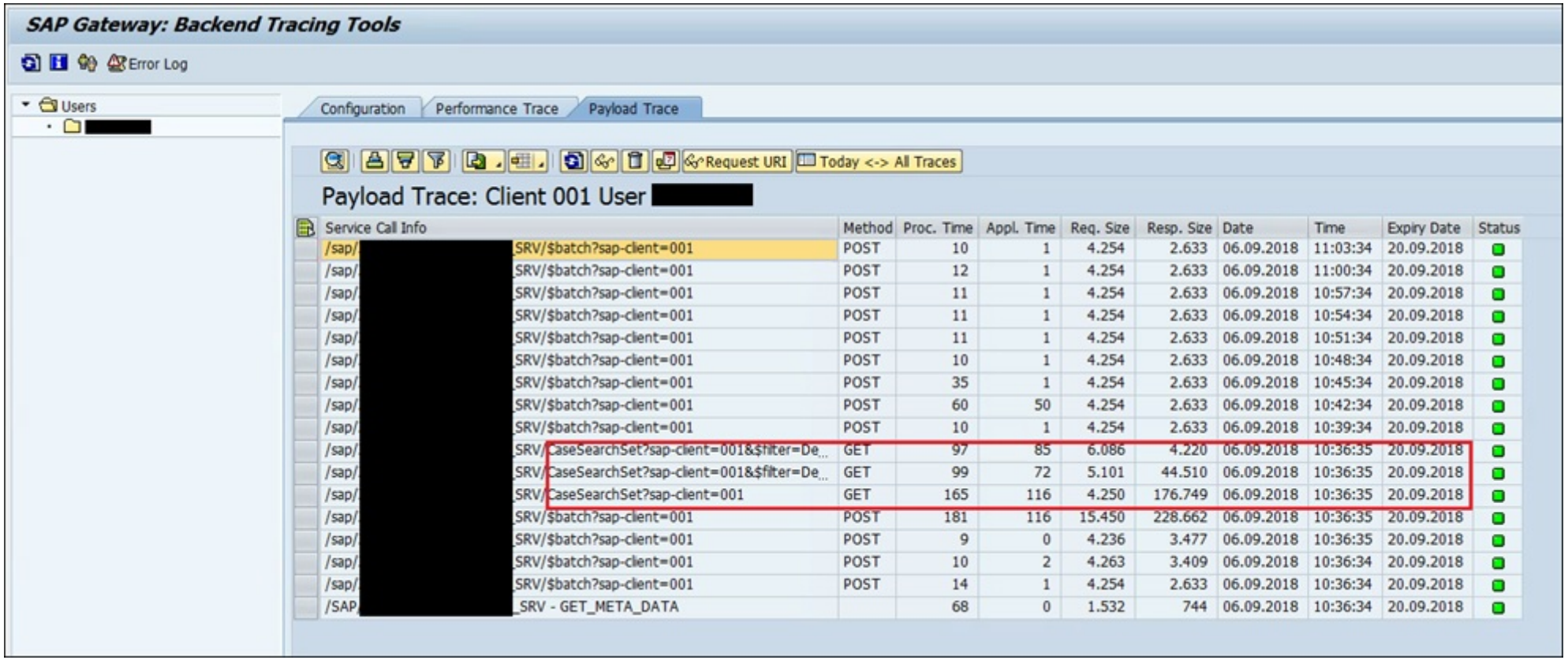
These will show similar info from the SAP Gateway framework, but in the backend system.

Note:
I have found that, on older SAP versions, these may not start properly from the startscreen, but for some weird reasons they can be started from SE93 or by adding them as favorites.

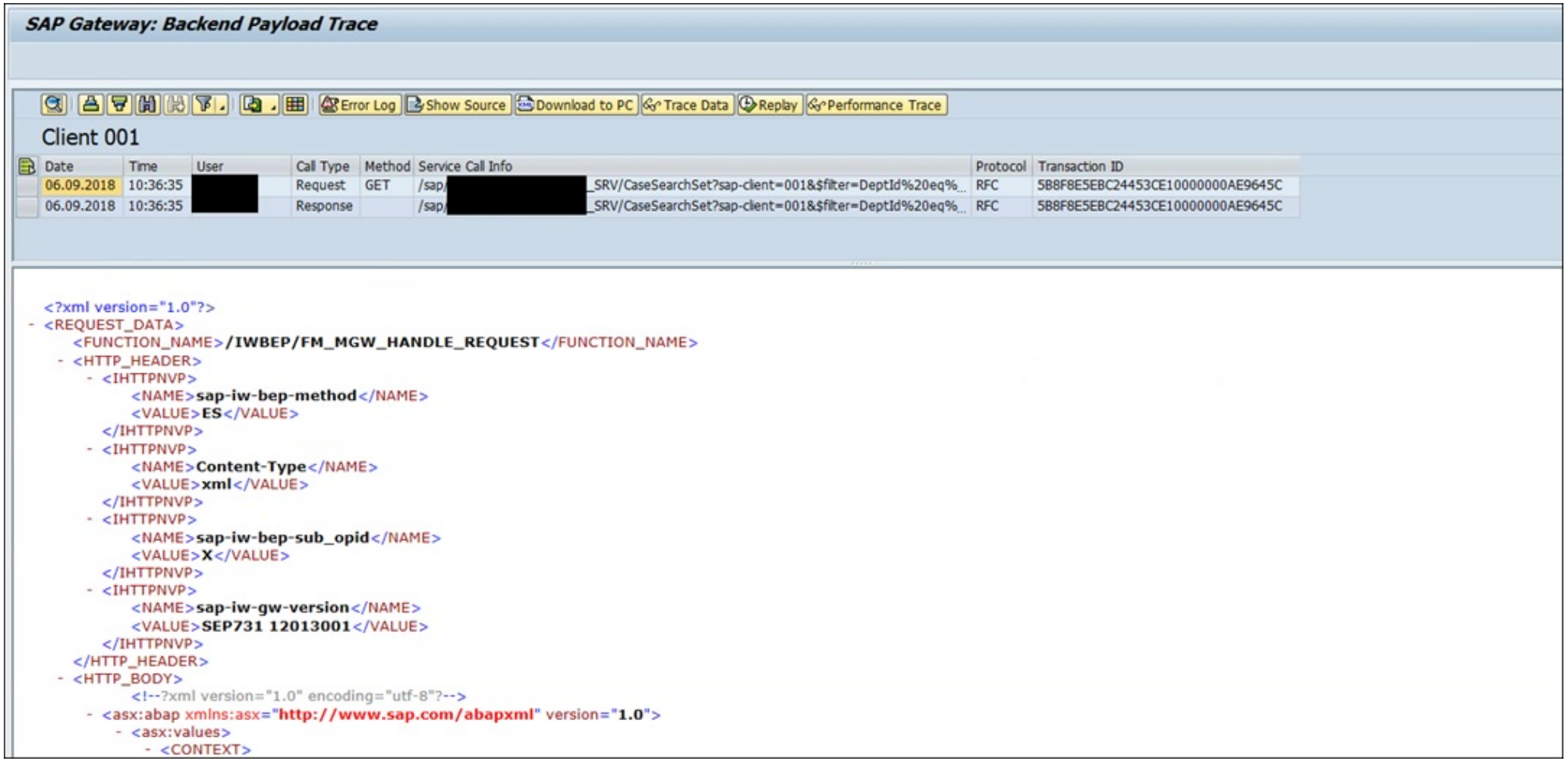
It is important to understand the way the Gateway framework works, to understand what we can use these tools for. When the SAP Gateway receives an OData request from a client it validates and de-serialize the JSON request data into ABAP data. Then it identifies which Backend system the call should be routed to and calls the backend system via an RFC (function module), passing data in a mapped format. On the backend system, the Data Provider Class is called to execute the request.



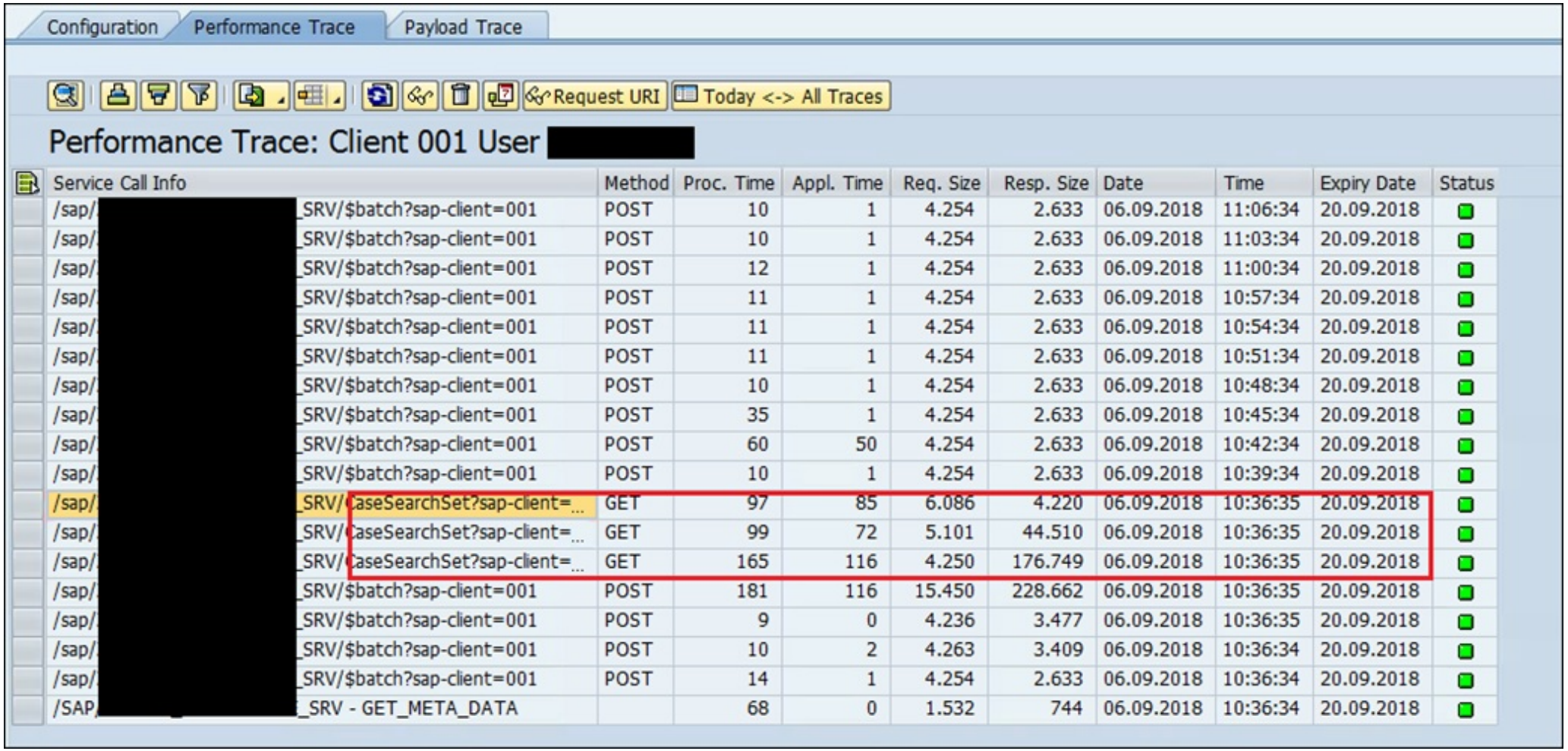
The Backend Traces can be enabled the same way as on the SAP Gateway.



In the backend Payload Trace we will see separate measures for each OData call contained in $batch multi requests. This will give us a little more detailed info than we got on the SAP Gateway. Requests of type $batch with only a single measure in this list contains only one OData request.

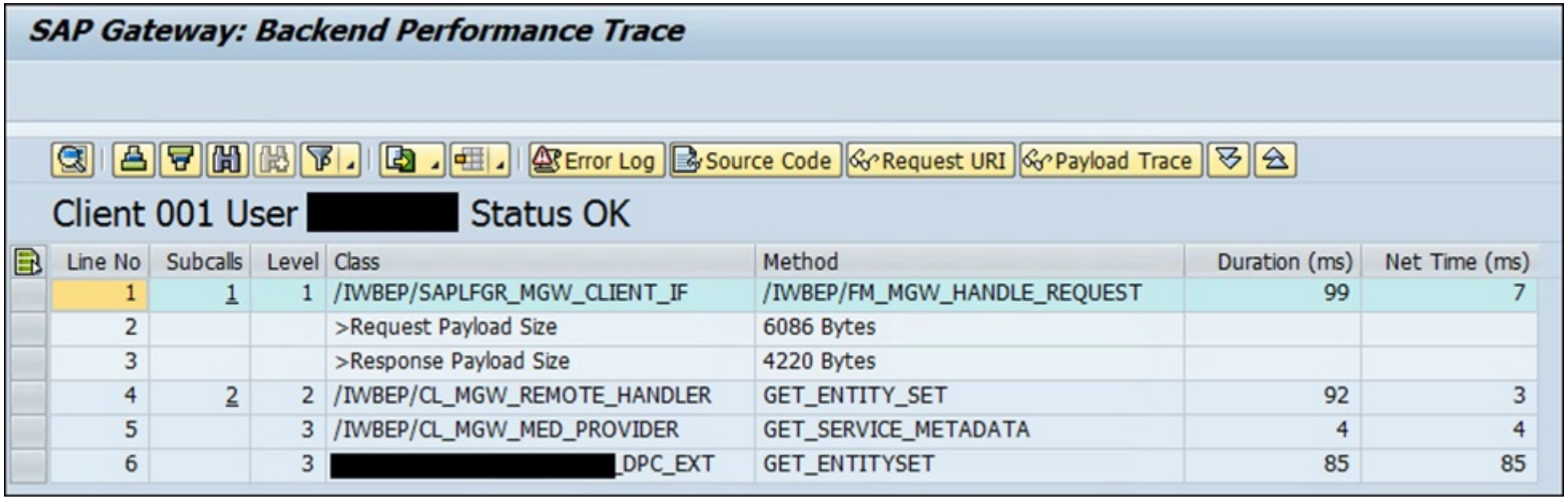


When the payload is examined, we will now see request payloads in XML format. This is the data format being passed to the RFC Function Module from the SAP Gateway, after the json request has been deserialized.



Service Call Info	Method	Proc. Time	Appl. Time	Req. Size	Resp. Size	Date	Time	Expiry Date	Status
/sap/[redacted]_SRV/$batch?sap-client=001	POST	10	1	4.254	2.633	06.09.2018	11:06:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	10	1	4.254	2.633	06.09.2018	11:03:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	12	1	4.254	2.633	06.09.2018	11:00:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	11	1	4.254	2.633	06.09.2018	10:57:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	11	1	4.254	2.633	06.09.2018	10:54:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	11	1	4.254	2.633	06.09.2018	10:51:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	10	1	4.254	2.633	06.09.2018	10:48:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	35	1	4.254	2.633	06.09.2018	10:45:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	60	50	4.254	2.633	06.09.2018	10:42:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	10	1	4.254	2.633	06.09.2018	10:39:34	20.09.2018	
/sap/[redacted]_SRV/CaseSearchSet?sap-client=...	GET	97	85	6.086	4.220	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/CaseSearchSet?sap-client=...	GET	99	72	5.101	44.510	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/CaseSearchSet?sap-client=...	GET	165	116	4.250	176.749	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	181	116	15.450	228.662	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	9	0	4.236	3.477	06.09.2018	10:36:35	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	10	2	4.263	3.409	06.09.2018	10:36:34	20.09.2018	
/sap/[redacted]_SRV/$batch?sap-client=001	POST	14	1	4.254	2.633	06.09.2018	10:36:34	20.09.2018	
/SAP[redacted]_SRV - GET_META_DATA		68	0	1.532	744	06.09.2018	10:36:34	20.09.2018	

In the Performance Trace, we will also see distinct measures for each OData sub request of a $batch request containing multiple requests.



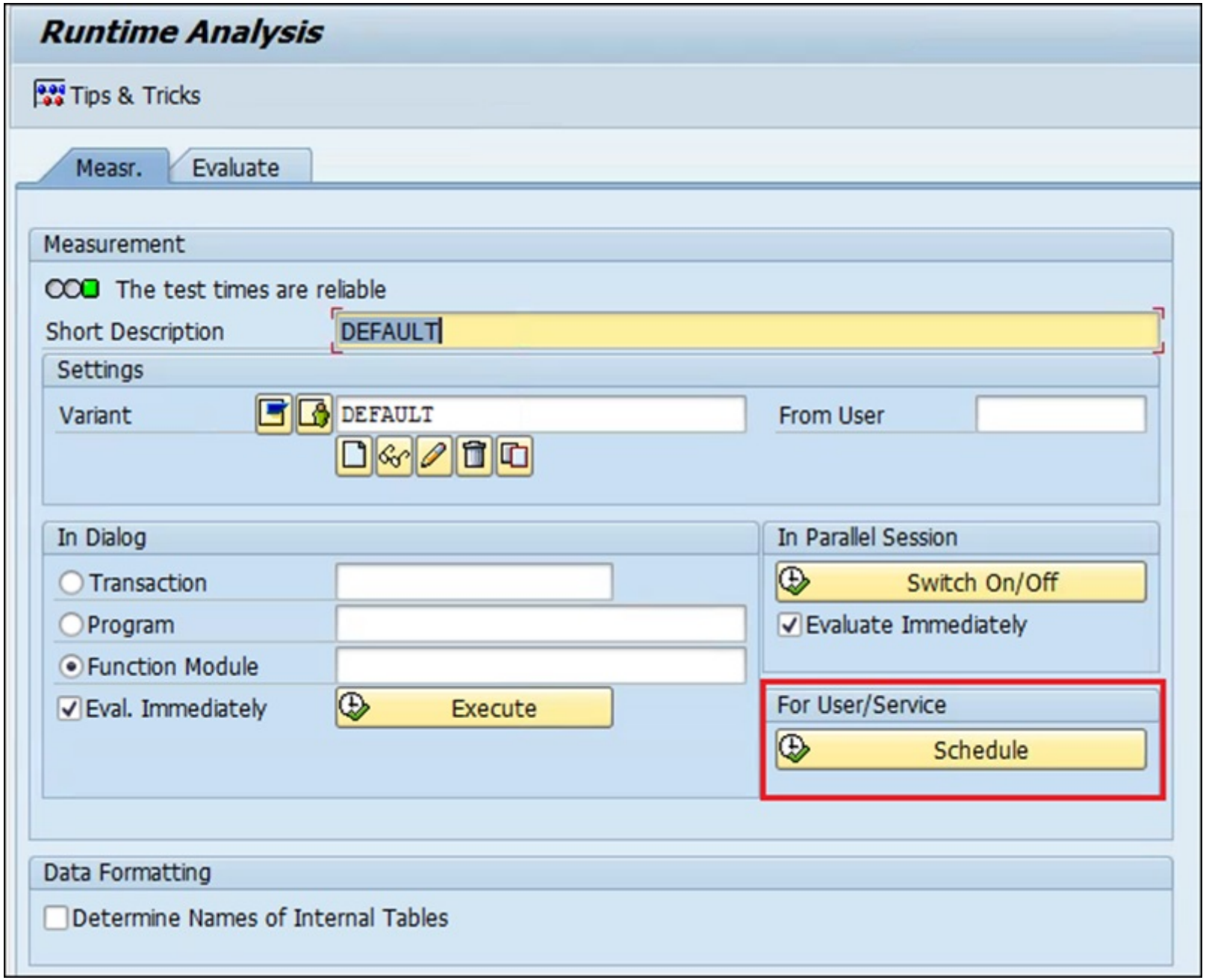
By double clicking an entry we can examine the path to the Data Provider Class executing the request call.

But still we cannot get detailed analysis data on what is happening in the Data Provider Class.

ABAP Runtime Analysis (SAT)

To get very detailed runtime analysis data, of what actions are being performed in the Data Provider Class we can use the ABAP Runtime Analysis tool.

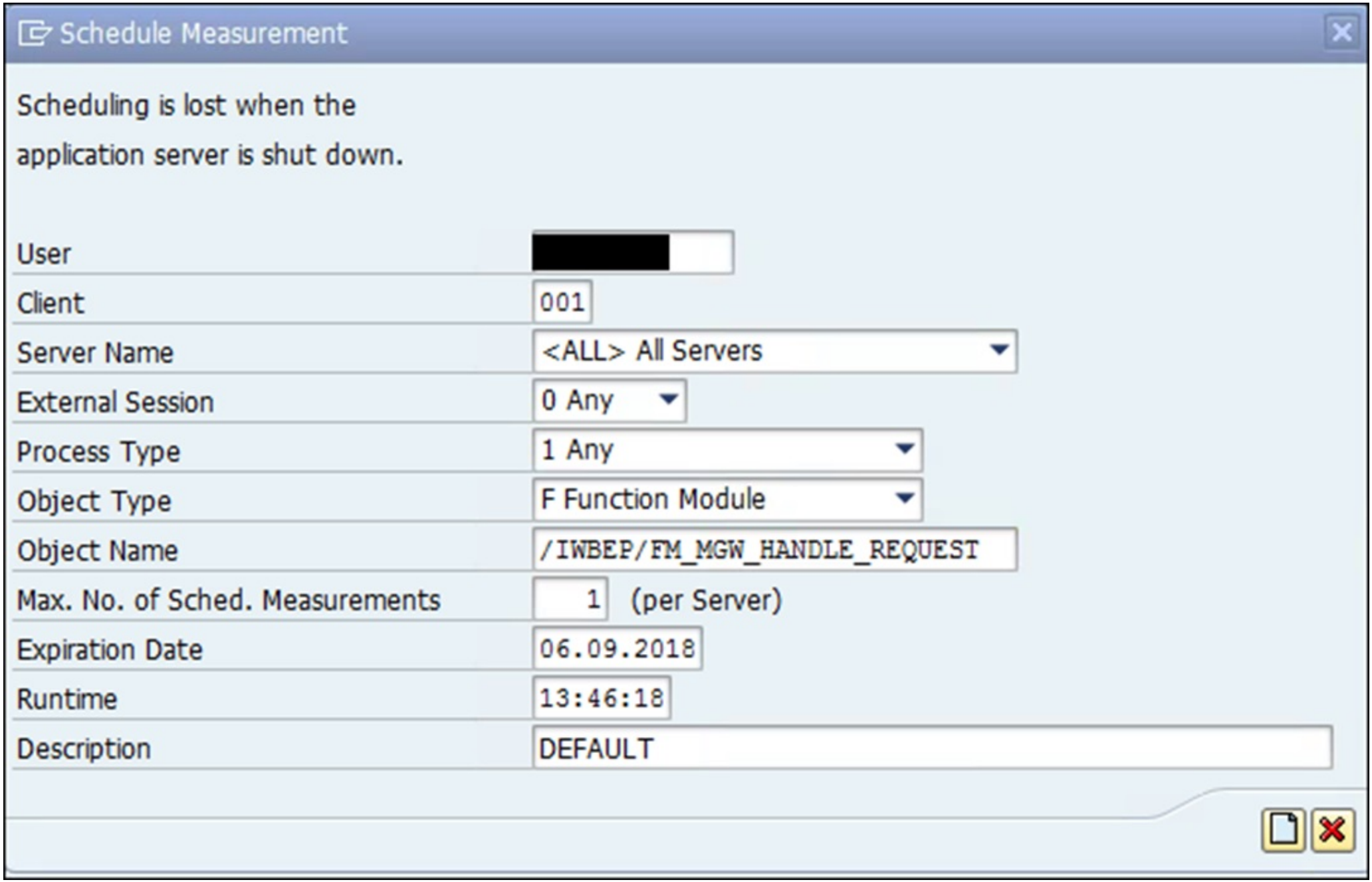
This is done in transaction SAT in the backend system.


Runtime Analysis
Tips & Tricks
Measr.
Evaluate
Measurement
The test times are reliable
Short Description
DEFAULT
Settings
Variant
DEFAULT
From User
In Dialog
Transaction
Program
Function Module
Eval. Immediately
Execute
In Parallel Session
Switch On/Off
Evaluate Immediately
For User/Service
Schedule
Data Formatting
Determine Names of Internal Tables

In SAT you can analyze code being executed from a Transaction, Program or a Function Module. In Gateway context we don't really have any of these directly. But since SAP Gateway calls are always being passed to the backend system via an RFC (Function Module) call, even when the Gateway is located on the backend system, we can use this to enable the Runtime Analysis.

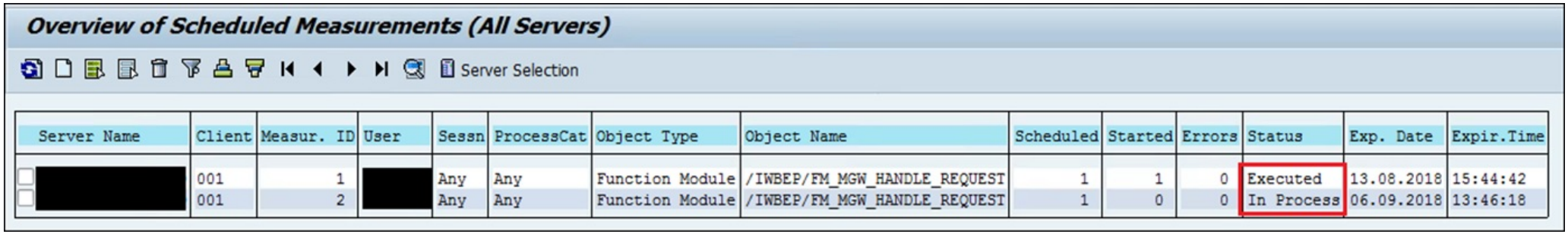
For this we use the scheduling utility (marked in red).

Click the create icon.


Schedule Measurement
Scheduling is lost when the
application server is shut down.
User
Client
001
Server Name
<ALL> All Servers
External Session
0 Any
Process Type
1 Any
Object Type
F Function Module
Object Name
/IWBEP/FM_MGW_HANDLE_REQUEST
Max. No. of Sched. Measurements
1
(per Server)
Expiration Date
06.09.2018
Runtime
13:46:18
Description
DEFAULT

Enter the Function Module '/IWBEP/FM_MGW_HANDLE_REQUEST' and other values as shown above or change to suit your needs. (See below how to find this Function Module).

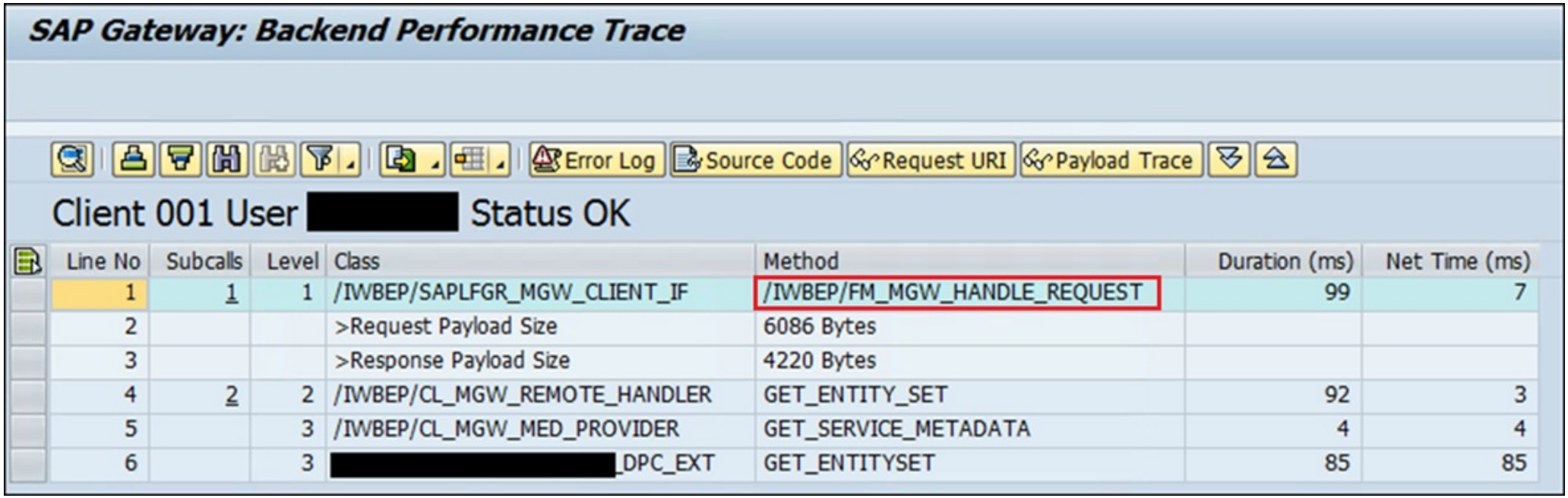
Press Enter.



Server Name	Client	Measur. ID	User	Sessn	ProcessCat	Object Type	Object Name	Scheduled	Started	Errors	Status	Exp. Date	Expir.Time
[redacted]	001	1	[redacted]	Any	Any	Function Module	/IWBEP/FM_MGW_HANDLE_REQUEST	1	1	0	Executed	13.08.2018	15:44:42
	001	2		Any	Any	Function Module	/IWBEP/FM_MGW_HANDLE_REQUEST	1	0	0	In Process	06.09.2018	13:46:18

From the overview, we now see an old measurement already executed, and our new one in status 'In Process', which means it is ready for capture.

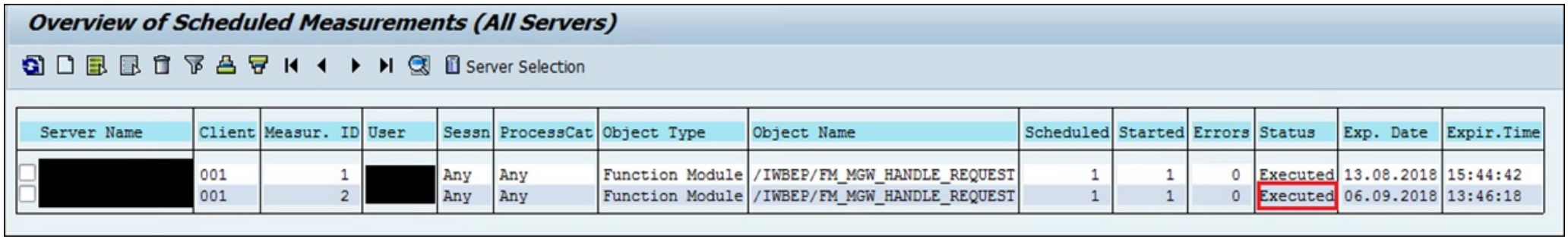
How to find the Function Module:



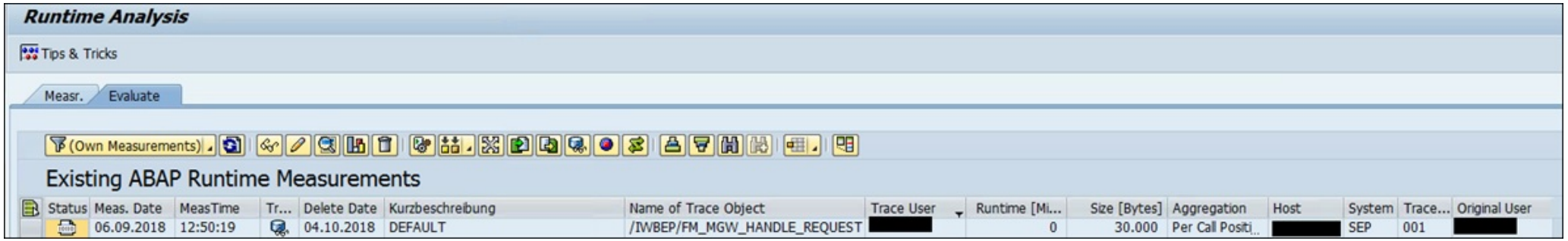
Previously we saw the call stack of the Backend Performance Trace. As explained earlier, the call stack on the backend always starts with the RFC Function Module, so from here we can find it in the Method column.

Start capture

Go to your browser or other client and make the OData call.

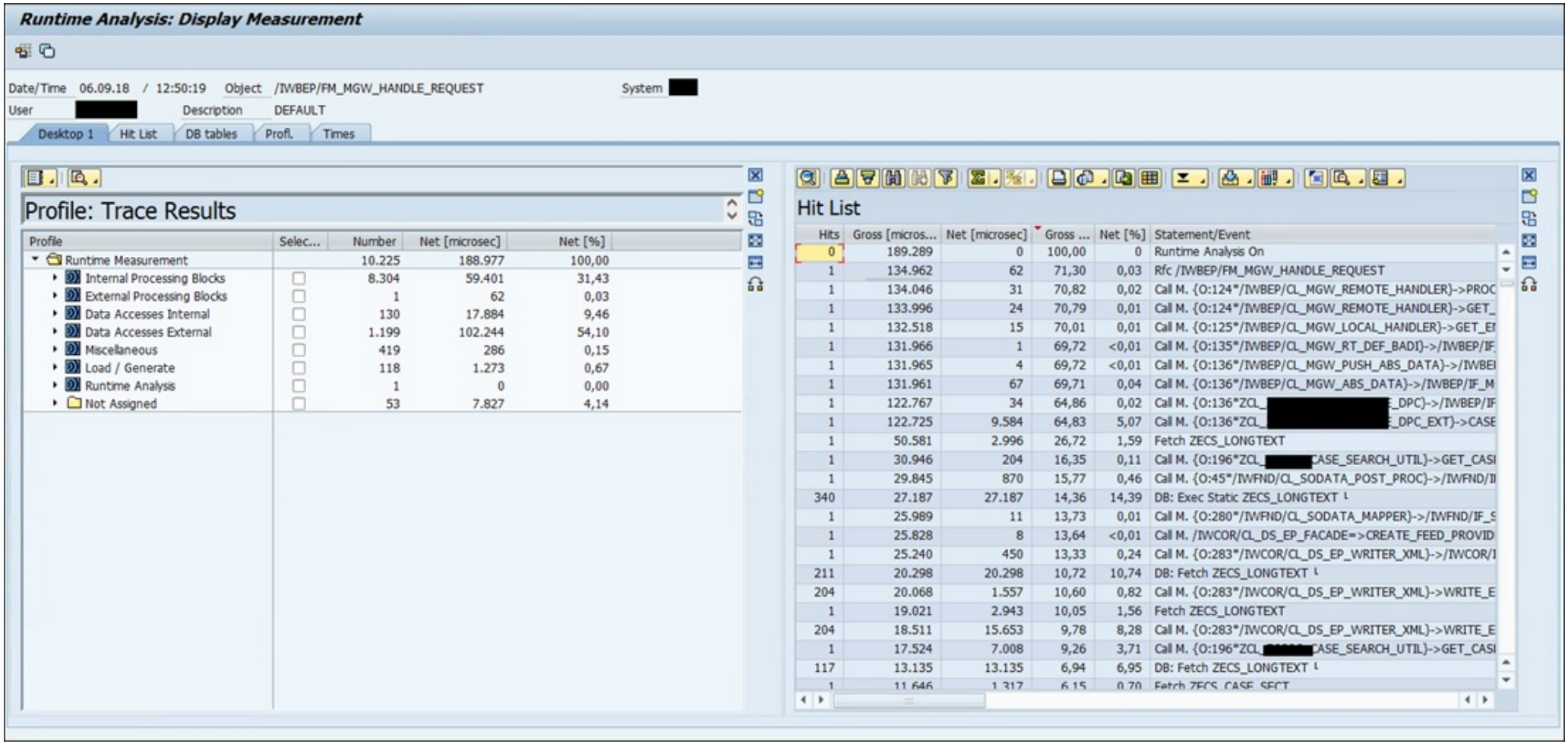


The status is now changed to ‘Executed’



Now go to the Evaluate tab of SAT, to find your measurement.

Double-click it.

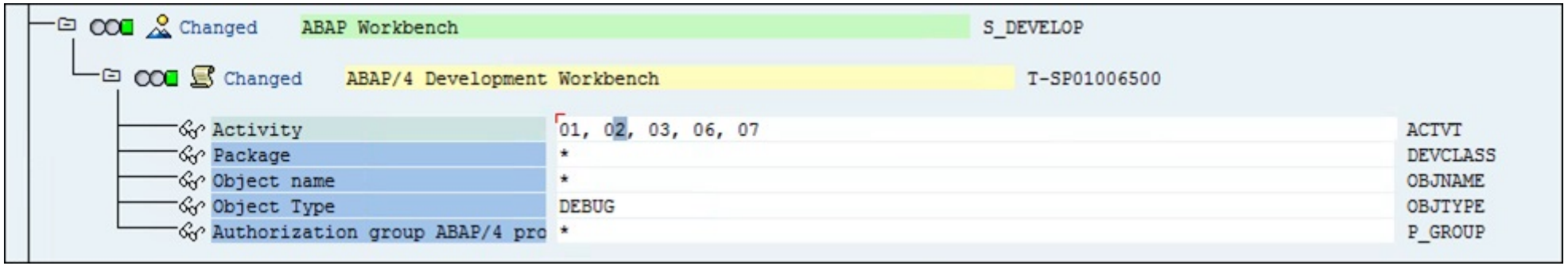


Now you can make a deep dive into the measurement, and see how many times each coding block has used, as well as examining which coding blocks have been called, and much more.

This tool is a very advanced tool, providing very detailed info about the runtime details in the backend system.

Debugging an Odata call

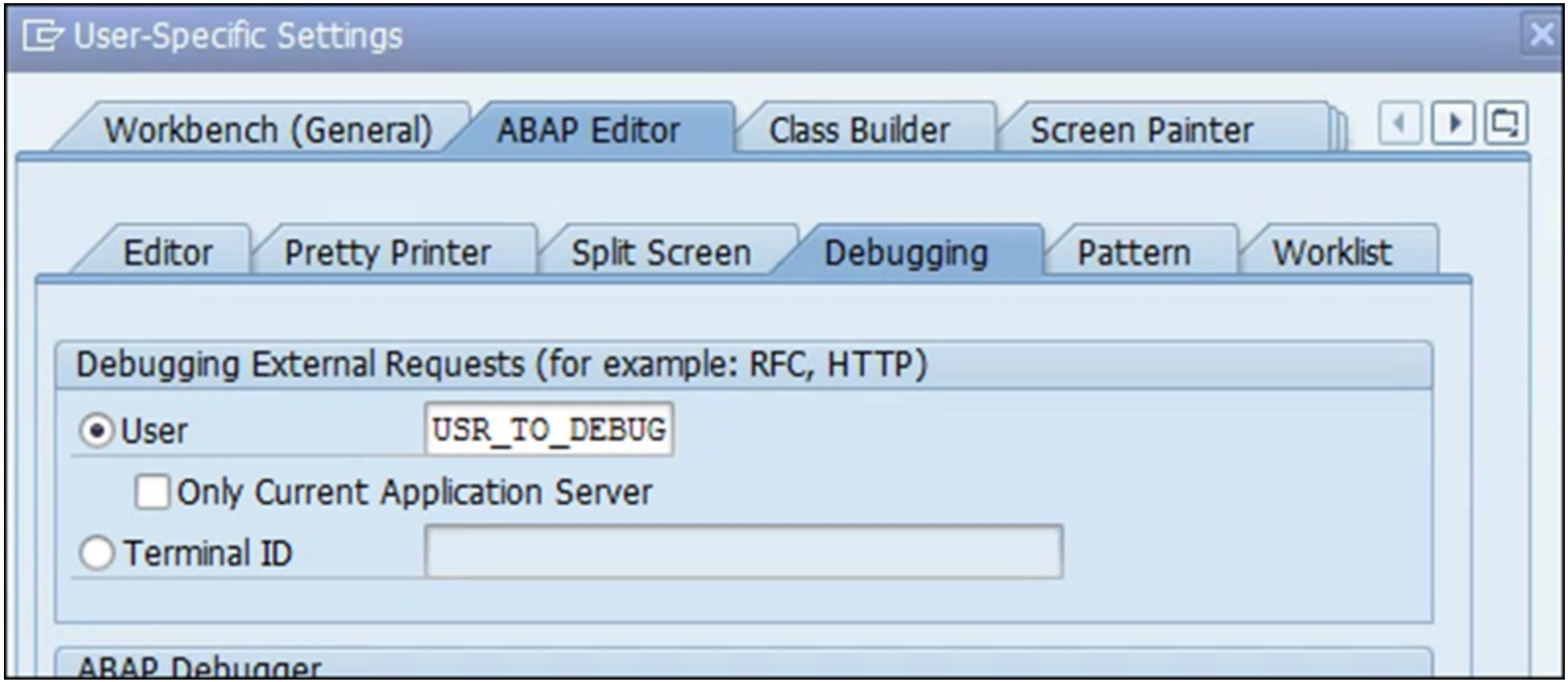
For completeness I will also show how you can debug an Odata call.



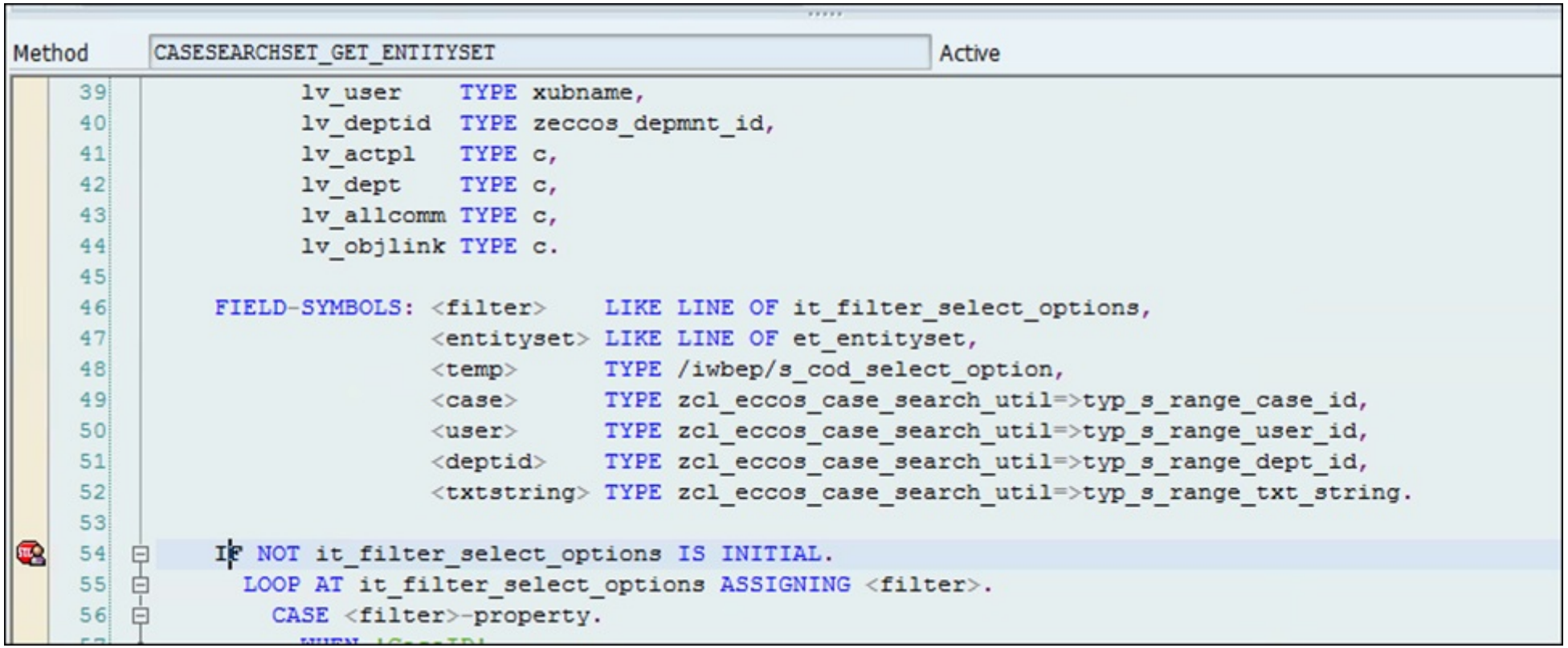
To perform external debugging in ABAP both the debug user (performing the debug) AND the debugee (user being debugged) must BOTH have debug authorizations, as displayed above.

It is not necessary to have authorization for all above activities, but I am not sure which are the minimum needed activities.

Go to transaction SE80, and choose Utilities -> Settings.

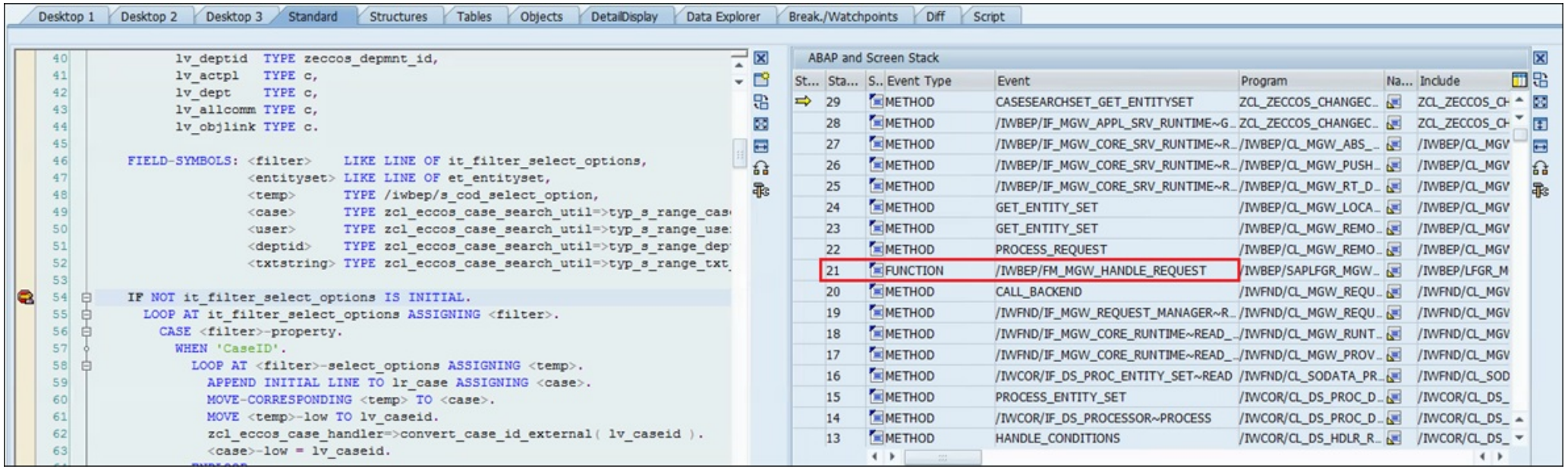


Set the user you want to debug and press Enter.



Find the Data Provider Class method you want to debug, and position the cursor on the line where the breakpoint should be placed.

Click the icon for External Break Point to set it.



When the user executes the call, the debugger will open once the Break Point is hit by the user.

In the above screen dump, it is shown where the RFC Function Module is located in the call stack. It can also be seen, that this system is a hybrid system, where SAP Gateway and Backend is in the same system, since the call stack includes all the Gateway calls as well.

Alert Moderator

I hope some of these insights are useful for some of you guys.

Assigned tags

NW ABAP Gateway (OData) | ABAP Testing and Analysis | OData | SAP Gateway | OData Debugging |

View more...

Søren Hansen, Senior Mobile Consultant, 2BM A/S, Denmark

Related Blog Posts

SAP Gateway Client – Tips and Tricks

By **Akhil Agarwal** , Jun 15, 2017

Register XSOdata servies into SAP Gateway

By **Srilaxmi divi** , Sep 05, 2018

How to do Odata Services from BEx Query

By **Adria Triquell Cristofol** , Feb 19, 2019

Related Questions

SAP ABAP Gateway configuration for ODATA service setup

By **Former Member** , Dec 06, 2016

Unable to create properties without ABAP fields in SAP Gateway 4.0

By **Sreenivas Pachva** , Aug 21, 2017

Redefine External OData Service with OData V4

By **Francesco Alborghetti** , Feb 21, 2018

4 Comments

You must be Logged on to comment or reply to a post.

Syambabu Allu

September 11, 2018 at 4:14 am

Nice information..Thanks for Sharing

Like (0)

Mike Doyle

September 13, 2018 at 3:46 am

Great work, this is a really well presented post. I will keep it for future reference. Thank you

Like (0)

Andre Fischer

September 27, 2018 at 10:30 pm

Nice and very informative post.

Like (0)

Jorge Cabanas

October 25, 2018 at 10:45 am

Good job, it is a very useful information

Like (0)

Share & Follow