

Deep-dive into Sage X3 Engine Traces

Richard Perrins — 5th October 2022

Sage



Contents

Introducing Sage X3 Engine Traces

Mode Notes

Enabling/Disabling Tracing

To 256

... and beyond

Timing Logs and Engine Traces

What Now?

Recap and conclusions

Appendix

Introducing Sage X3 Engine Traces

What are Engine Trace Files ?

Engine Trace files are text-files which record low-level information about the execution of processes associated with Sage X3 – as we shall see, Trace Files can be generated for both adonix and sadoss/sadora processes, depending on the Modes specified at the time the Trace is enabled.

These Engine Trace Files are under the control of the runtime process and are in addition to the normal Trace Files which are generated by the Sage X3 Functions.

The information output from the normal Trace Files is controlled by the Applications Code, and it normally provides information relevant to the task – for example, it could be a report of what Postings have been done and/or any errors encountered.

The Engine Trace Files are independent of the applications-code – the content is generated by the systems-code – so the feature is implicitly available in all Classic Functions.

Why are we interested in Engine Traces?

Engine Traces provide a mechanism for gaining insights into the inner workings of Sage X3 – at both the Applications and the Systems Levels.

In addition, as we'll see later, we can also use them to record the time taken to execute particular code-blocks within the Applications.

Both of the above can provide useful information when investigating issues in Sage X3.

The most value can be gained when using the Engine Trace files in conjunction with the Code – as such, the interpretation of the Engine Trace files is normally carried-out by Sage's Customer Services Team as they have access to the Code.

What can Engine Traces tell us?

Before enabling Engine Tracing, we should consider a Check-list

- 1) Confirm the Patch Level – this will enable us to look at the correct version of the Source Code
- 2) Provide relevant Trace-files – ensure that Tracing is enabled for the minimum time encapsulating the issue – if investigating a performance issue, then enable the Tracing just before executing the Action (such as Posting) and disable the Tracing as soon as control has been returned to the screen (able to execute another Calculator Action). If the Menu Item errors or exits, then the Trace will stop when control is returned to the Menu.
- 3) Use a relevant Trace Flag Level
- 4) Might SQL Traces (SQL Profiler/Extended Events) help? There can be instances where examining the SQL Statements in conjunction with Trace Files can help to understand what a program is doing – in particular, when your issue involves a function not returning expected results.

What can Engine Traces tell us?

Engine Tracing can provide varying levels of information, depending on what “Mode” you specify.

Here is the official list of Modes you can use:

Value	Element logged
1	Only the execution of the Gosub , Call , Callmet , and Fmet engine instructions.
2	Execution of all engine instructions.
4	Execution of the Read or For statements when they are used to access the database.
8	Technical data feed exchanged between the SAFE X3 engine and the database driver (sadora or sadoss query).
16	Technical JSON data feeds between the client and the web server.
32	Technical binary data feeds between the web server and the Sage X3 server for Classic pages.
64	Technical exchanges between the LDAP server and the Sage X3 server.
128	Only the execution of the Opldap instruction (Technical).
256	Technical exchanges linked to the runtime starting phase. (Technical)

Each mode has a “channel” prefix on each line of the Engine Trace file to distinguish which “Mode” it relates to.

Mode Notes

What do you get from the different Modes?

Mode 1: Only the execution of the Gosub, Call, Callmet, and Fmet engine instructions.

With Mode 1, you get <channel 1> and <channel 3> which equate to code-block calls and the returns:

<channel 3>@X3.TRT/ATRIFIL\$adx(79)	Gosub INIT_UUID , tick:32452
<channel 1>@X3.TRT/ATRIFIL\$adx(79)	End Gosub, tick:32452
<channel 1>@X3.TRT/GOBJACT\$adx(1183)	End Gosub, tick:32452
<channel 1>@X3.TRT/GOBJACT\$adx(1134)	End Gosub, tick:32454
<channel 1>@X3.TRT/GOBJSUB\$adx(2505)	End Gosub, tick:32454
<channel 1>@X3.TRT/GOBJSUB\$adx(2453)	End Gosub, tick:32457
<channel 3>@X3.TRT/GOBJSUB\$adx(2459)	Gosub ACTION , tick:32457
<channel 3>@X3.TRT/GOBJSUB\$adx(2642)	Gosub ACTION From SUBSOH, tick:32457
<channel 3>@X3.TRT/SUBSOH\$adx(26)	Gosub ENTREE From EXEFNC, tick:32457
<channel 1>@X3.TRT/SUBSOH\$adx(26)	End Gosub, tick:32457
<channel 3>@X3.TRT/SUBSOH\$adx(62)	Gosub LIENS From SUBSOHA, tick:32457
<channel 3>@X3.TRT/SUBSOHA\$adx(2104)	Gosub LIENS_SOH , tick:32457
<channel 3>@X3.TRT/SUBSOHA\$adx(2213)	Call GLOBVAR From TRTX3,

What do you get from the different Modes?

Mode 2: Execution of all engine instructions.

With Mode 2, you get <channel 2> and <channel 3> which equate to all 4GL instructions including code-block calls and the returns respectively:

<channel 1 2>@X3.TRT/ATRIFIL\$adx(74)												IT
<channel 1 2>@X3.TRT/ATRIFIL\$adx(74)												Assign
<channel 1 2>@X3.TRT/ATRIFIL\$adx(75)												If
<channel 1 3>@X3.TRT/ATRIFIL\$adx(79)												Gosub INIT_UUID , tick:892379
<channel 1 2>@X3.TRT/ATRIFIL\$adx(199)												If
<channel 1 2>@X3.TRT/ATRIFIL\$adx(200)												If
<channel 1 2>@X3.TRT/ATRIFIL\$adx(201)												If
<channel 1 2>@X3.TRT/ATRIFIL\$adx(208)												Return
<channel 1 2>@X3.TRT/ATRIFIL\$adx(90)												Return
<channel 1 2>@X3.TRT/GOBJACT\$adx(1183)												Rewrite
<channel 1 2>@X3.TRT/GOBJACT\$adx(1200)												Return
<channel 1 2>@X3.TRT/GOBJACT\$adx(1135)												Return
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2506)												Commit
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2507)												Return
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2455)												If
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2456)												If
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2459)												Assign
<channel 1 3>@X3.TRT/GOBJSUB\$adx(2459)												Gosub ACTION , tick:892382
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2543)												If
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2552)												If
<channel 1 2>@X3.TRT/GOBJSUB\$adx(2557)												If

What do you get from the different Modes?

Mode 4: Execution of the Read or For statements when they are used to access the database.

With Mode 4, you only get the Database Statements such as Read, Write, Delete and Re-write – this is recorded by the adonix process:

```
<channel 4>Query end, tick : 1027693
<channel 4>Execution SQL on Read clause in @X3.TRT/CONTOBJ$adx at line 861, tick : 1027694
<channel 4>Select BPC_.ROWID, BPC_.* From SEED.BPCUSTOMER BPC_ Where ( BPC_.BPCNUM_0 = ? ) Order by BPC_.BPCNUM_0 Option (FAST 1)
<channel 4>Query end, tick : 1027695
<channel 4>Execution SQL on Read clause in @X3.TRT/SUBSOHA$adx at line 2233, tick : 1027699
<channel 4>Select BPR_.ROWID, BPR_.* From SEED.BPARTNER BPR_ Where ( BPR_.BPRNUM_0 = ? ) Order by BPR_.BPRNUM_0 Option (FAST 1)
<channel 4>Query end, tick : 1027701
<channel 4>Execution SQL on Read clause in @X3.TRT/SUBSOHA$adx at line 2250, tick : 1027701
<channel 4>Select BPC_.ROWID, BPC_.* From SEED.BPCUSTOMER BPC_ Where ( BPC_.BPCNUM_0 = ? ) Order by BPC_.BPCNUM_0 Option (FAST 1)
<channel 4>Query end, tick : 1027702
```

This can be useful when correlating with SQL Tracing.

What do you get from the different Modes?

Mode 8: Technical data feed exchanged between the SAFE X3 engine and the sadxxx driver.

With Mode 8, you only get the Database Statements such as Read, Write, Delete and Re-write - the information is provided by the relevant sadoss/sadora process rather than adonix as is the case with Mode 4:

```
<channel 8>Resultat 1er Fetch to Record, 1 lignes : 0 at Fri Aug 19 15:05:20.10 2022

<channel 8>Rowid lu pour la table 10 : 55
<channel 8>fermeture du curseur de Read de SEED.TABVACBPR en close
<channel 8>Requete : 89, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Ouverture de la table SEED.TAXLINK 49
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Ouverture de la table SEED.TABVAC 62
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Ouverture de la table SEED.AGRPCPY 63
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>Requete : 0, Total : 1163
<channel 8>lecture 1er enreg du curseur de Read sur SEED.BPARTNER
<channel 8>Select BPR_.ROWID, BPR_.*
From SEED.BPARTNER BPR_
Where ( BPR_.BPRNUM_0 = ? )
Order by BPR_.BPRNUM_0
Option (FAST 1)
<channel 8>Resultat execution to Record : 0 at Fri Aug 19 15:05:20.36 2022
```

What do you get from the different Modes?

Mode 16: Technical JSON data feeds between the client and the web server.

With Mode 16, information relating to any JSON traffic is recorded – this could be Representations such as ACHGENVX3 (used during logging in), License checking, Queries such as ACH021 which is part of the Buyer Landing Page, Options such as Read-only pages > Financials > Analytical Dimensions (referencing CACCE Representation), Batch Server Requests, and even screen/message information.

This mode should be run with start_log (see later)

```
charset=utf-8", "x-requested-with": "XMLHttpRequest", "sec-ch-ua-platform": "\Windows\"", "sec-fetch-site": "same-origin", "sec-fetch-mode": "cors", "sec-:
ml?url=%2Fsdata%2Fx3%2Ferp%2FX3ERPVL2_SEED%2FCACCE%3Frepresentation%3DCACCE.%2524query%26profile%3D~(loc~%27en-GB~role~%27affb605c-50db-4e3c-a60f-:
g": "gzip, deflate, br", "cookie": "client.id=ceecd311-3ba8-48a5-ac29-3900f85a73c9;
user.profile.8443=~(user~'7430ac2f-a9e1-4ba0-92f8-9f58c82fe5a1~role~'affb605c-50db-4e3c-a60f-f2f17cf3aa49~erp~'2013e3e3-9c1e-4483-b66e-ae5b4f36f8e2~
syracuse.sid.8443=59ea87d5-8b37-4058-9a2d-68b8dd88d105", "method": "GET", "url": "/sdata/x3/erp/X3ERPVL2_SEED/$prototypes('CACCE.$query')?profile=(lo
4f36f8e2~appConn~())&browserTabId=625b9f1cf24f-1662037595940"} ]
<channel 16>Received Data end : [ len : 0, json : ]
```

```
<channel 16>Received Data part : [ len : 3439, json : binary data ]
<channel 16>Received Data end : [ len : 0, json : ]
<channel 16>Send Header : [ len : 82, json : {"status":201,"message":"Created","location":"","content-type":"application/json"} ]
<channel 16>Send Data end : [ len : 0, json : ]
<channel 16>Received Header : [ len : 136, json : {"method":"POST","url":"/sdata/x3/erp/SEED/$service/batch?requestId=1584","accept":"application/json","content-
type":"application/json"} ]
<channel 16>Received Data part : [ len : 73, json : {"maxDelay":0,"ctrlFolder":"X3","ctrlLogin":"admin","ctrlLanIso":"en-US"} ]
<channel 16>Received Data end : [ len : 0, json : ]
```

What do you get from the different Modes?

Mode 32: Technical binary data feeds between the web server and the Sage X3 server for Classic pages.

Mode 32 records information about screen-drawing in great detail – not sure how much assistance this would be?

```
<channel 32>wr_req(ids:1 f:0 l:15) Com[1]->com_vh:1
<channel 32>Send Query to client
<channel 32>/0x00/0x00/0x0f
<channel 32>/0x00/0x00/0x0f/0x00/0x94/0x00/0x00/0x00/0x01/0x00/0x00/0x00/0x00/0x00
<channel 32>clientExecuteRequest
<channel 32>clientDataEntry
<channel 32>wr_req(ids:1 f:0 l:8) Com[1]->com_vh:1
<channel 32>Send Query to client
<channel 32>/0x00/0x00/0x08
<channel 32>/0x00/0x00/0x08/0x00/0xfc/0x01/0x00/0x00
<channel 32>rc_ack:0
<channel 32>clientDataEntry action=ACK_SLST - Selection d'un Ligne Liste
<channel 32>clientDataEntry returns ACK_SLST - Selection d'un Ligne Liste
<channel 32>clientExecuteRequest => other
<channel 32>clientExecuteRequest returns ACK_SLST - Selection d'un Ligne Liste Vmkstat:0
<channel 32>wr_req(ids:1 f:0 l:8) Com[1]->com_vh:1
<channel 32>Send Query to client
<channel 32>/0x00/0x00/0x08
<channel 32>/0x00/0x00/0x08/0x00/0x00/0x00/0x00/0x00
<channel 32>io_mask get (name:ADBBPR, nbdim:1, index:0)
<channel 32>io_mask store (name:ADBBPR, nbdim:1, index:0)
<channel 32>io_mask get (name:ADBFLG, nbdim:1, index:0)
<channel 32>io_mask store (name:ADBFLG, nbdim:1, index:0)
```

What do you get from the different Modes?

Mode 64: Technical exchanges between the LDAP server and the Sage X3 server.

Mode 64 is designed to record LDAP traffic at the Classic level, but it's no-longer used since V6.

What do you get from the different Modes?

Mode 128: Only the execution of the Opadxd instruction (Technical).

Mode 128 records information about opadxd which is supposed to be a binary used by the Java Bridge. However, even if Java Bridge isn't installed, ordinary Classic functions trigger writes to the trace-file...

The output doesn't look very interesting.

```
|<channel 128>Automate event:3 => next state:3  
|<channel 128>Read parameters and execute node:3  
|<channel 128>Automate event:3 => next state:3  
|<channel 128>Read parameters and execute node:3  
|<channel 128>Automate event:3 => next state:3  
|<channel 128>Read parameters and execute node:3  
|<channel 128>Automate event:4 => next state:4  
|<channel 128>Read parameters and execute node:4
```


256...

Mode 256: Technical exchanges linked to the runtime starting phase.

This mode needs to be enabled in `start_log` as it relates to the initialisation of adonix processes.

Each adonix process has its own Trace file in `runtime\tmp` folder created when the adonix is executed and updated when the adonix is terminated.

WARNING : As this is done via `start_log`, even Batch Server and Web Services adonix processes are recorded – we'd advise you to set Batch Server and Web Pools to not auto-start if you're going to use `start_log`.

If `start_log` is set up before a User logs in, the initialisation of their first adonix is also recorded, and it is updated when the User logs out. As mentioned before, unfortunately, the Trace files for Function adonix processes don't record the deletion/destruction of the Function-level processes as well as the creation.

256...

Mode 256

The Trace files for adonix processes relating to Functions have “starting MENU script”

```
{channel 256>Set connection user:x3run password:*****
{channel 256>Creation vproto
{channel 256>Timestamp synchro
{channel 256>Init message file ENG
{channel 256>tokens and variables
{channel 256>ini Vadxenvsys
{channel 256>initAdonixSysVar
{channel 256>Severals inits
{channel 256>APL.ini execution...
{channel 256>APL.ini execution ok
{channel 256>Init counter class APLCOM...
{channel 256>Init counter class APLCOM ok
{channel 256>Debug thead init
{channel 256>Seting adxftl 1
{channel 256>Load OBJECT object ok
{channel 256>Load security path (configRuntime.json)
{channel 256>Check TMPDIR (D:\Sage\X3ERP12\Runtime\Tmp)...
{channel 256>TMPDIR ok
{channel 256>Challenge generation...
{channel 256>Challenge generation ok
{channel 256>Send prolog challenge...
{channel 256>Send prolog challenge ok
{channel 256>Begin syracuse session loop
{channel 256>Read session query
{channel 256>Read session ok with lan en-GB, amet X3RSA, kusr admin, key x3erp12sqlvm
{channel 256>Session type 25
{channel 256>Assignment adxenvsys
{channel 256>Verify signature
{channel 256>Convergence mode: starting MENU script
```

256...

Mode 256

Whereas, the initial login adonix is similar, but records “starting ASYRMAIN script”, and then “ASYRMAIN script finished” followed by five additional lines – see below:

```
<channel 256>Read session ok with lan en-GB, amet X3RSA, kusr admin, key x3erpvl2sqlvm
<channel 256>Session type 33
<channel 256>Assignment adxenvsys
<channel 256>Verify signature
<channel 256>Full syracuse mode: starting ASYRMAIN script
<channel 256>Full syracuse mode: ASYRMAIN script finished
<channel 256>Read session query
<channel 256>Read session ok with lan , amet , kusr , key
<channel 256>Session type -1
<channel 256>Assignment adxenvsys
<channel 768>Delete Session
```

How to enable/disable Engine Tracing

How to enable and disable Engine Traces

Now we know a little about Engine Traces and what's available, how do we actually enable and disable them?

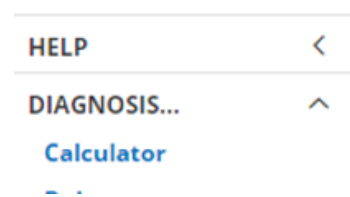
There are five mechanisms:

- 1) Interactive OpenLog()
- 2) Adding Tracing to the 4GL Code
- 3) X3 session log
- 4) “start_log” – the nuclear Option
- 5) Activation Timing (see next section)

How to enable and disable Engine Traces

1) Interactive OpenLog()

In a typical scenario, the User will be starting and stopping the Engine Trace via the built-in Diagnosis > Calculator option



The User will be prompted to type in text and should type in OpenLog() – it will be of the form OpenLog(<volume-name>,<flag-value>).

For example, OpenLog(“TRA”,2) – to start the Trace, and then CloseLog() to stop the trace.

The Trace-file will be created in the <folder>\<volume-name> directory – for example, d:\sage\X3\LIVE\TRA\.

How to enable and disable Engine Traces

2. Adding Tracing to 4GL Code

As well as starting and stopping Tracing interactively, the Developer can add code to manage Trace Files – this allows them to target certain areas of the programs without the need for Users to start/stop the Traces.

This is done using `OpenLog()`, `CloseLog()` and `GetLogname()` calls in the code.

If these are used, there should be some condition in the code for enabling/disabling the calls – perhaps an Activity Code, or execution under specific Conditions (e.g. a particular X3 User or Site) – to record information only when necessary.

Details can be found in the Online Help

<https://online-help.sageerpx3.com/erp/12/staticpost/openlog/>

Also, see Sage City articles cited in the Appendix.

How to enable and disable Engine Traces

3. X3 session log

Administration > Usage > Logs > X3 session log

This mechanism allows you to set up a Trace for a particular User and/or Function as necessary.

Note that this has changed in V12 Patch 27 – see [The Session Log Function got a revamped look](#) Sage City article.

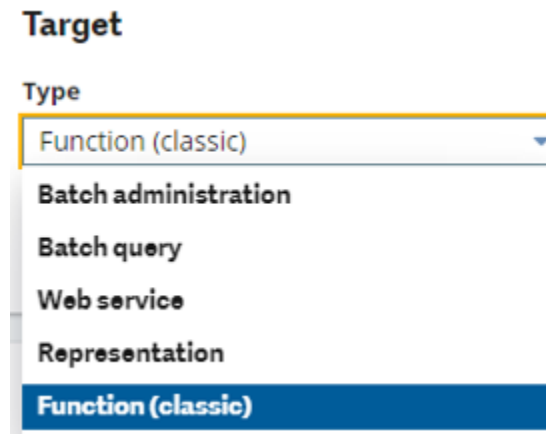
It has the following features:

- 1) As mentioned, you can target a particular User, Endpoint and/or Function
- 2) You can enable/disable the tracing outside the Function in question when required – so, a User can create a Session Log aimed at another User and enable it just before that User runs a process or task.
- 3) The recording starts as soon as the Function is launched, rather than when the Tracing is enabled within Calculator.

How to enable and disable Engine Traces

3. X3 session log

4. This sort of Trace can be associated with different types of task:



5. This Trace will remain active until it is explicitly disabled – please remember to do this as soon as possible



796E329C17a

How to enable and disable Engine Traces

3. X3 session log

6. It applies to all currently-running and future/new sessions – so, if you Enable the logging without specifying any User or Function, it will Trace any Function activity for launched after it's been Enabled. – you can see the list of Functions being actively Traced:

Levels selection

Levels

[LOG_TRT0](#) Function calls with timing(such as 'Gosub' and 'Call') ⋮ [LOG_SQL0](#) Read and For loop SQL Instructions ⋮ [LOG_TRT1](#) All 4GL instructions ⋮

Levels digest

7

Attached X3 sessions

	Type	X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3 locale code	Solution	Folder	Function
⋮	Classic page	7698	6168	Used	x3erpv12sqlvm	50012	admin	en-GB	X3ERP12	SEED	GESPOH
⋮	Classic page	7699	8912	Used	x3erpv12sqlvm	50012	admin	en-GB	X3ERP12	SEED	GESBPC

How to enable and disable Engine Traces

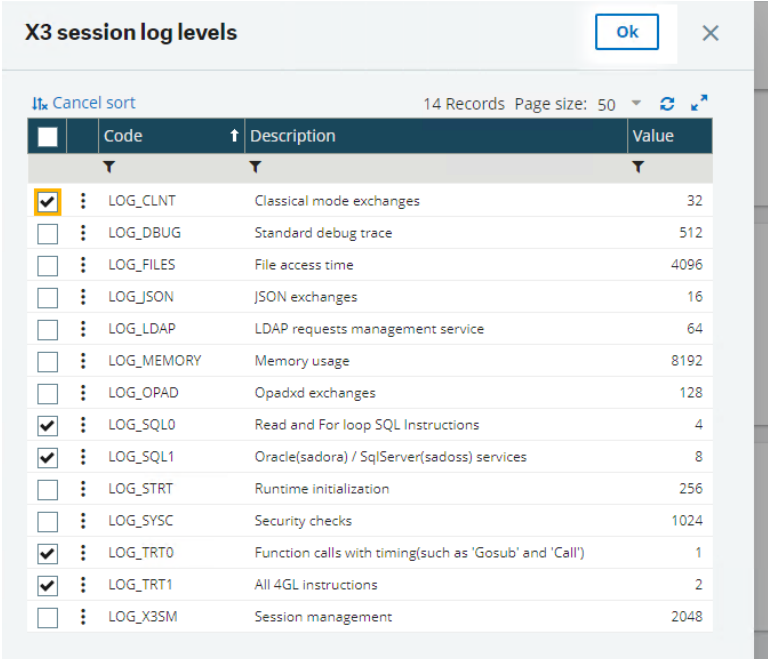
3. X3 session log

5. The Trace file name are much more meaningful than those from other mechanisms:

The screenshot shows the 'X3 session log' interface. The breadcrumb path is 'All > Administration > Usage > Logs'. The main heading is 'X3 session log'. There are four tabs: 'File', 'Target', 'Levels selection', and 'Attached X3 sessions'. The 'File' tab is selected. The 'File' section shows a table with columns 'Label' and 'Description'. The 'Label' column contains 'Flag_62' and the 'Description' column contains 'Flags 62'. Both are circled in red. Below the 'Label' is 'Expired at' with the value '2022-09-21T01:52:07.326Z'. The 'Target' section shows a preview of a Notepad file named '20220920155223921_adonix_62_admin_3408_Flag_62_.tra - Notepad'. The content of the file is visible, showing '<channel 62>Flags 62'. A red arrow points from the 'Flag_62' label to the 'Flags 62' description. Below the screenshot, the text 'FLAG VALUE IS 62' is displayed.

How to enable and disable Engine Traces

3. X3 session log



<input type="checkbox"/>	Code	Description	Value
<input checked="" type="checkbox"/>	LOG_CLNT	Classical mode exchanges	32
<input type="checkbox"/>	LOG_DEBUG	Standard debug trace	512
<input type="checkbox"/>	LOG_FILES	File access time	4096
<input type="checkbox"/>	LOG_JSON	JSON exchanges	16
<input type="checkbox"/>	LOG_LDAP	LDAP requests management service	64
<input type="checkbox"/>	LOG_MEMORY	Memory usage	8192
<input type="checkbox"/>	LOG_OPAD	Opadxd exchanges	128
<input checked="" type="checkbox"/>	LOG_SQL0	Read and For loop SQL Instructions	4
<input checked="" type="checkbox"/>	LOG_SQL1	Oracle(sadora) / SqlServer(sadoss) services	8
<input type="checkbox"/>	LOG_STRT	Runtime initialization	256
<input type="checkbox"/>	LOG_SYSC	Security checks	1024
<input checked="" type="checkbox"/>	LOG_TRT0	Function calls with timing(such as 'Gosub' and 'Call')	1
<input checked="" type="checkbox"/>	LOG_TRT1	All 4GL instructions	2
<input type="checkbox"/>	LOG_X3SM	Session management	2048

Levels selection

Levels

LOG_TRT1 All 4GL instructions LOG_CLNT Classical mode exchanges LOG_TRT0 Function calls with timing(such as 'Gosub' and 'Call') LOG_SQL1 Oracle(sadora) / SqlServer(sadoss) services LOG_SQL0 Read and For loop SQL Instructions

Levels digest

47

How to enable and disable Engine Traces

3. X3 session log

X3 session log

File Target Levels selection Attached X3 sessions

File

Label	Description	Enabled	Max log time (mn)
Flag_62	Flags 62	x	

Target

Type	Endpoint	Function (classic)
Function (classic)	X3ERP12 / SEED Endpoints describe services locations	: GESSOH

User

Levels selection

Levels

[LOG_CLNT](#) Classical mode exchanges : [LOG_JSON](#) JSON exchanges : [LOG_SQL1](#) Oracle(sadora) / SqlServer(sadoss) services : [LOG_SQL0](#) Read and For loop SQL Instructions : [LOG_TRT1](#) All 4GL instructions :

Levels digest

62

Attached X3 sessions

Type	X3 Session	X3 Pid	Status	X3 host	X3 port	X3 user login	X3 locale code	Solution	Folder	Function	Web Service	Soap P
: Classic page	12099	3408	Used	x3erp12sqlvm	50012	admin	en-GB	X3ERP12	SEED	GESSOH		

How to enable and disable Engine Traces

4. “start_log” – the Nuclear Option

As hinted-at earlier, there is a fourth way to enable Tracing: `start_log`. This is the name of a file which needs to be placed in the `runtime\tmp` folder, and it should contain a flag-value.

The actual Trace Files generated by this will be created in `runtime\tmp` folder as well.

I call this the “Nuclear Option” because it will record activity for all adonix sessions, including Batch Server and Web Service Pools, so it would be wise to stop Batch Server and Web Services Pools if possible in order to avoid confusion and too many spurious tra-files.

Even with just the Batch Server stopped, there seems to be a regular set of trace-files created for each Web Services Pool Channel.

It is required that you stop and restart the runtime when you create `start_log` and then when you remove/rename it.

This option should ONLY be used under supervision of Sage Support.

How to enable and disable Engine Traces

This option and X3 session log have to be used if you want to do a trace involving Mode 16 or 256.

As this affects all sessions, it must be used sparingly and for the minimal amount of time – once you've carried out the necessary task, remove the start_log file before doing anything else. Ideally, find out the process-ids of the Task before it's complete so you know which tra-file(s) are relevant.

This option and X3 Session log are the ones to use if you want to get a trace of what's going on before control is given over to the screen – before that point, you won't be able to enable Tracing through Diagnostics>Calculator.

To 256...

To 256...

But the above Modes are only the beginning – as mentioned in my previous BP Presentation (see Appendix), you can combine these Modes to record more “Channels”.

As you might remember from my previous Presentation on the subject, my “go-to” setting is “7”, so I’d do OpenLog(“TRA”,7) – this will generate an Engine Trace file combining Modes 1, 2 and 4 – i.e. channels 1, 2, 3 and 4

```

<channel 2>@X3.TRT/ASYRSUB$adx(440)          | | | | | | | | | | | | | | Next
<channel 2>@X3.TRT/ASYRSUB$adx(442)          | | | | | | | | | | | | | | Return
<channel 1>@SEED.TRT/WMC0ACTXPARAM$adx(47)   | | | | | | | | | | | | | | End Gosub, tick:176022
<channel 2>@SEED.TRT/WMC0ACTXPARAM$adx(48)   | | | | | | | | | | | | | | Return
<channel 1>@SEED.STC/C_ACTXPARAM$adx(326)    | | | | | | | | | | | | | | End Gosub, tick:176022
<channel 2>@SEED.STC/C_ACTXPARAM$adx(327)    | | | | | | | | | | | | | | End
<channel 1>@X3.TRT/SUBAUS$adx(1974)          | | | | | | | | | | | | | | End Method, tick:176022
<channel 2>@X3.TRT/SUBAUS$adx(1987)          | | | | | | | | | | | | | | End
<channel 1>@X3.TRT/CALCULETTE$adx(239)       | | | | | | | | | | | | | | End Call, tick:176022
<channel 2>@X3.TRT/CALCULETTE$adx(240)       | | | | | | | | | | | | | | Read
<channel 4>Execution SQL on Read clause in @X3.TRT/CALCULETTE$adx at line 239, tick : 176022
<channel 4>Select AKL_.ROWID, AKL_.* From SEED.ACALCUL AKL_ Where ( AKL_.USR_0 = ? And AKL_.NUM_0 = ? ) Order by AKL_.USR_0,AKL_.NUM_0 Option (FAST 1)
<channel 4>Query end, tick : 176049
<channel 2>@X3.TRT/CALCULETTE$adx(241)       | | | | | | | | | | | | | | If
<channel 2>@X3.TRT/CALCULETTE$adx(242)       | | | | | | | | | | | | | | Assign
<channel 2>@X3.TRT/CALCULETTE$adx(243)       | | | | | | | | | | | | | | Assign
<channel 2>@X3.TRT/CALCULETTE$adx(244)       | | | | | | | | | | | | | | If

```

256...

I've created a spread-sheet to summarise how the Tracing actually behaves when combining different Modes to generate the Flag Value used at the time of enabling the Trace (in whichever mechanism you decide to use)

In the following slides, I show the behaviour around each of the official Modes (i.e. 1, 2, 4, 8, 16, 32, 64, 128 and 256).

There are, of course, more combinations but I just show the “highlights”.

So, the TRA-files will record activity for the channels in each cell with a cross in it – as per my Flag 7 example above.

256...

The different combinations around the first three Modes (1, 2 and 4) can be summarised as follows:

	A	B	C	D	E	F
1	Flag Value	1	2	3	4	8
2	1	x		x		
3	2		x	x		
4	3	x	x	x		
5	4				x	
6	5	x		x	x	
7	6		x	x	x	
8	7	x	x	x	x	
9	8					x

TRACE VALUE (pointing to cell A9)

<CHANNEL ???> (pointing to cell F1)

256...

This shows the different combinations when using Mode 8 which records sadoss/sadora calls.

1	Flag Value	1	2	3	4	8
9	8					x
10	9	x		x		x (separate file)
11	10		x	x		x (separate file)
12	11	x		x		x (separate file)
13	12				x	x (separate file)
14	13	x		x	x	x (separate file)
15	14		x	x	x	x (separate file)
16	15	x	x	x	x	x (separate file)

Note the “separate file” comment for Mode 8 – this is because Flag values 9 to 15 combine “sadoss/sadora” Trace Mode 8 with “adonix” Trace Modes 1, 2 and 4 – these two types of Trace go into separate Trace Files.

There is no way to “link” the content of the sadxxx and adonix Trace-files.

256...

Here's the behaviour from 16 to 20 when you want to record JSON traffic

1	Flag Value	1	2	3	4	8	16
17	16*						x
18	17*	x		x			x
19	18*		x	x			x
20	19*	x	x	x			x

Note the asterisk – this means the `start_log` should be used, rather than `Diagnosis > Calculator` to get any output

256...

Here's the behaviour involving Mode 32 if you're very interested in tracing Screen-handling activity

Flag Value	1	2	3	4	8	16	32
32							X
33	X		X				X
34		X	X				X
35	X	X	X				X
36				X			X
37	X		X	X			X
38		X	X	X			X
39	X	X	X	X			X

256...

When I tried using Mode 64 to see LDAP traffic, there was no output as it was only relevant to V6.

Flag Value	1	2	3	4	8	16	32	64	128
35	x	x	x				x		
36				x			x		
37	x		x	x			x		
38		x	x	x			x		
39	x	x	x	x			x		
64									
65	x		x						
66		x	x						
67	x	x	x						

256...

Here's the behaviour from 128 to record opadxd / JavaBridge traffic (even if JavaBridge isn't installed) – the content without JavaBridge doesn't look very informative...

Flag Value	1	2	3	4	8	16	32	64	128
128									x
129	x		x						x
130		x	x						x
131	x	x	x						x
132				x					x
255	x	x	x	x	x (separate file)	x	x		x

Again, as you go up the scale of Flag Values, more Modes are included – right up to the maximum of 255 where all of them are being recorded (even if Mode 64 is not outputting).

256...

Mode 256

Here are some highlights for 256 plus:

Flag Value	1	2	3	4	8	16	32	64	128	256
255	x	x	x	x	x (separate file)	x	x		x	
256*	x		x							x
257*	x		x							x
258*		x	x							x
259*	x	x	x							x
260*				x						x

... and Beyond

... and beyond

While preparing for this Presentation, I came across 2 additional, undocumented Modes:

Mode 512

This mode records every code-block call and variables as well as the runtime stack and 4GL Call Stack, so you can see the call-nesting.

Mode 514

This mode records the values of Systems variables.

... and beyond

Mode 512

This mode records every code-block call and variable values as well as the runtime stack and 4GL Call Stack, so you can see the call-nesting.

It also records low-level calls which are related to tasks such as opening Tables

```
<channel 512>createSqlKey
<channel 512>loadFde(filnom:'ADOPAR', fileName:'ADOPAR' apl:'' application:'SEED')
<channel 512>serverStart(OSSSRV (8),,,SEED)
<channel 512>serverInitialize(host:,hapl:,tyc:8=OSSSRV (8))
<channel 512>getHostSlot()
<channel 512>getHostSlot returns srvId=1
<channel 512>comSlot(OSSSRV (8),1)
<channel 512>f_load(@SEED.FIL/ADOPAR)
<channel 512>tblouv(@SEED.FIL/ADOPAR,ADP)
```

... and beyond

<channel 512>Local variables:

<channel 512>Integer - I=21

<channel 512>Integer - J=13

<channel 512>Integer - K=0

<channel 512>Char(255) - WABRFIC=SLT

<channel 512>Char(20) - WDTRNUM=STD

<channel 512>Integer - WDTRTYP=5

<channel 512>Char(3) - WMODELE=ARE

<channel 512>Char(5) - WSITE=AE011

<channel 512>

<channel 512>4GL stack:

<channel 512>@X3.TRT/TRTX3CPT\$adx(1021):X3.TRTX3CPT.ACTUALIS_FMTDIE23

<channel 512> from @X3.TRT/TRTX3CPT\$adx(771):TRTX3CPT.GETCPY

<channel 512> from @X3.TRT/SUBSIHA\$adx(6436):SUBSIHA.CHARG_PARAM

<channel 512> from @X3.TRT/SUBSIHA\$adx(1093):LIENS

<channel 512> from @X3.TRT/SUBSIH\$adx(49):ACTION

<channel 512> from @X3.TRT/GOBJSUB\$adx(2642):GOBJSUB.ACTION

<channel 512> from @X3.TRT/GOBJSUB\$adx(2459):GOBJSUB.LECTURE

<channel 512> from @X3.TRT/GOBJSUB\$adx(1597):GOBJSUB.SUIVANT

<channel 512> from @X3.TRT/GOBJSUB\$adx(221):CHOI

<channel 512> from @X3.TRT/GOBJET\$adx(881):GOBJET.OBJET

<channel 512> from @X3.TRT/GOBJET\$adx(25):GOBJET.OBJET_CHAR

... and beyond

```
<channel 512>runtime stack:  
<channel 512>48: flushContext - 0x44D21CE0, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\adxerr.c.459  
<channel 512>47: err_adx - 0x44D220B8, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\adxerr.c.560  
<channel 512>46: gp_var - 0x44CF750C, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\calvar.c.2052  
<channel 512>45: dim_f - 0x44CF4AB0, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\calvar.c.227  
<channel 512>44: calcul - 0x44CF2F1C, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\acalc.c.68  
<channel 512>43: ccompar - 0x44CF437C, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\acalc.c.549  
<channel 512>42: cgrth - 0x44CF446C, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\acalc.c.591  
<channel 512>41: calcul - 0x44CF2F1C, d:\jenkinsslave\workspace\runtime_release_release_r094.001@2\src\acalc.c.68
```

So, this is interesting – being able to see the values of variables at runtime means you can almost see the data flowing – variables are used when constructing SQL Statements.

... and beyond

Mode 514

Well, it was all going so well – we had a very logical set of Modes before we got to this point!

You'd expect 514 to be 512 + 2, but you get extras in the shape of Systems variable such as COUZON and ACTION, as well as Mode 2 channels 2 and 3.

```
<channel 512>clientReadValue(1)
<channel 512>clientReadValue(1) type:378 length:20
<channel 512>clientReadValue(1) st:0 type:381
<channel 514>REPONSE=98
<channel 2>@SEED.TRT/WGWOSIHSTD$adx(192) | | | | | | | Return
<channel 2>@X3.TRT/GOBJSUB$adx(66) | | | | | | | Assign
<channel 514>COUZON=NUM
<channel 2>@X3.TRT/GOBJSUB$adx(67) | | | | | | | Onevent
<channel 2>@X3.TRT/GOBJSUB$adx(68) | | | | | | | Assign
<channel 514>ACTION=APRES_CHOI
<channel 3>@X3.TRT/GOBJSUB$adx(68) | | | | | | | Gosub ACTION , tick:2347025
<channel 2>@X3.TRT/GOBJSUB$adx(2543) | | | | | | | If
<channel 2>@X3.TRT/GOBJSUB$adx(2552) | | | | | | | If
```

... and beyond

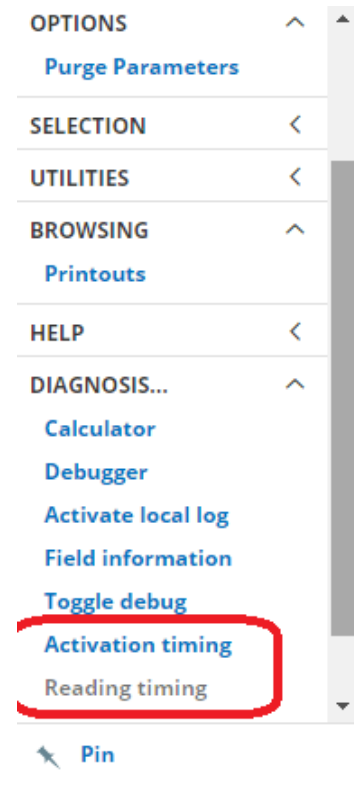
As we've just seen, you *can* get Modes beyond the official 128, and the combinations of the Modes are more complex for 512 and 514. So, this is another quick summary of the different combinations – other combinations are available....

Flag Value	1	2	3	4	8	16	32	64	128	256	512	514
512											X	
513	X		X								X	
514		X	X								X	X
515	X	X	X								X	X
516				X							X	
517	X		X	X							X	
518		X	X	X							X	X
519	X	X	X	X							X	X
520											X	
521	X		X		x (separate file)						X	
522		X	X		x (separate file)						X	X
523	X	X	X		x (separate file)						X	X
524				X	x (separate file)						X	
527	X	X	X	X	x (separate file)						X	X

Timing Logs and Engine Traces

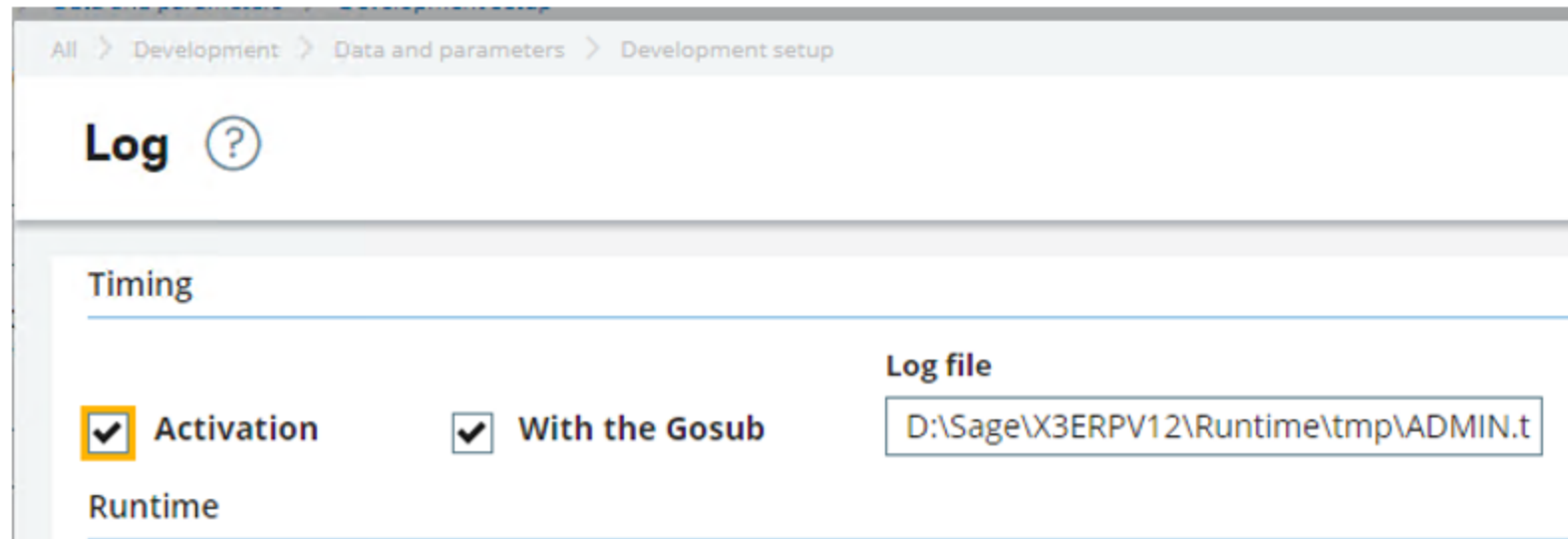
Timing Logs and Engine Traces

X3 has a built-in facility to record and analyse the execution times of each call to a subroutine or function – this is accessed via the options Activation timing and Reading timing which turn on and turn off the Timing Trace respectively.



Timing Logs and Engine Traces

This feature generates a tra-file in the Folder's TRA directory and a report which defaults to being called <X3-user>.tra in the runtime\tmp folder:



The screenshot shows a web-based configuration interface for logging. At the top, there is a breadcrumb navigation: "All > Development > Data and parameters > Development setup". Below this is a section titled "Log" with a help icon (?). Underneath, there is a "Timing" section with a horizontal line below it. In this section, there are three items: "Activation" with a checked checkbox, "With the Gosub" with a checked checkbox, and "Log file" with a text input field containing the path "D:\Sage\X3ERP12\Runtime\tmp\ADMIN.t". Below the "Timing" section is a "Runtime" section, also separated by a horizontal line.

Timing Logs and Engine Traces

The “normal” F?????.tra file in the Folder’s TRA directory shows a summary of how long was spent in each code-block - the number of calls to the block, the total duration, and the percentage of the full time recorded.

↑ ↑ ↓ ↓ **Log reading F38966**

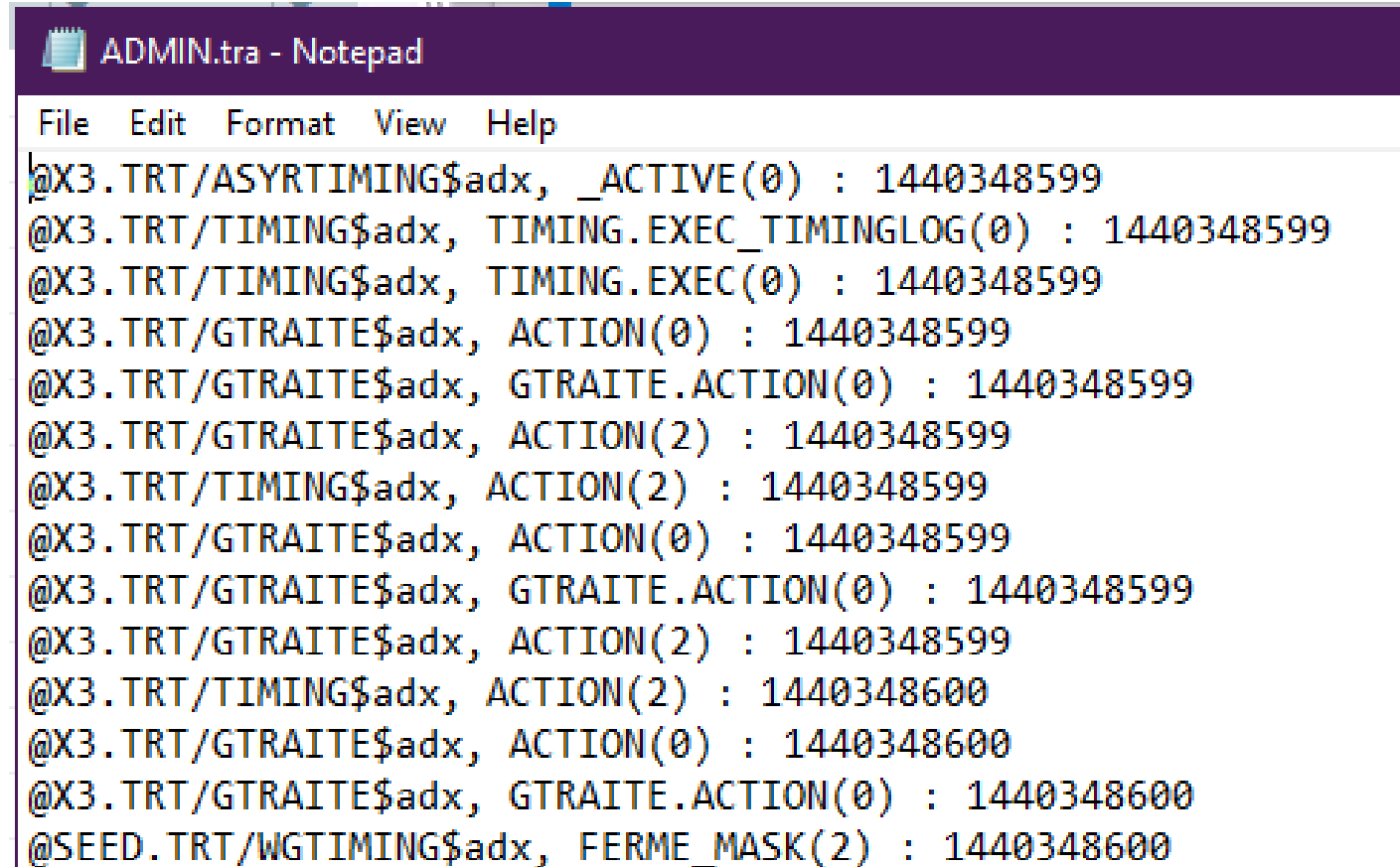


The screenshot shows a web-based interface for viewing a timing log. At the top, there are navigation icons (up, down) and the title "Log reading F38966". Below this is a search bar and a list of items. The first item is a header for the log entry: "19/09/22 15:05:08 (ADMIN)". The subsequent items are rows in a table, each starting with a number (1-7) and a vertical ellipsis icon. The table contains the following data:

Line	Code Block	Calls	Duration	Percentage
1	Total time : 341 ms -> 0:00:00:341 (hh:mm:ss:000)			
2	Gosub AFFZONE From GOBJSUB	1	31 ms	9.09 %
3	Call SHOW_HIDE_PREPAYMENT_INVOICE_FIELDS From SUBS	1	18 ms	5.27 %
4	Call ACTIV From AFNC	22	15 ms	4.39 %
5	Call IMAGE_COM From GOBJACT	1	12 ms	3.51 %
6	Call ISWINTOGENER From INTRUTILB	1	11 ms	3.22 %
7	Gosub ACTION_MET From ASYRSUB	240	10 ms	2.93 %

Timing Logs and Engine Traces

The other Trace File, typically runtime\tmp\



```
ADMIN.tra - Notepad
File Edit Format View Help
@X3.TRT/ASYRTIMING$adx, _ACTIVE(0) : 1440348599
@X3.TRT/TIMING$adx, TIMING.EXEC_TIMINGLOG(0) : 1440348599
@X3.TRT/TIMING$adx, TIMING.EXEC(0) : 1440348599
@X3.TRT/GTRAITE$adx, ACTION(0) : 1440348599
@X3.TRT/GTRAITE$adx, GTRAITE.ACTION(0) : 1440348599
@X3.TRT/GTRAITE$adx, ACTION(2) : 1440348599
@X3.TRT/TIMING$adx, ACTION(2) : 1440348599
@X3.TRT/GTRAITE$adx, ACTION(0) : 1440348599
@X3.TRT/GTRAITE$adx, GTRAITE.ACTION(0) : 1440348599
@X3.TRT/GTRAITE$adx, ACTION(2) : 1440348599
@X3.TRT/TIMING$adx, ACTION(2) : 1440348600
@X3.TRT/GTRAITE$adx, ACTION(0) : 1440348600
@X3.TRT/GTRAITE$adx, GTRAITE.ACTION(0) : 1440348600
@SEED.TRT/WGTIMING$adx, FERME_MASK(2) : 1440348600
```

Timing Logs and Engine Traces

Note that V12 Patch 27 introduced a new feature – in the same way as X3 session logs has been enhanced - you can now launch the equivalent of an OpenLog() by ticking the second Activation button in the “Runtime” section, and select the Flag-values.

The screenshot shows a web interface for configuring logging. At the top, there is a breadcrumb trail: 'All > Sales > Orders'. Below this is a 'Log' header with a help icon. The interface is divided into two main sections: 'Timing' and 'Runtime'. In the 'Timing' section, there are three checkboxes: 'Activation' (unchecked), 'With the Gosub' (unchecked), and 'Log file' (with an empty text input field). In the 'Runtime' section, there is a checked 'Activation' checkbox and a 'Level' dropdown menu currently set to '0'. The dropdown menu also contains a search icon and a list icon.

The Tracing is enabled for the current function and only stops recording when the Trace is disabled.

Timing Logs and Engine Traces

The screenshot shows a 'Log level selection' dialog box with a table of 15 log levels. Each level has a checkbox for selection and a corresponding mode value. The modes are powers of 2, ranging from 1 to 8192.

Level	Description	MODE
1	Function calls with timing (such as 'Gosub' and 'Ca	1
2	All 4GL instructions	2
3	Read and For loop SQL Instructions	4
4	Oracle (sadora) / SqlServer (sados) services	8
5	JSON exchanges	16
6	Classical mode exchanges	32
7	LDAP requests management services	64
8	Opadxd exchanges	128
9	Runtime initialization	256
10	Standard debug trace	512
11	Security checks	1024
12	Session management	2048
13	File access time	4096
14	Memory usage	8192
15		

This shows that there are even more modes which aren't normally publicised!

Timing Logs and Engine Traces

Mode 1024

This is related to the sandbox – for example, loading files and attachments.

```
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
<channel 1024>Sandbox management(find_syst_pattern ) - Sandbox disabled (is_admin)
```


Timing Logs and Engine Traces

Mode 2048 – Session Management

This Mode records the starting-up of several phases of X3 sessions and should be enabled via `start_log` for most effect (i.e. recording the login/logout):

Logging in

```
x3diary_adonix_9968_0.tra - Notepad
File Edit Format View Help
<channel 2048>Write main session internal with database id:60, 2022-09-13 18:20:30.177
<channel 2048>Write main session internal with uid :11223
<channel 2048>found another 60, deleting current
<channel 2048>Delete ok
<channel 2048>delete ASYSSMPROCES on SESSIONID= 11222
<channel 2048>delete ASYSSMEXTERN on SESSIONID= 11222
<channel 2048>delete ASYSSMINTERN on SESSIONID= 11222
<channel 2048>add_process 9968 adonix on X3ERP12SQLVM
<channel 2048>Process created with id 22427
<channel 2048>add_process 8524 sadoss on X3ERP12SQLVM
<channel 2048>Process created with id 22428
```

Timing Logs and Engine Traces

Mode 2048 – Session Management

Launching Sales Orders

```
x3diary_adonix_1104_0.tra - Notepad
File Edit Format View Help
<channel 2048>Write main session internal with database id:60, 2022
<channel 2048>Write main session internal with uid :11848
<channel 2048>found another 60, deleting current
<channel 2048>Delete ok
<channel 2048>delete ASYSSMPROCES on SESSIONID= 11846
<channel 2048>delete ASYSSMEXTERN on SESSIONID= 11846
<channel 2048>delete ASYSSMINTERN on SESSIONID= 11846
<channel 2048>add_process 1104 adonix on X3ERP12SQLVM
<channel 2048>Process created with id 23677
<channel 2048>add_process 7064 sadoss on X3ERP12SQLVM
<channel 2048>Process created with id 23678
```

Timing Logs and Engine Traces

Mode 2048 – Session Management

Logging out of X3

```
x3diary_adonix_1104_0.tra - Notepad
File Edit Format View Help
|<channel 2048>Write main session internal with database id:60, 202:
<channel 2048>Write main session internal with uid :11848
<channel 2048>found another 60, deleting current
<channel 2048>Delete ok
<channel 2048>delete ASYSSMPROCES on SESSIONID= 11846
<channel 2048>delete ASYSSMEXTERN on SESSIONID= 11846
<channel 2048>delete ASYSSMINTERN on SESSIONID= 11846
<channel 2048>add_process 1104 adonix on X3ERP12SQLVM
<channel 2048>Process created with id 23677
<channel 2048>add_process 7064 sadoss on X3ERP12SQLVM
<channel 2048>Process created with id 23678
<channel 2048>Compute user number
<channel 2048>Number of User on SEED : 6
```

Timing Logs and Engine Traces

Flag Value 2049 – Session Management Mode 2048 plus Mode 1

This shows how just adding one to the flag-value can increase the tracing quite substantially:

Launching Sales Orders

```
<channel 2048>Write main session internal with database id:60, 2022-09-19 18:53:57.257
<channel 2048>Write main session internal with uid :11852
<channel 2048>found another 60, deleting current
<channel 2048>Delete ok
<channel 2048>delete ASYSSMPROCES on SESSIONID= 11848
<channel 2048>delete ASYSSMEXTERN on SESSIONID= 11848
<channel 2048>delete ASYSSMINTERN on SESSIONID= 11848
<channel 2048>add_process 1312 adonix on X3ERP12SQLVM
<channel 2048>Process created with id 23685
<channel 2048>add_process 472 sadoss on X3ERP12SQLVM
<channel 2048>Process created with id 23686
<channel 3>@X3.TRT/MENU$adx(10)                               Func EXISTE , tick:290
<channel 3>@X3.TRT/MENU$adx(375)                               | Call EXISTE_ADX From ORDSYS
<channel 1>@X3.TRT/MENU$adx(375)                               | End Call, tick:295
<channel 1>@X3.TRT/MENU$adx(10)                               End Func, tick:295
```

Launching GESSOH

```
<channel 3>@X3.TRT/GOBJET$adx(1334)                            | | | | | | | Gosub ACTION From GOBJSUB, tick:27665
<channel 3>@X3.TRT/GOBJSUB$adx(2642)                            | | | | | | | Gosub ACTION From SUBSOH, tick:27665
<channel 3>@X3.TRT/SUBSOH$adx(26)                               | | | | | | | Gosub ENTREE From EXEFNC, tick:27665
<channel 1>@X3.TRT/SUBSOH$adx(26)                               | | | | | | | End Gosub, tick:27665
```

Timing Logs and Engine Traces

Flag Value 2049 – Session Management Mode 2048 plus Mode 1

This shows how just adding one to the flag-value can increase the tracing quite substantially:

Exiting Sales Orders and logging out – this is recorded in the Function's adonix trace-file as

```
<channel 1>@X3.TRT/COMPFUN$adx(226)
<channel 1>@X3.TRT/AWRK$adx(134)
<channel 1>@X3.TRT/FIN$adx(26)
<channel 3>@X3.TRT/FIN$adx(27)
<channel 2048>Compute user number
<channel 2048>Number of User on SEED : 6
<channel 3>@X3.TRT/GESUSER$adx(1929)
```

```
| | | | End Func, tick:155135
| | | | End Gosub, tick:155135
| | | | End Call, tick:155138
| | | | Call DECONNECT From GESUSER, tick:155138
| | | | Call ESPION From ESPION, tick:155140
```

Followed by

```
<channel 1>@X3.TRT/GESUSER$adx(1958)
<channel 1>@X3.TRT/GESUSER$adx(1947)
<channel 1>@X3.TRT/GESUSER$adx(1933)
<channel 1>@X3.TRT/FIN$adx(27)
<channel 1>@X3.TRT/FIN$adx(3)
```

```
| | | | End Gosub, tick:155156
| | | | End Gosub, tick:155163
| | | | End Call, tick:155163
| | | | End Call, tick:155163
End Call, tick:155163
```

Timing Logs and Engine Traces

Flag Value 2049 – Session Management Mode 2048 plus Mode 1

Logging out – this is recorded in the Initial Login's adonix trace-file as

```
<channel 3>@X3.TRT/AWRK$adx(226)
<channel 1>@X3.TRT/AWRK$adx(226)
<channel 1>@X3.TRT/AWRK$adx(134)
<channel 1>@X3.TRT/FIN$adx(26)
<channel 3>@X3.TRT/FIN$adx(27)
<channel 2048>Compute user number
<channel 2048>Number of User on SEED : 6
<channel 3>@X3.TRT/GESUSER$adx(1929)
```

```
      | | | Func DIMPLED , tick:38559
      | | | End Func, tick:38560
      | | | End Gosub, tick:38560
      | | End Call, tick:38563
      | Call DECONNECT From GESUSER, tick:38563
      |
      | Call ESPION From ESPION, tick:38565
```

And at the end of the Trace file:

```
<channel 1>@SEED.STC/C_WMACONTEXTAFOLD$adx(72)
<channel 1>@SEED.STC/C_WMACONTEXTAFOLD$adx(66)
<channel 3>@X3.TRT/ASYRCONNECT$adx(187)
<channel 1>@X3.TRT/ASYRCONNECT$adx(187)
<channel 1>@X3.TRT/FIN_SYRA$adx(3)
```

```
      | | | End Gosub, tick:470976
      | | End Gosub, tick:470976
      | | End Method, tick:470976
      | Gosub SEARCH_ALOGIN , tick:470976
      | End Gosub, tick:470976
      End Call, tick:470977
```

Timing Logs and Engine Traces

Mode 4096 – File Access Time

```
File Edit Format View Help
<channel 4096>;path;function;elapsed;tick;misc;CLOCKS_PER_SEC:1000;
<channel 4096>;@!.tmp/start_log;close;0;0;0;
<channel 4096>;@!.sem/semaphore$map;open;1;4;;
<channel 4096>;@!.sem/semaphore$map;stat;0;4;8;
<channel 4096>;@!.lan/ISO;open;0;4;;
<channel 4096>;@!.lan/ISO;close;0;4;0;
<channel 4096>;@SEED..users;stat;0;4;0;
<channel 4096>;@!.tmp/vproto;stat;0;4;10;
<channel 4096>;@!.lan/ENG;open;0;4;;
<channel 4096>;@SEED.valfil$timestamp;open;0;14;;
<channel 4096>;@SEED.valfil$timestamp;close;1;15;0;
<channel 4096>;@SEED.valtrt$timestamp;open;0;15;;
<channel 4096>;@SEED.valtrt$timestamp;close;0;15;0;
<channel 4096>;@SEED.APL$ini;open;2;18;;
<channel 4096>;@X3..users;stat;0;19;0;
<channel 4096>;@X3..users;stat;0;19;0;
<channel 4096>;@SEED.valfil$timestamp;open;0;20;0;
```

Timing Logs and Engine Traces

Mode 4100 – File Access Time (4096) plus Modes 1, 2, 3 and 4

Again, adding extra Modes to the Flag Value enhances the Tracing

```
<channel 4096>;@X3.TRT/ADP_TOOL$adx;close;0;178;0;
<channel 4096>;@SEED.FIL/ADOVAL$fde;open;0;178;;
<channel 4096>;@SEED.FIL/ADOVAL$fde;close;0;178;0;
<channel 4096>;@SEED.FIL/ADOPAR$fde;open;0;179;;
<channel 4096>;@SEED.FIL/ADOPAR$fde;close;0;179;0;
<channel 4>Execution SQL on Read clause in @X3.TRT/ADP_TOOL$adx at line 183, tick : 179
<channel 4>Select ADP_.ROWID, ADP_.* From SEED.ADOPAR ADP_ Where ( ADP_.PARAM_0 = ? ) Order by ADP_.PARAM_0 Option (FAST 1)
<channel 4>Query end, tick : 180
<channel 4>Execution SQL on Read clause in @X3.TRT/ADP_TOOL$adx at line 485, tick : 180
<channel 4>Select ADW_.ROWID, ADW_.* From SEED.ADOVAL ADW_ Where ( ADW_.CMP_0 = ? And ADW_.FCY_0 = ? And ADW_.PARAM_0 = ? )
<channel 4>Query end, tick : 181
<channel 4096>;;close;0;237;0;
<channel 4>Execution SQL on Read clause in @X3.TRT/ADP_TOOL$adx at line 183, tick : 238
<channel 4>Select ADP_.ROWID, ADP_.* From SEED.ADOPAR ADP_ Where ( ADP_.PARAM_0 = ? ) Order by ADP_.PARAM_0 Option (FAST 1)
<channel 4>Query end, tick : 239
```


Timing Logs and Engine Traces

Mode 8192 – Memory Usage

This records the memory usage of each Adonix process – the last figure is the adonix process's heap-size in bytes.

```
THE    LUL    INITIAL    VIEW    HEAP
|-----|
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;6;297199;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;6;301296;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;16;301400;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;16;301500;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;0;303184;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;0;303632;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;0;303720;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;0;303733;
|<channel 8192>;Tue Sep 20 11:35:31 2022;HEAP;0;304733;
```

Timing Logs and Engine Traces

Mode 8207 – Memory Usage + 1, 2, 3, 4, 8

Again, adding extra Modes to the Flag Value enhances the Tracing

```
<channel 8192>;Tue Sep 20 11:53:47 2022;HEAP;6;35765289;
<channel 8192>;Tue Sep 20 11:53:47 2022;HEAP;6;35765289;
<channel 2>@X3.TRT/ATEXTRAB$adx(182)                | | | | | | | | | | | | | | | | If
<channel 2>@X3.TRT/ATEXTRAB$adx(183)                | | | | | | | | | | | | | | | | Read
<channel 4>Execution SQL on Read clause in @X3.TRT/ATEXTRAB$adx at line 182, tick : 1096344
<channel 8192>;Tue Sep 20 11:53:47 2022;HEAP;6;35765033;
<channel 4>Select AXX_.ROWID, AXX_.* From SEED.ATEXTRA AXX_ Where ( AXX_.CODFIC_0 = ? And AXX_.ZONE_0 = ? And AXX_.LANGUE_0 = ? And AXX_.IDENT1_0 = ? Ar
<channel 4>Query end, tick : 1096345
<channel 2>@X3.TRT/ATEXTRAB$adx(184)                | | | | | | | | | | | | | | | | If
<channel 2>@X3.TRT/ATEXTRAB$adx(187)                | | | | | | | | | | | | | | | | Read
<channel 4>Execution SQL on Read clause in @X3.TRT/ATEXTRAB$adx at line 186, tick : 1096346
<channel 8192>;Tue Sep 20 11:53:47 2022;HEAP;6;35765033;
<channel 4>Select AXX_.ROWID. AXX_.* From SEED.ATEXTRA AXX_ Where ( AXX_.CODFIC_0 = ? And AXX_.ZONE_0 = ? And AXX_.LANGUE_0 = ? And AXX_.IDENT1_0 = ? Ar
```

What now?

What now?

So, you're now the proud owner of one or more Engine Trace files – what do you do with them?

Well, you *could* send them to Sage Customer Services for analysis – especially if we've requested them!

OR

You could do some analysis in-house.

It depends on a few things:

- 1) The environment – has it got specific Customisations in it which may have an impact on the issue being investigated?
- 2) Is the Data/Configuration very complex – probably not recreateable outside this environment?
- 3) Who requested the trace?

What now?

What sorts of things can you do?

Example 1: Performance issue in one Folder

Well, the simplest thing you could do is start a Timing Trace – that would provide a high-level analysis of how long the program is in each code-block – it could give you a hint as to where the issue lies. If further analysis is required, the two trace-files could be sent to Sage Customer Services for analysis.

Another option is to run an Engine Trace such as OpenLog(“TRA”,7) and send it to Customer Services for analysis – the timing information is embedded in the Trace File in the form of the “tick” values which are in milliseconds. The advantage of this is, that it will show the flow through the program and also show the Database interactions – this will allow us to correlate it with the Code itself.

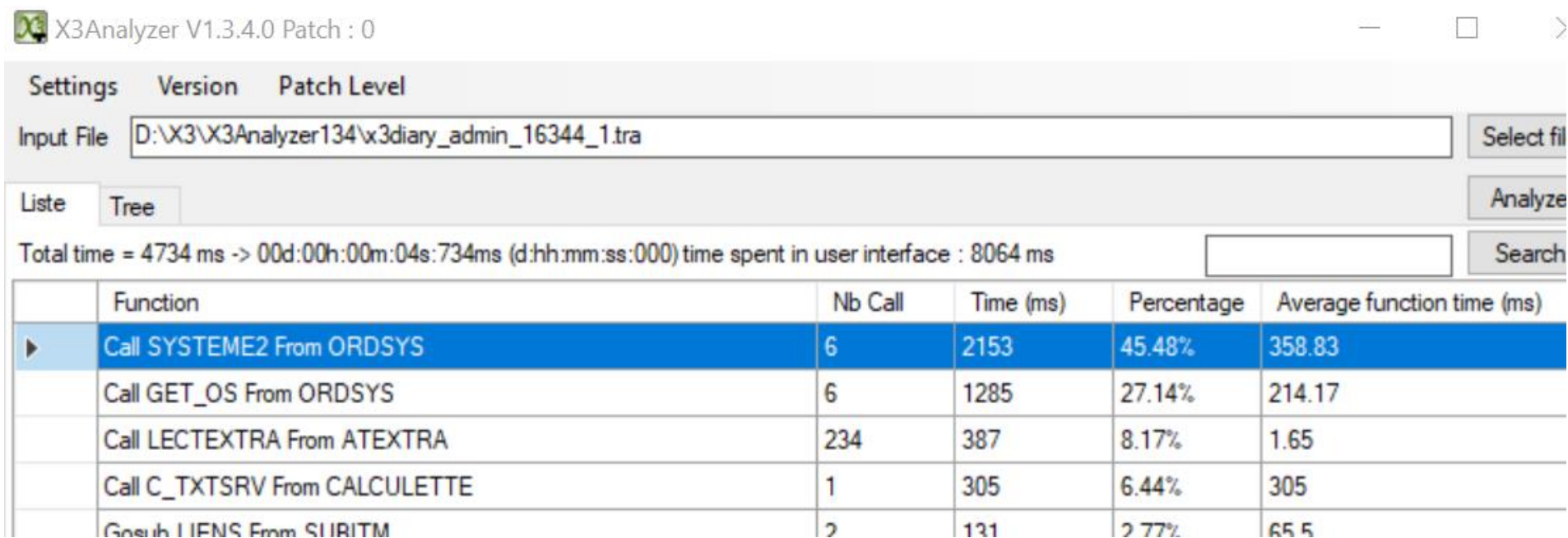
What now?

Example 1: Performance issue in one Folder

I had a case where a BP reported that certain functions were taking longer than expected to save new data or open the screens in a particular Folder but not in another Folder.

I asked for trace-files from both Folders and used our Internal X3Analyzer to produce a read-out of where the time was taken – the results were as follows

What now?



The screenshot shows the X3Analyzer V1.3.4.0 Patch : 0 interface. The 'Input File' field contains 'D:\X3\X3Analyzer134\x3diary_admin_16344_1.tra'. The 'Liste' tab is selected, and the table below displays the results of the analysis. The table has columns for Function, Nb Call, Time (ms), Percentage, and Average function time (ms). The row for 'Call SYSTEME2 From ORDSYS' is highlighted in blue, indicating it is the most significant call.

Function	Nb Call	Time (ms)	Percentage	Average function time (ms)
Call SYSTEME2 From ORDSYS	6	2153	45.48%	358.83
Call GET_OS From ORDSYS	6	1285	27.14%	214.17
Call LECTEXTRA From ATEXTTRA	234	387	8.17%	1.65
Call C_TXTSRV From CALCULETTE	1	305	6.44%	305
Call LIENS From SIRITM	2	131	2.77%	65.5

This shows that the largest amount of time was taken by calls to SYSTEME2 in ORDSYS.

With this in mind, we looked at the Trace file in more depth to see what was being executed within that code-block

What now?

```
channel 3>@X3.TRT/GOBSUB$adx(52)      | | | | Gosub SETBOUT , tick:285104
<channel 3>@X3.TRT/GOBSUB$adx(418)    | | | | Gosub ACTION , tick:285104
<channel 3>@X3.TRT/GOBSUB$adx(2618)   | | | | Gosub ACTION From XV1IMAGE_CHECK, tick:285104
<channel 3>@TEST.TRT/XV1IMAGE_CHECK$adx(14) | | | | Gosub SETBOUT , tick:285104
<channel 3>@TEST.TRT/XV1IMAGE_CHECK$adx(132) | | | | Call SYSTEME2 From ORDSYS, tick:285105
<channel 3>@X3.TRT/ORDSYS$adx(939)    | | | | Call GET_OS , tick:285105
<channel 1>@X3.TRT/ORDSYS$adx(939)    | | | | End Call, tick:285319
<channel 3>@X3.TRT/ORDSYS$adx(1073)   | | | | Call MACHINE , tick:285322
<channel 1>@X3.TRT/ORDSYS$adx(1073)   | | | | End Call, tick:285322
<channel 3>@X3.TRT/ORDSYS$adx(1151)   | | | | Gosub LIGNE , tick:285323
<channel 1>@X3.TRT/ORDSYS$adx(1151)   | | | | End Gosub, tick:285323
<channel 1>@TEST.TRT/XV1IMAGE_CHECK$adx(132) | | | | End Call, tick:285661
<channel 3>@TEST.TRT/XV1IMAGE_CHECK$adx(143) | | | | Call SYSTEME2 From ORDSYS, tick:285661
<channel 3>@X3.TRT/ORDSYS$adx(939)    | | | | Call GET_OS , tick:285662
<channel 1>@X3.TRT/ORDSYS$adx(939)    | | | | End Call, tick:285866
<channel 3>@X3.TRT/ORDSYS$adx(1073)   | | | | Call MACHINE , tick:285869
<channel 1>@X3.TRT/ORDSYS$adx(1073)   | | | | End Call, tick:285869
<channel 3>@X3.TRT/ORDSYS$adx(1151)   | | | | Gosub LIGNE , tick:285869
<channel 1>@X3.TRT/ORDSYS$adx(1151)   | | | | End Gosub, tick:285869
<channel 1>@TEST.TRT/XV1IMAGE_CHECK$adx(143) | | | | End Call, tick:286181
<channel 1>@TEST.TRT/XV1IMAGE_CHECK$adx(14) | | | | End Gosub, tick:286181
<channel 1>@X3.TRT/GOBSUB$adx(2618)   | | | | End Gosub, tick:286181
```


What now?

From experience, this shows that VersionOne's EDM Suite has been installed and Activated.

Looking at the Trace-file from the other Folder, there is no mention of XV1IMAGE_CHECK so EDM is not Active in the Folder.

The Business Partner confirmed this to be the case, and de-activated EDM – this resolved the issue

What now?

Example 2: Is it Standard?

As you all know, X3 is very customisable – either through Configuration settings, or by adding Customised Code.

Again, we can use `OpenLog("TRA",7)` to record the path taken during execution and see if there are any Custom executables being called.

Having taken a copy of the Trace-file, you can just find calls to Folder-specific executables :

```
type <trace-file> | findstr "@TEST.TRT" > non_standard.txt
```

For example `x3diary_reg_43334_0.tra | findstr "@TEST.TRT" > non_standard.txt`

This will produce a list of lines where the binary is in the Folder's TRT directory – either because it's generated by a Folder Validation, or it's Custom Code.

What now?

You can then weed-out most of the executables generated by a Validation

```
Type non_standard.txt | findstr /v "TRT/W" > not_validated.txt
```

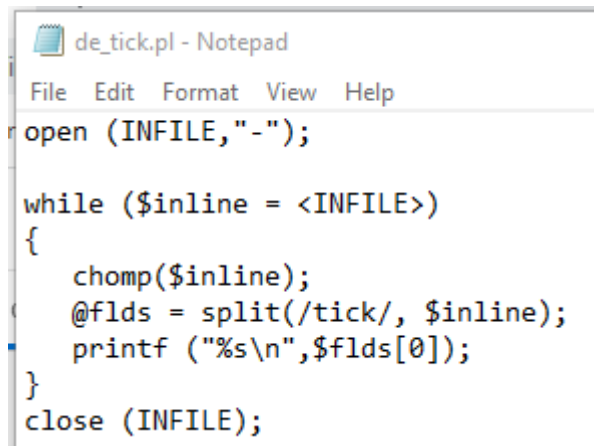
Now you can see if there are any executables with the name including "SPE"/"SPV" or starting with X, Y or Z – hopefully, the Developers are using the Naming Conventions!

What now?

Example 3: Spot the difference

You may be lucky enough to have an issue which occurs in one Folder or with one Site, but the function is ok with a different Folder or Site. In this sort of case, you could do separate Traces under the different conditions and then do a “diff” on them to see what is different.

Unfortunately, many lines will be different by default due to the “tick” values at the end of each line - I wrote a little bit of Perl Script to chop off the tick-values to avoid this



```
de_tick.pl - Notepad
File Edit Format View Help
open (INFILE, "-");

while ($inline = <INFILE>)
{
    chomp($inline);
    @flds = split(/tick/, $inline);
    printf ("%s\n", $flds[0]);
}
close (INFILE);
```

```
D:\X3\Support Tips and Tricks Notes\Autumn2022\flags>type x3diary_admin_flag007.tra | D:\TECH\pl_scripts\de_tick.pl > x3diary_1.tra
D:\X3\Support Tips and Tricks Notes\Autumn2022\flags>type x3diary_admin_flag007a.tra | D:\TECH\pl_scripts\de_tick.pl > x3diary_2.tra
```

What now?

Example 4: What did this process call?

I've had a case where adonix processes have continued to run, even when nobody's logged in and it looked like there was no way to determine what the adonix process was running because the Development > Utilities > Verifications > System monitor > Users (PSADX) option didn't show the Process ID seen in Task Manager – the processes weren't showing in ASYSINTERN, ASYSSMPROCES tables either.

One way of discovering what Function was associate with an adonix process is to set up start_log with Flag Value 1 – this will record the minimum amount of information for any adonix processes.

Then, you can look at the Trace file with that Process ID in runtime\tmp folder – search for “Gosub ENTREE From EXEFNC” and you'll find an instance which is a call to the Function

```
<channel 3>@X3.TRT/GOBJET$adx(1334) | | | | | | Gosub ACTION From GOBJSUB, tick:1882
<channel 3>@X3.TRT/GOBJSUB$adx(2642) | | | | | | | Gosub ACTION From SUBSOH, tick:1882
<channel 3>@X3.TRT/SUBSOH$adx(26) | | | | | | | Gosub ENTREE From EXEFNC, tick:1883
<channel 1>@X3.TRT/SUBSOH$adx(26) | | | | | | | End Gosub, tick:1883
<channel 1>@X3.TRT/GOBJSUB$adx(2642) | | | | | | | End Gosub, tick:1883
```

What now?

Example 4: What did this process call?

Fortunately, it can record more than one Function launch in the same Trace File, so it's best to go to the end of the file and search backwards

```
<channel 3>@X3.TRT/GOBJSUB$adx(55)          | | | | Gosub ACTION , tick:68149
<channel 3>@X3.TRT/GOBJSUB$adx(2642)       | | | | Gosub ACTION From SUBPOH, tick:68149
<channel 3>@X3.TRT/SUBPOH$adx(14)          | | | | Gosub ENTREE From EXEFNC, tick:68149
<channel 1>@X3.TRT/SUBPOH$adx(14)          | | | | End Gosub, tick:68149
<channel 3>@X3.TRT/SUBPOH$adx(49)          | | | | Gosub AVANT_CHOI From SUBPOHA, tick:68149
<channel 1>@X3.TRT/SUBPOH$adx(49)          | | | | End Gosub, tick:68149
<channel 1>@X3.TRT/GOBJSUB$adx(2642)       | | | | End Gosub, tick:68149
<channel 3>@X3.TRT/GOBJSUB$adx(2644)       | | | | Gosub AFF From GSATSTF. tick:68149
```

You could refine the search for the call by seeing what GOBJSUB line triggers the Action – in this case, both SUBSOH and SUBPOH were launched from the line @X3.TRT/GOBJSUB\$adx(2642).

If you want the Session ID and sadoss/sadora Process IDs as well, you could use Flag Value 2049

What now?

Example 4: What did this process call?

```
x3diary_adonix_8356_0.tra - Notepad
File Edit Format View Help
<channel 2048>Write main session internal with database id:59, 2022-09-30 17:18:02.110
<channel 2048>Write main session internal with uid :12282
<channel 2048>found another 59, deleting current
<channel 2048>Delete ok
<channel 2048>delete ASYSSMPROCES on SESSIONID= 12276
<channel 2048>delete ASYSSMEXTERN on SESSIONID= 12276
<channel 2048>delete ASYSSMINTERN on SESSIONID= 12276
<channel 2048>add_process 8356 adonix on X3ERP12SQLVM
<channel 2048>Process created with id 24545
<channel 2048>add_process 8236 sadoss on X3ERP12SQLVM
<channel 2048>Process created with id 24546
<channel 3>@X3.TRT/MENU$adx(10)                               Func EXISTE , tick:207
```

```
x3diary_adonix_8356_0.tra - Notepad
File Edit Format View Help
<channel 1>@X3.TRT/GOBJSUB$adx(2644)                          | | | | | | | | End Gosub, tick:494691
<channel 1>@X3.TRT/GOBJET$adx(1334)                          | | | | | | | | End Gosub, tick:494691
<channel 1>@X3.TRT/GOBJET$adx(1094)                          | | | | | | | | End Gosub, tick:494691
<channel 3>@X3.TRT/GOBJET$adx(1098)                          | | | | | | | | Gosub ACTION , tick:494707
<channel 3>@X3.TRT/GOBJET$adx(1334)                          | | | | | | | | Gosub ACTION From GOBJSUB, tick:494707
<channel 3>@X3.TRT/GOBJSUB$adx(2642)                          | | | | | | | | Gosub ACTION From SUBPOH, tick:494707
<channel 3>@X3.TRT/SUBPOH$adx(14)                            | | | | | | | | Gosub ENTREE From EXEFNC, tick:494707
<channel 1>@X3.TRT/SUBPOH$adx(14)                            | | | | | | | | End Gosub, tick:494708
<channel 1>@X3.TRT/GOBJSUB$adx(2642)                          | | | | | | | | End Gosub, tick:494708
<channel 3>@X3.TRT/GOBJSUB$adx(2644)                          | | | | | | | | Gosub AFF From CATETE, tick:494708
```

What now?

Example 5: Where did it go wrong?

Sometimes, a Function fails and doesn't update some tables – how do you find out where abouts this happens in the code?

Well, this calls for Mode 4 to be active for all Database events – as usual, I'd go for Flag Value 7 to get the best idea of the code-flow.

When our Applications update the Database, they do so within a Database Transaction. If a Database Transaction fails, then a “Rollback” is executed to revert any data updated within the Transaction's scope.

So, look for Trbegin which denotes the start of a Database Transaction and then Commit and Rollback denote the points at which Transactions are committed to the database on successful completion or rolled-back on failure.

What now?

Example 5: Where did it go wrong?

A Transaction that fails – Trbegin followed by a Rollback

<channel 3>@X3.IRI/GOBJET1\$adx(104)								Call DEBITRANS From GLOCK, tick:63887
<channel 2>@X3.TRT/GLOCK\$adx(83)								Assign
<channel 2>@X3.TRT/GLOCK\$adx(84)								Assign
<channel 2>@X3.TRT/GLOCK\$adx(85)								Assign
<channel 2>@X3.TRT/GLOCK\$adx(86)								Assign
<channel 2>@X3.TRT/GLOCK\$adx(87)								End
<channel 3>@X3.TRT/GOBJET1\$adx(105)								Gosub TR1 , tick:63887
<channel 2>@X3.TRT/GOBJET1\$adx(128)								Assign
<channel 2>@X3.TRT/GOBJET1\$adx(129)								Assign
<channel 2>@X3.TRT/GOBJET1\$adx(130)								Trbegin
<channel 3>@X3.TRT/GOBJET1\$adx(131)								Gosub DEFAULT From WOGAS, tick:63895
<channel 2>@TEST.TRT/WOGAS\$adx(16)								Default
<channel 2>@TEST.TRT/WOGAS\$adx(17)								Return
-----								-----
<channel 2>@X3.TRT/GOBJET1\$adx(173)								If
<channel 2>@X3.TRT/GOBJET1\$adx(174)								Goto
<channel 2>@X3.TRT/GOBJET1\$adx(190)								Rollback
<channel 2>@X3.TRT/GOBJET1\$adx(191)								Assign
<channel 3>@X3.TRT/GOBJET1\$adx(191)								Gosub ACTION From GOBJSUB, tick:67109
<channel 2>@X3.TRT/GOBJSUB\$adx(2520)								Tf

What now?

Example 6: What's wrong with my Web Services?

I've just had a case where a Web Service wasn't working, (giving a timeout error) – even in the Classic SOAP option – so we needed to investigate where it was failing.

So, I advocated using X3 session log with Trace Value 7 - I went for this as it allows for enabling Tracing “remotely” on a specific type of adonix sessions such as Web Services.

This showed that the problem was with a particular Read statement. Running the statement in SMSS gave the same behaviour, so the underlying view was modified and this improved matters.

Recap and Conclusions

What's it good for?

What do you get from it?

Self Help

What's it good for?

Well, it's good for gaining insight into what's going on within the **code** when you run Functions – you can see what adonix executables are being run and which code-blocks are being called.

You can see what **Database Statements** are being run – this can be useful in conjunction with Database Traces.

You're even privy to the values of runtime **variables**.

You can see how much **time** is spent within code-blocks – more information can be obtained by sending the Trace Files to Customer Services.

You can see what **JSON** text is being processed.

What do you get from it?

Well, this depends on what you put in!

As discussed, you can enable/disable Engine Tracing from five places:

1. Diagnosis > Calculator – good for ad hoc Tracing
2. 4GL Code – selective Tracing controlled by the Program
3. X3 session log – can configure and control Tracing for other sessions
4. Runtime\tmp\start_log – last restart!
5. Activation Timing – good for high-level timing investigations. Also overlaps mechanism 1

The goal of your investigation determines which mechanism you should use.

Your goal also determines what Flag Value you decide to use – what sort of aspect(s) of X3 are you looking at?

What do you get from it?

Where did it go?

1. Diagnosis > Calculator – Trace Files are output to the volume specified in the OpenLog() call – typically, to <folder>\TRA;
2. 4GL Code – as with Calculator, the Code will have the Volume to be written to x3diary_<user>_<proc-id>_<revision>.tra.
3. X3 session log – to runtime\logs\<timestamp>_adonix_<flags>_<user>_<proc-id>_<label>_<revision>.Tra
4. Start_log – this will output runtime\tmp\x3diary_<user>_<proc-id>_<revision>.tra
5. Activate Timing - to runtime\logs\<timestamp>_adonix_<flags>_<user>_<proc-id>_<function>_<revision>.Tra

What do you get from it?

The fourth mechanism seems to be different in V11 and V12 – in V11, it records Mode 8 in its separate file, but in V12, it doesn't seem to.

The fifth mechanism may be useful if you're interested in Mode 8 and don't want to use `start_log` but it does have the disadvantage that it doesn't record activity before control is handed-over to the User (to use Activation Timing to start the recording), and is a bit more cumbersome.

The Flag Value will also determine if you get a second Trace File for the `sadoss/sadora` activity by including Mode 8.

Personally, I can't see any need to use Mode 8.

Self Help

I have seen cases where Business Partners have been able to “zero-in” on where the issue lies, and this can be very helpful – for resolving issues internally, or for passing on as part of the initial Case comment.

I didn't test every Flag Value and my highlights from the spreadsheet are not comprehensive, so you could try your own Flag Value(s) – you could use the Mode-selector in Activation timing to calculate the Flag Value you want.

Remember to think about what sorts of information you want to get from the Trace – I'd go for a “More is better” approach if possible as you can always discount particular <channel> lines after getting the trace file in, but you can't get them if they're not there.

Remember that start_log is the last resort - virtually all scenarios will be covered by the other mechanisms, with X3 session log being a good substitute for start_log.

Appendix

Some additional reading

Previous presentations

- Index page: Sage X3 Technical Support Tips and Tricks (March 2021) [Index page: Sage X3 Technical Support Tips and Tricks \(March 2021\) - Sage X3 UK Support & Insights - Sage X3 UK - Sage City Community](#)
 - Introduction to tracing classic functions

Knowledgebase articles

- How to run an Engine Trace in Sage X3?

<https://support.na.sage.com/selfservice/viewdocument.do?externalId=73801>

- Microsoft's Extended Events and Sage X3?

<https://support.na.sage.com/selfservice/viewdocument.do?externalId=83828>

Sage City Blog articles

Extra articles demonstrating how Tracing can be added to Import/Export routines and Web Services programmatically:

- Illustrated guide to tracing Imports and Exports

<https://www.sagecity.com/gb/sage-x3-uk/b/sage-x3-uk-support-insights/posts/tracing-imports-and-exports>

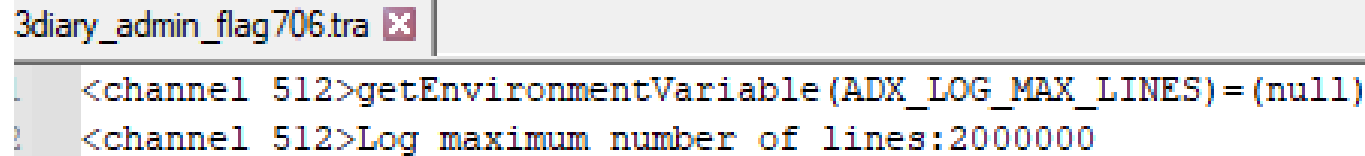
- Illustrated guide to tracing Web Services

<https://www.sagecity.com/gb/sage-x3-uk/b/sage-x3-uk-support-insights/posts/illustrated-guide-to-tracing-web-services>

Miscellaneous

Trace File size limit

An interesting point, which may come in handy, can be noted when Mode 512 is used – it reveals a configuration setting which can be used to increase the number of lines recorded in a Trace file :



```
3diary_admin_flag706 tra x  
<channel 512>getEnvironmentVariable (ADX_LOG_MAX_LINES) = (null)  
<channel 512>Log maximum number of lines:2000000
```

HKEY_LOCAL_MACHINE\SOFTWARE\Adonix\X3RUNTIME\<<solution-name>\ADX_LOG_MAX_LINES is a String (REG_SZ) with the maximum number of lines per Trace file – the default is 2000000.

Miscellaneous

Timing/Ticks

When you're using Engine Traces, you may have noticed lines with "tick" on them.

These are actually timing values in milliseconds, so you could do some timing/performance analysis yourselves – as per Example 6 earlier.

As mentioned earlier, having these tick-values available in the Engine Traces does make it possible to drill into the code. Again, Trace Value 7 is a good start.

Questions?

Thank you!

