



SAP Note 2491748 provides a tool that can help view SAP HANA Dump files

2491748 - How-To: Analyzing Runtime Dumps with SAP HANADumpViewer

- It is a python script to be downloaded from SAP Note 2491748

Attachments	
File Name	File Size
hanadumpviewer.pdf	1043 KB
hanadumpviewer.py	23 KB

- It is intended to be executed as <sid>adm on your SAP HANA Server (since then the proper python version is already in your path, installed together with hana)
- It does not need any user in hdbuserstore since it does never connect to hana
- So it can also be executed from your local laptop (e.g. on windows, see later slides)

For more about the SAP HANADumpViewer see SAP Note 2xxxxx9

Monitoring HANADumpViewer – Dot Files



The SAP HANA Dump Viewer creates .dot files out of .trc files:

Dump File (.trc)

```
[STACK_SHORT] Short call stacks and pending exceptions
335413741[thr=40822]: JobWrk0058 is inactive
--
50744[thr=45394]: SqlExecutor at
1: 0x0000000000000000 in <no symbol>+0x0 (<unknown>)
2: 0x000007faaa1855ed6 in AnalyticalAuthorization::SqlA
3: 0x000007faaa0700f35 in ptime::qo_PredGrantInjection:
4: 0x000007faaa0702d3c in ptime::qo_PredGrantInjection:
5: 0x000007faaa06f676a in ptime::qo_Rule::checkAndApply
6: 0x000007faaa030933c in ptime::qo_Normalizer::applyRu
7: 0x000007faaa0309b0e in ptime::qo_Normalizer::rewrite
8: 0x000007faaa030a0f1 in ptime::qo_Normalizer::normali
9: 0x000007faaa028b5a6 in ptime::QueryOptimizer::optimi
```



Dot File (.dot)

```
digraph StackGraph {
  ratio=compress
  rankdir=BT
  nlegend [shape=record,label="{{#T =
process}}",style=filled,fillcolor="
nC0
[shape=record,label="{lft::allocate
lled,fillcolor="#ffffff",fontname=s
nC1
```

The Dot File can then easily be viewed in e.g. <http://www.webgraphviz.com> (see next slide)

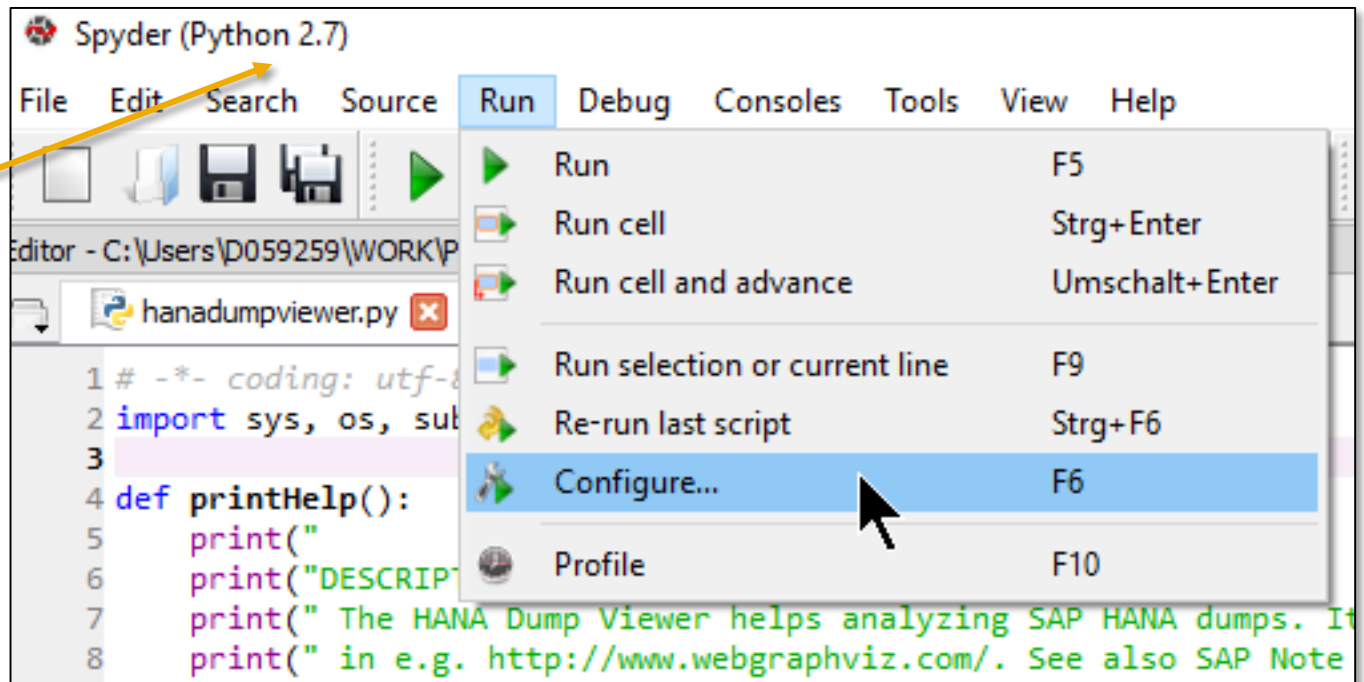
Monitoring HANADumpViewer – On Windows (1/3)



There are uncountable ways of running a python program on Windows
Here is one example of how HANADumpViewer can run from Windows

Download and install Python(x,y) with e.g. these instructions , then open handumpviewer.py in Spyder (that comes with Python(x,y)) and Run → Configure

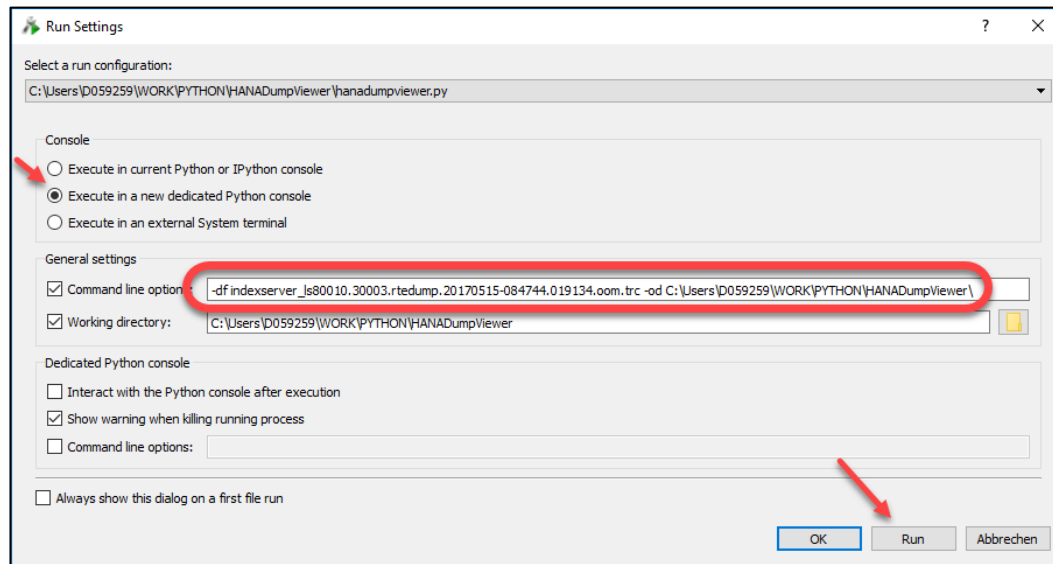
Make sure you get a 2.7.x version as this is still the default version on your HANA server and this is the version HANADumpViewer was developed for



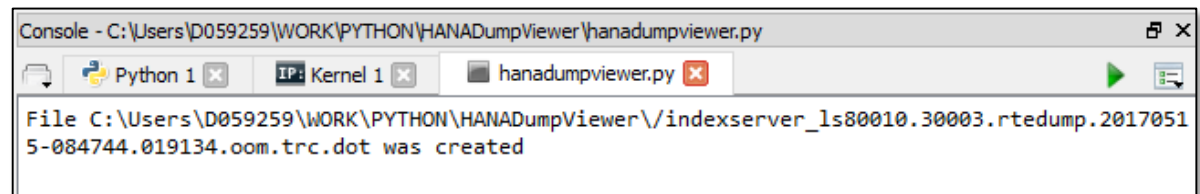
Monitoring HANADumpViewer – On Windows (2/3)



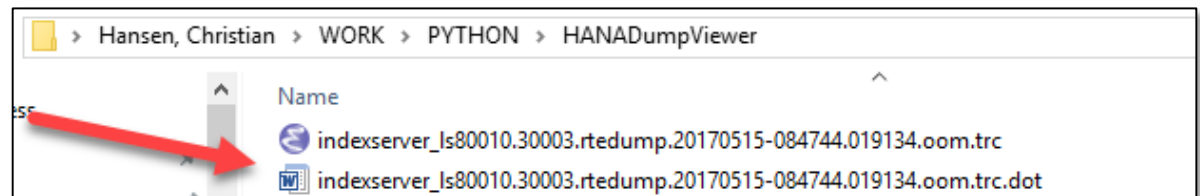
Specify the input file with the `-df` flag and output folder with the `-od` flag



**In the python console
we see that the .dot file
was created**



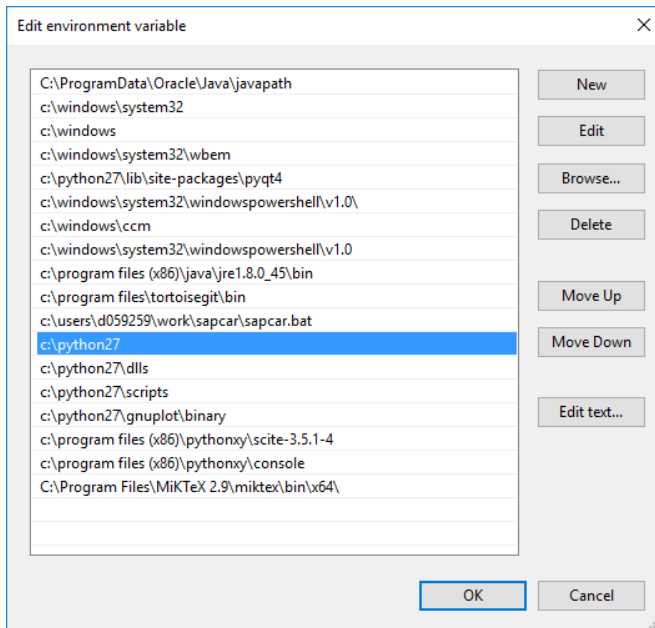
**And we find it in
Windows Explorer**



Monitoring HANADumpViewer – On Windows (3/3)



Make sure the python binaries can be found in your path (Control Panel → System → Advanced Settings → Environmental Variables → Path → Edit)



Then HANADumpViewer can be executed from your CMD prompt:

```
C:\Users\D059259\WORK\PYTHON\HANADumpViewer>python hanadumpviewer.py -df rtedump.txt -od C:\Users\D059259\WORK\PYTHON\HANADumpViewer\  
File C:\Users\D059259\WORK\PYTHON\HANADumpViewer\rtedump.txt.dot was created
```

Monitoring HANADumpViewer – Graph Viz



The Dot Files can easily be viewed in e.g. <http://www.webgraphviz.com>

← → ↻ 🏠 webgraphviz.com

WebGraphviz is [Graphviz](#) in the Browser

Enter your graphviz data into the Text Area:
Your Graphviz data is private and never harvested

Sample 1 Sample 2 Sample 3 Sample 4 Sample 5

```
digraph StackGraph {
    ratio=compress
    rankdir=BT
    nlegend [shape=record,label="{#T = Number threads executing the stack process}"]
    nC8 [shape=record,label="{Thread ID: 29253\nThread Type: JobWrk0046}"]
    nC1 [shape=record,label="{syscall+0x15\n#T=14}"]
    nC2 [shape=record,label="{Synchronization::Semaphore::timedWait\n#T=4}"]
    nC3 [shape=record,label="{MemoryManager::GlobalMemoryHandler::pmWait\n#T=4}"]
    nC4 [shape=record,label="{MemoryManager::GlobalMemoryHandler::pmAcquireIPMMLock\n#T=4}"]
    nC5 [shape=record,label="{MemoryManager::GlobalMemoryHandler::pmAcquireIPMMLockOrSuppo}"]
    nC6 [shape=record,label="{MemoryManager::GlobalMemoryHandler::pmAcquireIPMMLockOrSuppo}"]
    nC7 [shape=record,label="{MemoryManager::GlobalMemoryHandler::provideMemoryAndReturnSi}"]
}
```

Generate Graph!

```
graph BT
    A["Thread ID: 7133  
Thread Type: WorkerThread (StatisticsServer)"]
    B["MemoryManager::SmallBlockAllocator::processDelayedDeallocs  
#T=1"]
    C["MemoryManager::MemoryPool::collectGarbage  
#T=1"]
    C --> B
    B --> A
```

Monitoring

HANADumpViewer – Dump Files



HANA Dump Viewer takes either the full paths of dump files or finds them in the folder defined by the alias “cdtrace”

Flag	Unit	Details	Explanation	Default
-df		dump file names	List of dumpfile path names, separated by a comma only	“
-nd		number indexserver dumpfiles	Number of indexserver .trc files that HANA Dump Viewer will try to collect from “cdtrace” and create .dot files for	0
-dt		dump type	HANA Dump Viewer will only look for indexserver .trc files of this type, i.e. this string has to be part of the dump file name, e.g. rte, oom, ...	“

Example:

Here 2 indexserver .trc files are specified and for each a .dot file is created:

```
mo-fc8d991e0:/tmp/HANADumpViewer> python hanadumpviewer.py -df indexserver_ls80010.oom.trc,indexserver_mo.rtedump.trc
File /tmp/hanadumpviewer_output/indexserver_ls80010.oom.trc.dot was created
File /tmp/hanadumpviewer_output/indexserver_mo.rtedump.trc.dot was created
```

Example:

Here 3 crashdump indexserver .trc files are collected from “cdtrace” and for each a .dot file is created:

```
mo-fc8d991e0:/tmp/HANADumpViewer> python hanadumpviewer.py -nd 3 -dt crashdump
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170227-042318.006632.trc.dot was created
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170321-114146.044877.trc.dot was created
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170408-102430.001820.trc.dot was created
```

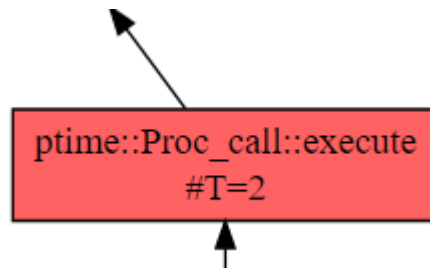
Monitoring HANADumpViewer – Dot Files



The .dot files can be viewed in
e.g. <http://www.webgraphviz.com>

Each stack component is shown
in its own box:

The red color
scale and the
#T label shows
how many threads that executed
that particular stack process, as is
also explained in the legend box:



#T = Number threads executing the stack process



Monitoring HANADumpViewer – Plot Threads



One can also specify to add the thread definitions in the .dot files

Flag	Unit	Details	Explanation	Default
-pt	true/ false	plot threads	Show the thread definitions as additional components	false

Example:

Here .dot files with thread definitions are created from 3 crashdump indexserver .trc files from “cdtrace”:

```
mo-fc8d991e0:/tmp/HANADumpViewer> python hanadumpviewer.py -nd 3 -dt crashdump -pt true
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170227-042318.006632.trc.dot was created
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170321-114146.044877.trc.dot was created
File /tmp/hanadumpviewer_output/indexserver_mo-fc8d991e0.30003.crashdump.20170408-102430.001820.trc.dot was created
```

Monitoring HANADumpViewer – Threads



There are two type of threads;
Normal threads and **Exception threads**

Normal threads are shown in cyan colored boxes and have a thread id and thread type

Thread ID: 6992
Thread Type: SqlExecutor

Exception threads are shown in orange colored boxes, their id are simply their order number in the .trc file and they have a reason for failure:

Exception ID: 0
Reason: Allocation failed ; size=600; name=DPAPI_DataEncApi; type=pool; inuse_count=4; allocated_size=2896

The Legend is showing number of threads:

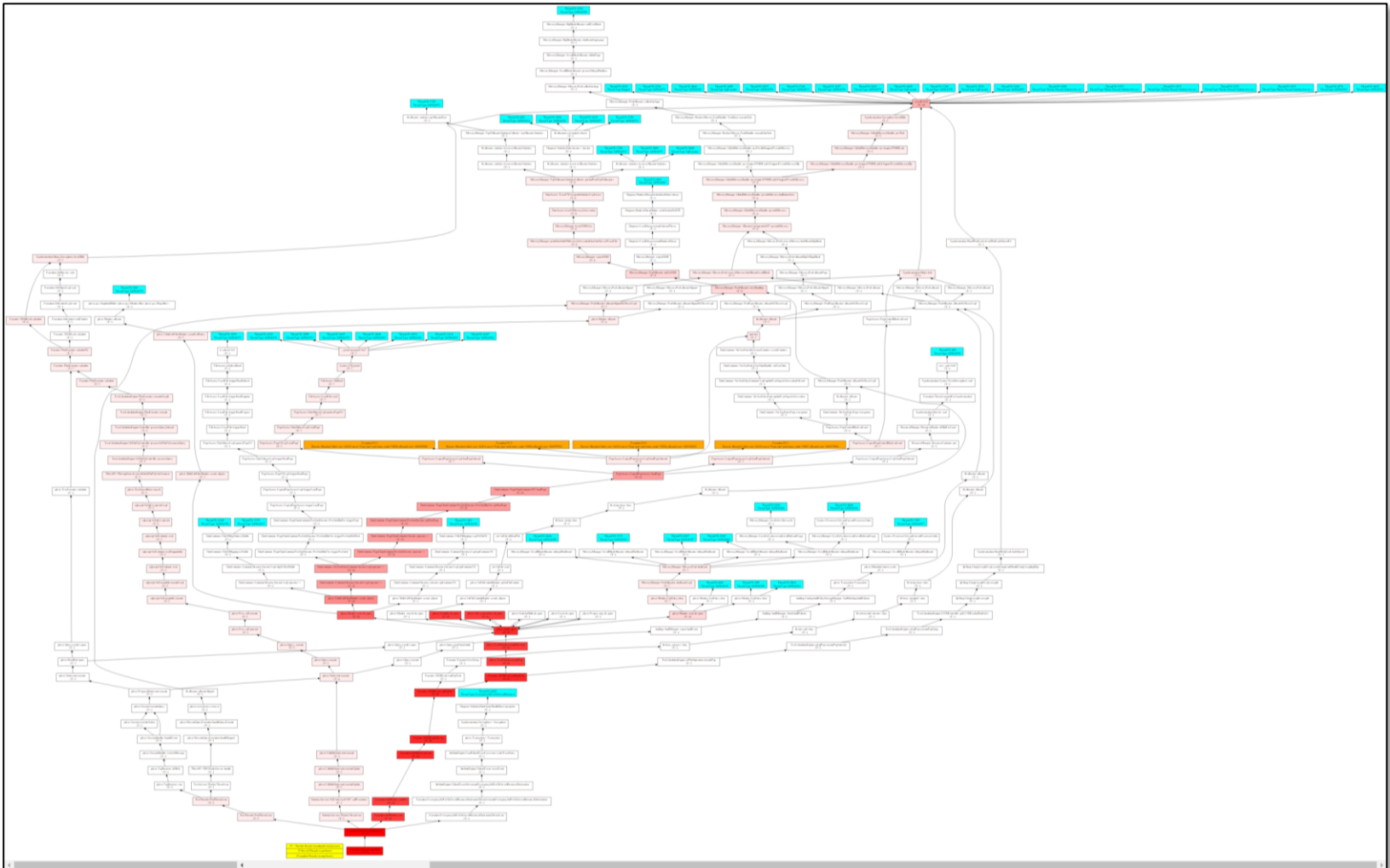
#T = Number threads executing the stack process
2 Normal Threads (cyan boxes)
1 Exception Threads (orange boxes)



Monitoring HANADumpViewer – Large Results



The result from the HANA Dump Viewer can be rather extensive:



Monitoring HANADumpViewer – Wait Graph



The section [INDEXMANAGER_WAITGRAPH] consist of a .dot graph

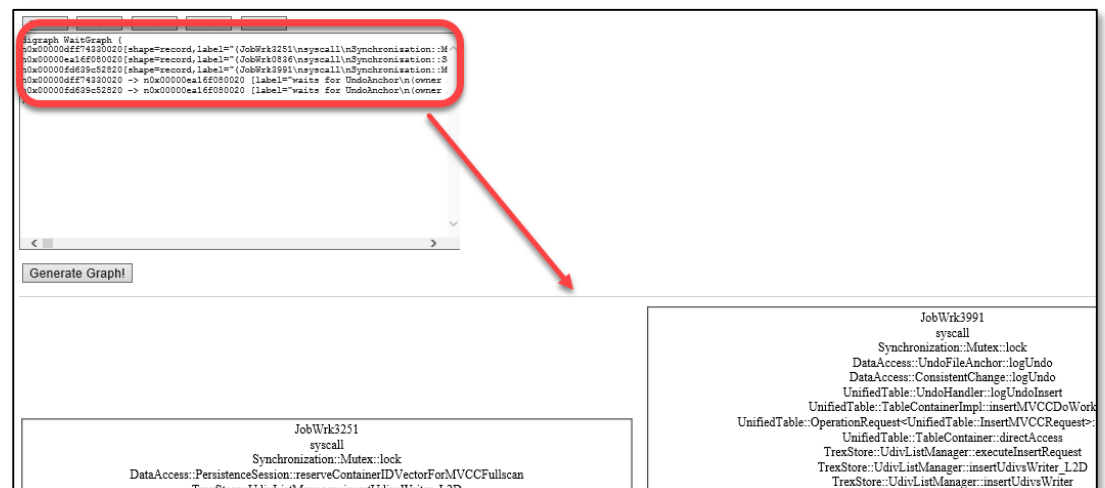
With the flag **-mw** this .dot graph is saved in a dedicated file:

Flag	Unit	Details	Explanation	Default
-mw	true/ false	make wait graph	Creates a file with the indexserver wait graph which is simply the content of the section [INDEXSERVER_WAITGRAPH]	false

Example:

```
oqladm@ls80010:/tmp/HANADumpViewer> rm -r ../hanadumpviewer_output/
oqladm@ls80010:/tmp/HANADumpViewer> python hanadumpviewer.py -md false -mw true -df rtedump.txt
oqladm@ls80010:/tmp/HANADumpViewer> ls ../hanadumpviewer_output/
indexmanager_waitgraph_rtedump_txt.dot
```

The content of this file
can be viewed with e.g.
webgraphviz.com:



Monitoring HANADumpViewer – Views for Excel



All system views, e.g. the M_ views, are dumped in the dump files

With the flag **-mv** these files are saved in a dedicated .csv files:

Flag	Unit	Details	Explanation	Default
-mv	true/ false	make view files	Creates a .csv file for for each system view, e.g. the M_ views, in a VIEW folder in the output directory	false

Example:

```
oqladm@ls80010:/tmp/HANADumpViewer> rm -r ../hanadumpviewer_output/  
oqladm@ls80010:/tmp/HANADumpViewer> python hanadumpviewer.py -md false -mv true -df rtedump.txt  
oqladm@ls80010:/tmp/HANADumpViewer> ls ../hanadumpviewer_output/VIEWS_rtedump_txt/  
DATABASE_STATISTICS.csv          M_DEV_REP_LOGSENDER.csv          M_LAZY_  
EMERGENCY_JOBS.csv              M_DEV_REP_TRANSACTIONS.csv       M_LOG_E  
M_BLOCKED_TRANSACTIONS .csv      M_DEV_REP_TRANSLOGREPLAYER.csv  M_LOG_F
```

These files can be opened with e.g. Excel:



CONNECTION_ID	OBJECT_HANDLE	TRANSACTION_ID	STAR
-350604	1,59575E+13	823	1494
-350724	1,67648E+13	559	1494
-350232	1,58932E+13	745	1494
-350049	1,64568E+13	687	1494
-350384	1,62905E+13	564	1494