Efficient extraction of messages from an IM message log using MS Excel with ABB DataDirect[1] (or any other Oracle compliant tool, e.g. MS Access)

[1] An EXCEL_DA license seat is required in the CLS to use Excel for this method of extraction.

1. Gain database access to the `ops$ocshis` account owning the tables we are about to retrieve data from The password was set by the installing person during database creation. To attempt login, type the following command in a Command Prompt (if login is successful, the following text is output):

```
C:\> sqlplus ops$ocshis/password@localhost

SQL*Plus: Release 11.2.0.2.0 Production on Wed Dec 4 16:07:32 2019

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Release 11.2.0.2.0 - 64bit Production
```

   If the password cannot be found, it can be reset by any user member of the local **ORA_DBA** Microsoft Windows security group in the IM server and the following commands ("as sysdba" removes the need for entering a password):

```
C:\> sqlplus /@localhost as sysdba

SQL> alter user ops$ocshis identified by newpassword;
User altered.
```

   Going onwards; commands to be given to Oracle are preceded by `SQL>`.

2. Find the name of the message log table (NUMxxxx) by executing the following query:

```
SQL> select owner, table_name from dba_tables where table_name like 'MSG%';

OWNER                          TABLE_NAME
------------------------------ ------------------------------
OPS$OCSHIS                     MSGVENDORS_RST
OPS$OCSHIS                     MSGVENDORS
OPS$OCSHIS                     MSGTYPEID_FORM
OPS$OCSHIS                     MSGCOUNT272
OPS$OCSHIS                     MSGCOUNT229
OPS$OCSHIS                     MSGATTRS272
OPS$OCSHIS                     MSGATTRS229
OPS$OCSHIS                     MSG272
OPS$OCSHIS                     MSG229
```

   This IM has two message logs, MSG229 and MSG272. In fact, one of them is Batch PDL messages and the other OPC messages. The content can be queried with "`SELECT * from ops$ocshis.msgXXX`". We aim for audit messages which should only be available in one of them.

3. Identify category IDs

   An IM can log many event categories (process, system, audit, etc.) - this guide will make use of two of them: *AuditEvent_AspectDirectory* and *AuditEvent_OperatorAction*. The category IDs need to be found.

```
SQL> select distinct category_name, category from ops$ocshis.msgvendors;

CATEGORY_NAME                                              CATEGORY
…
AuditEvent_OperatorAction                                        74
AuditEvent_AspectDirectory                                       56
…
```

4. Create some custom *views* to ease access to Audit Messages. Please notice the use of the table names (MSGxxx) and category IDs (xx) we established earlier.

```
SQL> create view audit_operatoraction (localtime, object, message, username)
  as select a.localtime, a. source, a.message, b.str_value
  from ops$ocshis.msg272 a, ops$ocshis.msgattrs272 b
  where a.idx = b.idx and a.category = 74 and b.attr_id in ( select attr_id
  from ops$ocshis.msgvendors where attr_name = 'UserFullName' );
View created.

SQL> create view audit_aspectdirectory (localtime, object, message,username)
  as select a.localtime, a. source, a.message, b.str_value
  from ops$ocshis.msg272 a, ops$ocshis.msgattrs272 b
  where a.idx = b.idx and a.category = 56 and b.attr_id in ( select attr_id
  from ops$ocshis.msgvendors where attr_name = 'UserFullName' );
View created.
```

5. Try retrieving data from the view, e.g.

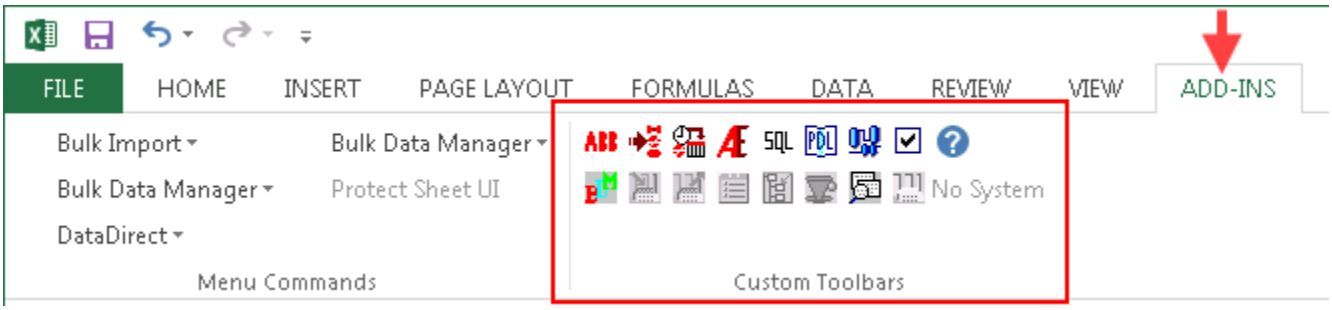
```
SQL> select * from audit_operatoraction;

LOCALTIME           OBJECT      MESSAGE               USERNAME
------------------- ----------- --------------------- --------
2019-10-07 10:14:21 AlarmTest01 Signal True -> False admin

209 rows selected.
```

   The output format has been slightly altered in the above example. Use statements like in the below to adjust output (NOTE: this will only affect the current SQL*Plus session and is not needed for any later use).

```
set linesize 132
set wrap off
column object   format a20
column message  format a25
column username format a15
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS';
```

6. This concludes the preparations needed in Oracle. Now let us move on to Excel and DataDirect

7. Start Excel and ensure the DataDirect add-in is loaded (C:\ProgramData\ABB\IM\DataDirect\Bin\ABBDataDirect.xla)

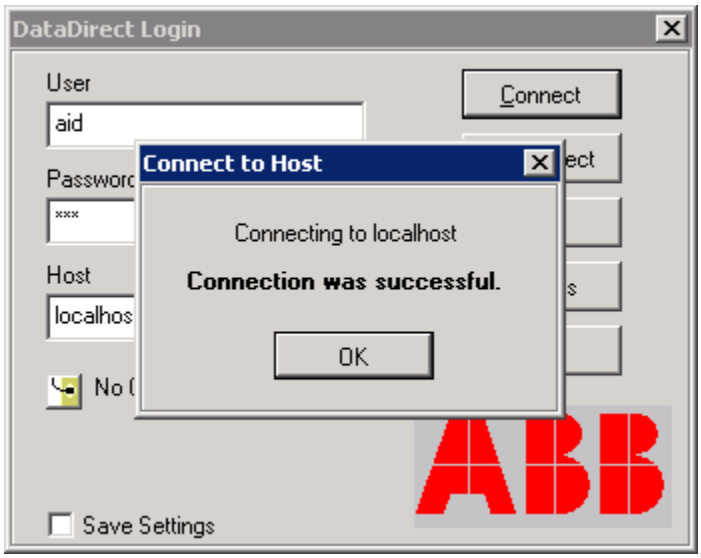


8. The add-in must make a login to the IM server to be able to run SQL queries.

   This requires

   - An available EXCEL_DA license seat in the CLS
   - The *ABB Data Service Supervision* service in Windows must be running under the 800xA service account.
   - The ADO Data Provider must be running (see step 4.4 in the IM Configuration Assistant)
   - If Excel is running from a remote location, the network path must be open and not blocked by any firewall (TCP ports 19014 – 19017).
   - A display services user must be active (refer to “Managing Users for Display Services” in the IM Configuration User’s Guide, e.g. pages 596-601 of 3BUF001092-510 C).

     This example users the demo user “aid” (having the same password).



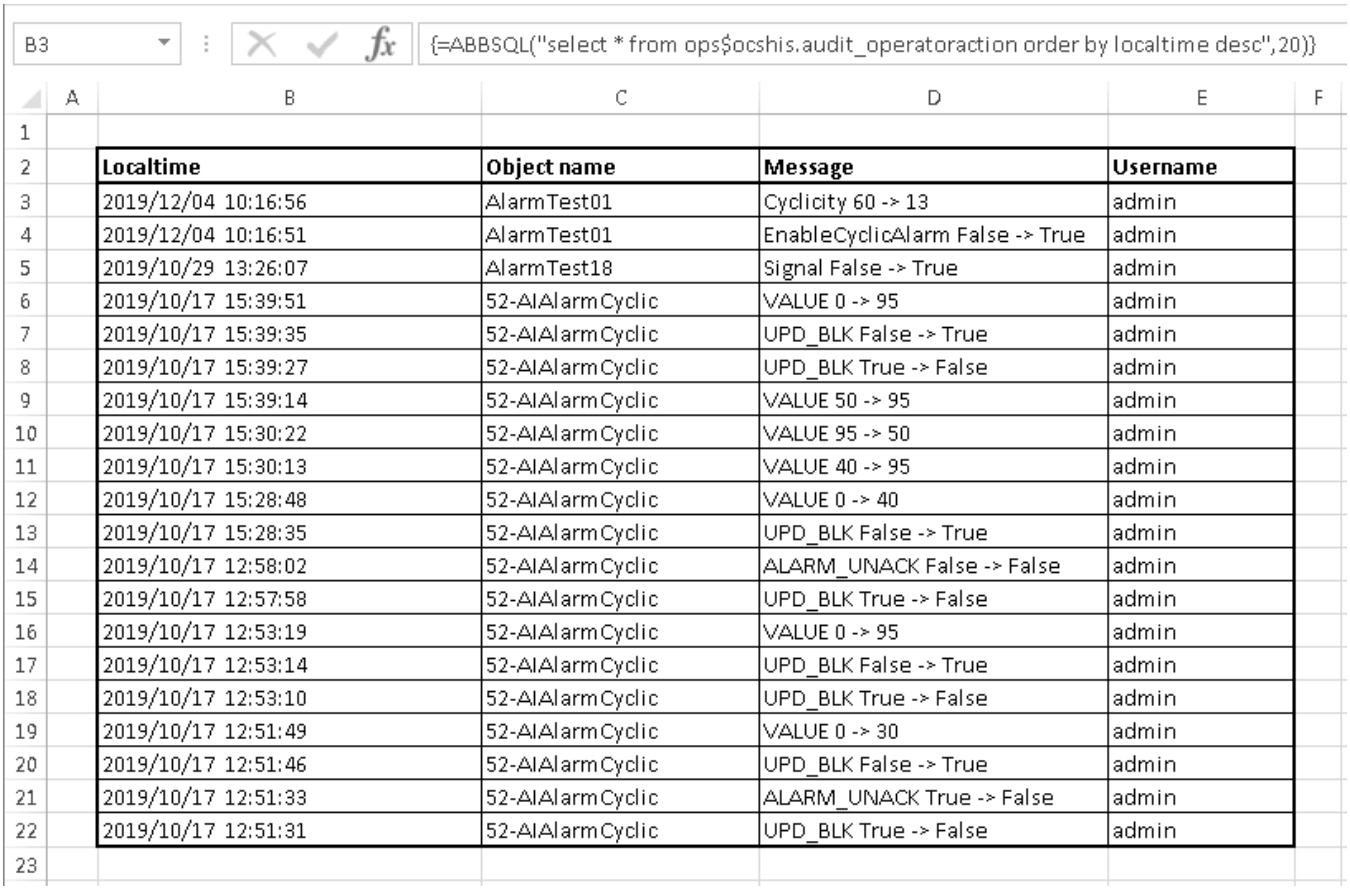
9. The ABBSQL() function will return the result of an SQL statement. Do not terminate the command with semicolon (;) as you usually do in SQL, e.g. SQL*Plus.

```
select * from ops$ocshis.audit_operatoraction order by localtime desc
```

Note: since multiple rows can be returned from a single SQL statement, the spreadsheet must be prepared, and the formula must be embraced by curly brackets.

```
{=ABBSQL("select * from ops$ocshis.audit_operatoraction order by localtime desc",20)}
```

Mark the cells that the output shall be put in (e.g. an array of 4 x 20), enter the formula, ensure the output does not overflow by giving the maximum number of rows as the second argument (here I set "20"). Press & hold the CTRL-SHIFT keys then hit ENTER to apply the formula content on to the selected area



Localtime	Object name	Message	Username
2019/12/04 10:16:56	AlarmTest01	Cyclicity 60 -> 13	admin
2019/12/04 10:16:51	AlarmTest01	EnableCyclicAlarm False -> True	admin
2019/10/29 13:26:07	AlarmTest18	Signal False -> True	admin
2019/10/17 15:39:51	52-AIAlarmCyclic	VALUE 0 -> 95	admin
2019/10/17 15:39:35	52-AIAlarmCyclic	UPD_BLK False -> True	admin
2019/10/17 15:39:27	52-AIAlarmCyclic	UPD_BLK True -> False	admin
2019/10/17 15:39:14	52-AIAlarmCyclic	VALUE 50 -> 95	admin
2019/10/17 15:30:22	52-AIAlarmCyclic	VALUE 95 -> 50	admin
2019/10/17 15:30:13	52-AIAlarmCyclic	VALUE 40 -> 95	admin
2019/10/17 15:28:48	52-AIAlarmCyclic	VALUE 0 -> 40	admin
2019/10/17 15:28:35	52-AIAlarmCyclic	UPD_BLK False -> True	admin
2019/10/17 12:58:02	52-AIAlarmCyclic	ALARM_UNACK False -> False	admin
2019/10/17 12:57:58	52-AIAlarmCyclic	UPD_BLK True -> False	admin
2019/10/17 12:53:19	52-AIAlarmCyclic	VALUE 0 -> 95	admin
2019/10/17 12:53:14	52-AIAlarmCyclic	UPD_BLK False -> True	admin
2019/10/17 12:53:10	52-AIAlarmCyclic	UPD_BLK True -> False	admin
2019/10/17 12:51:49	52-AIAlarmCyclic	VALUE 0 -> 30	admin
2019/10/17 12:51:46	52-AIAlarmCyclic	UPD_BLK False -> True	admin
2019/10/17 12:51:33	52-AIAlarmCyclic	ALARM_UNACK True -> False	admin
2019/10/17 12:51:31	52-AIAlarmCyclic	UPD_BLK True -> False	admin

The SQL statement can be refreshed by pressing CTRL-ALT-SHIFT-F9 (=Excel Calculate All)

The formula can be made dynamic, e.g. to fetch events at a certain point in time, indicated by a cell in Excel.

E.g. in the below more advanced example, the date put in B3 will be converted into SQL by the formula in B2. The formula in B2 is shown in B1. The example uses Excel's **TEXT()** function to convert B3's content into a known date format which is sent in to ABBSQL subsequently converted into Oracle time by **TO_DATE()**.

```
=CONCATENATE("select * from ops$ocshis.audit_operatoraction where localtime
<= TO_DATE(","'",TEXT(B3,"yyyy-mm-dd hh:mm:ss"),"'","","'YYYY-MM-DD
HH24:MI:SS","')"," order by localtime desc")
```

Which becomes transformed into:

```
select * from ops$ocshis.audit_operatoraction where localtime <=
TO_DATE('2019-11-01 00:00:00','YYYY-MM-DD HH24:MI:SS') order by localtime
desc
```

And finally sent into ABBSQL() with:

```
=ABBSQL(B1,20)
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Formula:	=CONCATENATE("select * from ops$ocshis.audit_operatoraction where localtime <= TO_DATE(","'",TEXT(B3,"yyyy-mm-dd hh:mm:ss"),"'","","'YYYY-MM-DD HH24:MI:SS","')"," order by localtime desc")												
2	Raw SQL:	select * from ops$ocshis.audit_operatoraction where localtime <= TO_DATE('2019-11-01 00:00:00','YYYY-MM-DD HH24:MI:SS') order by localtime desc												
3	Starttim	2019-11-01 00:00:00												
4														
5		**Localtime**	**Object name**	**Message**	**Username**									
6		2019/10/29 13:26:07	AlarmTest18	Signal False -> True	admin									
7		2019/10/17 15:39:51	52-AIAlarmCyclic	VALUE 0 -> 95	admin									
8		2019/10/17 15:39:35	52-AIAlarmCyclic	UPD_BLK False -> True	admin									
9		2019/10/17 15:39:27	52-AIAlarmCyclic	UPD_BLK True -> False	admin									
10		2019/10/17 15:39:14	52-AIAlarmCyclic	VALUE 50 -> 95	admin									
11		2019/10/17 15:30:22	52-AIAlarmCyclic	VALUE 95 -> 50	admin									
12		2019/10/17 15:30:13	52-AIAlarmCyclic	VALUE 40 -> 95	admin									
13		2019/10/17 15:28:48	52-AIAlarmCyclic	VALUE 0 -> 40	admin									
14		2019/10/17 15:28:35	52-AIAlarmCyclic	UPD_BLK False -> True	admin									
15		2019/10/17 12:58:02	52-AIAlarmCyclic	ALARM_UNACK False -> False	admin									
16		2019/10/17 12:57:58	52-AIAlarmCyclic	UPD_BLK True -> False	admin									
17		2019/10/17 12:53:19	52-AIAlarmCyclic	VALUE 0 -> 95	admin									
18		2019/10/17 12:53:14	52-AIAlarmCyclic	UPD_BLK False -> True	admin									
19		2019/10/17 12:53:10	52-AIAlarmCyclic	UPD_BLK True -> False	admin									
20		2019/10/17 12:51:49	52-AIAlarmCyclic	VALUE 0 -> 30	admin									
21		2019/10/17 12:51:46	52-AIAlarmCyclic	UPD_BLK False -> True	admin									
22		2019/10/17 12:51:33	52-AIAlarmCyclic	ALARM_UNACK True -> False	admin									
23		2019/10/17 12:51:31	52-AIAlarmCyclic	UPD_BLK True -> False	admin									
24		2019/10/17 12:51:26	52-AIAlarmCyclic	VALUE 45 -> 95	admin									
25		2019/10/17 12:51:17	52-AIAlarmCyclic	VALUE 0 -> 45	admin									
26														

The user enters an arbitrary date into B3 and performs a *Calculate All* to retrieve the data

10. If the user prefers to work from e.g. MS Access, the path should be open as soon as a suitable driver for Oracle is found. The IM should have the driver pre-loaded locally, you only need to install MS Access there.

11. More information about how to extract data from IM can be found in the IM Data Access and Reports guide:
System 800xA Information Management 5.1 Data Access and Reports, 3BUF001094-510
System 800xA Information Management 6.0 Data Access and Reports, 3BUF001094-600
System 800xA Information Management 6.1 Data Access and Reports, 3BUF001094-610

Stefan Strömqvist
SE L3 Operation Support Team