SEK Audit extension for SAP Sybase ASE

# Background

Some time ago SEK tried to do Sybase Auditing via the built in Auditing system provided by SAP/Sybase. However during that attempt to much resources was used up by the Sybase Auditing subsystem and the project was “not successful”...

This time a new attempt will be taken with a slightly different approach!

* Mainly using the Monitoring tables “monSysSQLText” and “monSysStatement” to gather information about all SQL Statements that has been executed.  
  The “mon” tables are “in memory” tables, which is not persisted to disk, hence the more lightweight approach!
* SAP Sybase Auditing subsystem will be used as a complement to the above monitoring tables to collect things that we might not be able to capture from the monitoring tables.  
  Like: User Success-Logins, Failed-Logins, Logout, Server restarts, Server Reconfigurations...

**The above is to minimize impact on performance in the current Production System.**

Filtering capability on “Program Name” and “SQL Text” has also been implemented, which can not be done with the Sybase default implementation.

Those filtering capability is because we don’t want to store *everything* that happens... and possibly “flood” the Splunk system, which is the “endpoint” for storing Auditing data.

# Overall Architecture

A picture containing text, diagram, map, plan

Description automatically generated

1. Collect information from Sybase via two different collectors (described in detail later)
   1. SybStmntCollector: From in-memory monitoring tables
   2. SybAuditCollector: From Sybase Auditing subsystem

Both collectors enrich its information from the SysProcessesCache to get “Program Name” and other fields that may be used for filtering...

Filters for “Login Name” and “Program Name” will be applied before adding information to a “container” that is written into a “Queue Handler”, then step 2 will continue processing the information.

**Note**: If we filter on “Program Name” and we can’t (for various reasons) not find the *SPID* in the “SybSysProcessesCache”. The entry will be sent anyway (better to send than to remove or lose-data).

1. Queue Handler wakes up on new entries, and simply writes the “container” to the **registered** writers... each writer will also have a Queue, which the information is written into.
2. Now the writer wakes up on new entries in the local queue and “do it’s work”  
   For example the “WriterToSplunk” does the following:
   1. Format the java object into JSON Message
   2. Then send the information via HTTP request to Splunk

**Note 1**: Any “Writer” can implement a local filter to remove, transform or enrich data before sending it of to its destination.  
**Note 2**: A writer can/will use several threads to send it’s data (this due to a network call to the destination is usually a “slow” operation... if the queue starts to build up, warning messages will be written to the application error log, so we can investigate why and take appropriate actions.

## SybStmntCollector

This module pulls information from in-memory monitoring tables (monSysSQLText, monSysStatement). Using those tables has a much less impact on the overall performance impact on the DBMS, compared to using the normal Auditing subsystem provided by SAP Sybase.

Information from the monitoring tables are also “put together” in a more comprehensive message packet (than just reading from the tables)... and we will also get some extra information, like: Statement “Execution time”, “Rows Affected”, “Error Information”, etc.

## SybAuditCollector

This module gets information from the SAP Sybase Auditing database (sybsecurity), and the table ‘sysaudits\_01’. This is done by calling the stored procedure ‘audit\_collector’.

Overall logic of the stored procedure (which is self maintainable)

* Get various configuration from table ‘audit\_collector\_config’
* **Transfer records** from ‘sysaudits\_01’ to SybAuditCollector (since last “last\_poll\_ts”)
* Update config “last\_poll\_ts”
* Check if data or log space is below threshold; if so *@do\_truncate = 1*
* Check if it’s time to truncate ‘sysaudits\_01’ (default every 6 hour); if so *@do\_truncate = 1*
* If @do\_truncate
  + **Transfer records** from ‘sysaudits\_01’ to SybAuditCollector (since bullet two)
  + Truncate table ‘sysaudits\_01’
  + Update config “last\_truncate\_ts”

It also merges several audit records into 1 record (a record from sysaudits\_01 can only hold 255 characters, so if there are several records they needs to be “merges” into 1 record)

It also “decodes” different message structures in the column “extrainfo” into individual fields.  
And for Configuration History it also “decodes” them into individual fields.  
This so it should be easier to read!

**Note:** Configuration of how to configure the Sybase Auditing system and what information that is stored in there, can be found in the Sybase manuals “Security Administration Guide”  
<https://help.sap.com/docs/SAP_ASE/2705a3b1e3df4514ab089cfedf87750d/a94a794cbc2b1014b9fae018f9300b23.html>

## WriterToSplunk

This module sends information in the JSON format to Splunk.  
Splunk can then do additional filtering, creating alarms and/or further enhance the logging data

## WriterToXxx

For future compatibility you can later create new writers for SEK modules that has special needs.

**One example** of this could be: a module for SIP, which may want the information in “some other way”... SIP was added as a “add-on” to the previous logging project.

## Filter functionality

The above modules also have some filter capabilities, this so we can exclude information that comes from specific logins, “Program Names” or even filter on the “SQL Text” itself.  
*(See more about how to do this in the “Configuration File” section.)*

**Filter on “Program Name” or “SQL Text” is not possible with the SAP Sybase Auditing System.**

# Monitoring the SEK ASE Auditing System

No alarm module has been built into “SEK ASE Auditor”, so the error log should be monitored for “various” errors, like:

* If Queue Sizes goes above default thresholds
* If monitoring tables configuration is set to low (we are missing SQL Statements due to few slots in the “ring buffer”, which is detected by adding a known SQL Statement and detect that it’s picked up by the Statement Collector)
* *And other various error messages.*

## Basic monitoring via AseTune

AseTune can monitor that some application names are logged in ...  
This is done with config: CmProcessActivity.alarm.system.mandatory.contentList.forColName.Application = SekAseAuditor  
And will send the alarm: AlarmEventMissingMandatoryContent

# Automatic startup of: SEK ASE Auditor

A script for ‘systemd’ is shipped with the package that can be used for automatic start after reboot or manual stop/start via the ‘systemctl’ command.

# Configuration File

See last in this documentation

# From SAP Sybase ASE Manuals (Auditing Subsystem):

A principal element of a secure system is accountability. One way to ensure accountability is to audit events on the system. Many events that occur in SAP ASE can be recorded.

Auditing is an important part of security in a database management system. An audit trail can be used to detect penetration of the system and misuse of resources. By examining the audit trail, a system security officer can inspect patterns of access to objects in databases and can monitor the activity of specific users. Audit records are traceable to specific users, which may act as a deterrent to users who are misusing the system.

Each audit record can log the nature of the event, the date and time, the user responsible for it, and the success or failure of the event. Among the events that can be audited are log ins and log outs, server starts, use of data access commands, attempts to access particular objects, and a particular user’s actions. The audit trail, or log of audit records, allows the system security officer to reconstruct events that occurred on the system and evaluate their impact.

The system security officer is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a system security officer, you can establish auditing events such as:

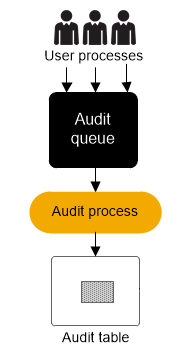
* Server-wide, security-relevant events
* Creating, deleting, and modifying database objects
* All actions by a particular user or all actions by users with a particular role active
* Granting or revoking database access
* Importing or exporting data
* Log ins and log outs

# Single Audit table:

At SEK we use a “single audit table”, which means that there are a small window where (a minimal amount of) audit records might get lost. This model was chosen due to simplicity.

Below is how the auditing system works “at a high level”

* Any login executes something in the DBMS



* If the specific “event” that the user did is set to   
  be recorded by the DBMS auditing meta data  
  The event is inserted in the “in memory” audit queue
* The audit queue is the persisted into the audit table  
  sysaudits\_01 where any users can read the entry.

**Note:** The rule of what’s to be recorded by the audit   
system is specified by SEK Business or Regulation Rules.

From here on we need to transfer the audit data into any   
Company (SEK) specific subsystem that will make something  
useful with the audit information.

At SEK this will be Splunk and possibly other systems.

# SAP Sybase ASE Documentation

More information about Auditing can be found in SAP Sybase ASE -- Security Administration Guide: <https://help.sap.com/viewer/2705a3b1e3df4514ab089cfedf87750d/16.0.3.4/en-US/a94a794cbc2b1014b9fae018f9300b23.html>

# SEK current Audit Settings

The below is the current setting installed by default

* **"anything" that has with security is logged, which is a bunch of things:**
  + Audit table access and changes;
  + server start/stop;
  + create/drop/alter login/login-profile/role;
  + online database;
  + set role/user;
  + sp\_configure;
  + sp\_passwordpolicy;
  + sp\_ssladmin;
  + x number of builtin functions(authmech, show\_cached\_plan\_in\_xml, show\_cached\_text, show\_cached\_text\_long, show\_condensed\_text, show\_plan, valid\_user, workload\_metric)

exec sp\_audit 'security', 'all', 'all', 'on'

* **everything the "sa\_role" is doing (not enabled in this project)**  
  ~~exec sp\_audit 'all', 'sa\_role', 'all', 'on'~~
* **everything the "sso\_role" is doing *(can potentially be disabled...)***  
  exec sp\_audit 'all', 'sso\_role', 'all', 'on'
* **everything the "oper\_role" is doing *(can potentially be disabled...)***  
  exec sp\_audit 'all', 'oper\_role', 'all', 'on'
* **cmd text for 'sa' if that's not already in 'all' for 'sa\_role'... (not enabled in this project)**  
  ~~exec sp\_audit 'cmdtext', 'sa', 'all', 'on'~~
* **All logins to ASE (both Success and Failed)**  
  exec sp\_audit 'login', 'all', 'all', 'on'
* **All Logouts from ASE**   
  exec sp\_audit 'logout', 'all', 'all', 'on'
* **Any DBCC Commands executed by "anyone"**  
  exec sp\_audit 'dbcc', 'all', 'all', 'on'

What is “logged” in the Auditing system is based on the above information.  
And for each of the setting a bunch of “events” are logged, which can be found here: <https://help.sap.com/viewer/2705a3b1e3df4514ab089cfedf87750d/16.0.3.4/en-US/b59debd2ec7346fdb3beb4db9d9ae5c3.html>

## Descriptions of DBMS System defined roles:

* sa\_role Manage and maintain Adaptive Server databases and disk storage
* sso\_role Perform security-related tasks
* oper\_role Back up and load databases server-wide

The user ‘sa’ has all the above roles.

# SEK Extension in sybsecurity database

## Table: audit\_collector\_config

This table contains 1 record, and holds SEK configuration: How we interact with the DBMS audit log  
Below is the table definition:

create table audit\_collector\_config

(

*-- when did we last poll data*

last\_poll\_ts datetime not null

*-- truncate the sysaudits\_## after X hours*

,truncate\_after\_x\_hours int not null

*-- Check data size every x minute, and truncate if below data\_size\_free\_mb\_th*

,check\_data\_size\_x\_minute int not null

*-- Check xlog size every x minute, and truncate if below xlog\_size\_free\_mb\_th*

,check\_xlog\_size\_x\_minute int not null

*-- if data usage is below XXX MB free, do truncate*

,th\_data\_size\_free\_mb int not null

*-- if xlog usage is below XXX MB free, do truncate*

,th\_xlog\_size\_free\_mb int not null

*-- last time the sysaudits\_## was truncated*

,last\_truncate\_ts datetime not null

*-- last time the sysaudits\_## was truncated from the threshold actions procedure*

,last\_th\_truncate\_ts datetime null

*-- last time we checked for data size*

,last\_data\_size\_check\_ts datetime not null

*-- last time we checked for xlog size*

,last\_xlog\_size\_check\_ts datetime not null

)

## Procedure: audit\_collector

This is a procedure that hold the internal SEK logic  
See the overall logic in earlier section

# Configuration File

This is a property file

Important properties are marked with yellow

Most property values in the below is using “USE\_DEFAULT”, which means it gets the values from the default specification in the code.

Note: key/values in the properties file may change in the future versions...

##############################################################################

## Configuration file for: SEK ASE Auditor

##############################################################################

##----------------------------------------------------------------------------

## NOTE: To monitor that \*\*this\*\* process is up and running.

## Add the following configuration to AseTune config file

## CmProcessActivity.alarm.system.mandatory.contentList.forColName.Application = SekAseAuditor

##----------------------------------------------------------------------------

##----------------------------------------------------------------------------

## START: DBMS Config

##----------------------------------------------------------------------------

## Name of the Sybase server

## This can be specified in several ways

## \* SERVERNAME - Then hostname, port will be looked up from the interfaces file

## \* HOST:PORT - Hostname and the port number we want to connect to

## \* URL - Full URL, Example: jdbc:sybase:Tds:<hostname>:<port>

## DEFAULT: none - mandatory parameter

#dbms.server = GORAN\_UB1\_DS

#dbms.server = jdbc:sybase:Tds:gorans-ub3.home:1600

dbms.server = gorans-ub3.home:1600

## Username that we want to connect to the DBMS with

## DEFAULT: none - mandatory parameter

dbms.username = audit\_user

## Password for the above username

## This can be done in several ways

## \* Plain text: Just put the clear text password in here (not recomended)

## \* Encrypted 1: Place an encrypted password in this config file.

## \* To encrypt a password use 'sek\_ase\_auditor.sh --encrypt1 thePasswordToEncrypt'

## Then put the output in this file

## \* Encrypted 2: If the property key 'dbms.password' is NOT present in this file

## we will search the file ${HOME}/.passwd.enc for a password

## To add an entry to above file use 'sek\_ase\_auditor.sh --encrypt2 thePasswordToEncrypt'

## Then put the output in the file: ${HOME}/.passwd.enc

## Also make sure that wile isn't readable by anyone else 'chmod 400 ${HOME}/.passwd.enc'

## Or you can also use 'dbxPassword.sh' that comes with the DbxTune package.

## DEFAULT: none - mandatory parameter

#dbms.password = sybase11

dbms.password = encrypted:GNM8Vfo8e88ZBXtxuKSK0Q==

## If we have connections problems when starting up?

## - Should we continue startup of the service and trust that we will be able to connect at a later time?

## - Or should we exit early, so wee see the error strait away? (easier to trouble shoot)

## In early test phases we probably want to se a error strait away, so we can fix the issue!

## In a tested/working system, we probably want to set this to 'false' and let the service start...

## (hence trust that a connection will be available at a later time)

## For example (at host booot-time) if the DBMS is not yet started, we probably want to retry

## the connections later...

## DEFAULT: true -- Exit on first connect attempt

SybCollector.dbms.exit.if.first.connect.fails = false

## If we fail to connect (after a server restart or simular), we want to "reconnect"

## This is how long we should wait, before we "stop" trying...

## DEFAULT: sleepTimeMs = 10000 (10 seconds)

## DEFAULT: maxWaitTimeInSec = 7200 (2 hour)

## DEFAULT: count = 100000 (many times... if we calculate: way above 'maxWaitTimeInSec')

SybCollector.dbms.connect.fail.retry.sleepTimeMs = USE\_DEFAULT

SybCollector.dbms.connect.fail.retry.maxWaitTimeInSec = USE\_DEFAULT

SybCollector.dbms.connect.fail.retry.count = USE\_DEFAULT

##----------------------------------------------------------------------------

## END: DBMS Config

##----------------------------------------------------------------------------

## On shutdown, max wait time in Milliseconds before doing hard shutdown

## DEFAULT 60000 (60 seconds)

ShutdownHandler.maxWaitTime = USE\_DEFAULT

##----------------------------------------------------------------------------

## START: Collectors

##----------------------------------------------------------------------------

## Skip some login names.

## DEFAULT: null -- send all logins

SybCollector.skip.loginName.regex = USE\_DEFAULT

## Skip some program names.

## Example: to skip BOTH all 'AseTune' applications and 'MxApp-1'

## SybCollector.skip.programName.regex = (AseTune.\*|MxApp-1)

## To test your regex, please try them out on any "online" tester like (methos is: matches)

## 'https://www.regexplanet.com/advanced/java/index.html'

## 'https://regex101.com/'

## 'https://www.lddgo.net/en/string/regex' or similar.

## DEFAULT: (AseTune.\*)

#SybCollector.skip.programName.regex = USE\_DEFAULT

SybCollector.skip.programName.regex = (AseTune.\*|MxApp-1)

## When getting/refreshing information from sysprocesses, minimum time beetween refresh

## DEFAULT: 500

SybSysProcessesCache.refreshThresholdInMs = USE\_DEFAULT

## How often should we print statistics to the error log

## DEFAULT: 300 (5 minutes)

#SybCollector.statistics.message.period.seconds = USE\_DEFAULT

SybCollector.statistics.message.period.seconds = 10

##------------------------------------------------------------

## ----- SybAuditCollector -----

##------------------------------------------------------------

## Sleep time between collections.

## DEFAULT: 10

SybAuditCollector.sleepTime.seconds = USE\_DEFAULT

##------------------------------------------------------------

## ----- SybStmntCollector -----

##------------------------------------------------------------

## Filter out (skip) to send some SQL Text

## DEFAULT: null ==> Simply skip this filter/logic

SybStmntCollector.skip.sqlText.regex = USE\_DEFAULT

## Filter in (keep) only SQL Text that matches this RegEx will be sent

## Example to only check for 'INSERT', 'UPDATE' and 'DELETE': (?i)(INSERT\s|UPDATE\s|DELETE\s)

## Explaination: '(?i)' ==> means: use CASE INSENSITIVE

## Explaination: '\s' ==> means: any white spaces

## Explaination: '(str1|srt2)' ==> means: matches any string of: 'str1' or 'str2'

## To test your regex, please try them out on any "online" tester like (methos is: matches)

## 'https://www.regexplanet.com/advanced/java/index.html'

## 'https://regex101.com/'

## 'https://www.lddgo.net/en/string/regex' or similar.

## DEFAULT: null ==> Simply skip this filter/logic

SybStmntCollector.keep.sqlText.regex = USE\_DEFAULT

## Also Collect information from 'monSysStatement' which holds:

## 'RowsAffected', 'ErrorStatus', 'StartTime', 'EndTime', 'Elapsed\_ms' and a bunch

## of other performance counters

## Entries in 'monSysStatement' are visible on individual Statement completion, and there might be many

## entries for each call (if it's a SQL Batch that does many things, or a stored procedure that

## executes alot of SQL Statements)

## If this is enabled: There might be an extra delay (especially on last executed SQL) before it's

## sent to the Writer Queue.

## DEFAULT = true (but right now in development it's false)

#SybStmntCollector.sample.statement.details = USE\_DEFAULT

SybStmntCollector.sample.statement.details = true

## When 'SybStmntCollector.sample.statement.details' is true, we can choose to collect   
## procedure name (if it's a procedure that a statement record originates from)

## We normally would want to do that... But the requires the ASE Login we use has 'sa\_role'  
## or 'db-access, hence: sp\_adduser' if we do not have that

## we will receive ErrorNumber=10351, ErrorText="Server user id 25 is not a valid   
## user in database 'xxxxxx'."

## DEFAULT = false

SybStmntCollector.sample.statement.procName = USE\_DEFAULT

## Sleep time between collections.

## DEFAULT: 1

SybStmntCollector.sleepTime.seconds = USE\_DEFAULT

## Clear values in monSys\* tables before we start to collect.

## DEFAULT = false

#SybStmntCollector.clearMonTablesOnConnect = USE\_DEFAULT

SybStmntCollector.clearMonTablesOnConnect = true

## Print info/debug message on the usage for the 'pidSqlTextManager'

## This can be used to determen if we have any "messages" that yet hasn't been sent for "various reasons"

## DEFAULT = false

SybStmntCollector.statistics.printSpidSqlTextManagerInfo = USE\_DEFAULT

## not important, just for completenes (ASE Configuration, minimum "recomended" values for the Monitoring system)

## Can be used to test how system behaves when not properly configured...

#SybStmntCollector.ase.config.enable\_monitoring = 1

#SybStmntCollector.ase.config.sql\_text\_pipe\_active = 1

#SybStmntCollector.ase.config.sql\_text\_pipe\_max\_messages = 1000

#SybStmntCollector.ase.config.statement\_pipe\_active = 1

#SybStmntCollector.ase.config.statement\_pipe\_max\_messages = 10000

#SybStmntCollector.ase.config.statement\_statistics\_active = 1

#SybStmntCollector.ase.config.per\_object\_statistics\_active = 1

##----------------------------------------------------------------------------

## END: Collectors

##----------------------------------------------------------------------------

##----------------------------------------------------------------------------

## START: Writers

##----------------------------------------------------------------------------

## A Comma separated list of class names that implements: IWriterConsumer.

## DEFAULT: null/none - mandatory parameter

WriterQueueHandler.WriterClass = sek.ase.auditor.wqs.consumers.WriterToSplunk

#WriterQueueHandler.WriterClass = sek.ase.auditor.wqs.consumers.WriterToFile, sek.ase.auditor.wqs.consumers.WriterToSplunk

## Start to write warning messages into the errorlog when the queur size is above this value

## DEFAULT: 3

WriterQueueHandler.warnQueueSizeThresh = USE\_DEFAULT

## When the Queue Hander writes to each writer... print some info

## DEFAULT: false

WriterQueueHandler.writeInfoOnEachConsumeWrite = USE\_DEFAULT;

## How often should we print statistics to the error log

## DEFAULT: 300 (5 minutes)

#WriterQueueHandler.statistics.message.period.seconds = USE\_DEFAULT

WriterQueueHandler.statistics.message.period.seconds = 10

##------------------------------------------------------------

## ----- WriterToSplunk -----

##------------------------------------------------------------

## How many threads should we use when sending data to Splunk.

## DEFAULT: 3

WriterToSplunk.senderThreadCount = USE\_DEFAULT

## What URL should we send data to.

## DEFAULT: null/none - mandatory parameter

## @home

##WriterToSplunk.url = http://splunk-1-gs:8088/services/collector/event

## @SEK-test-env

WriterToSplunk.url = http://sek-splunktest.sek.se:8088/services/collector/event

## Access token to use when sending data to Splunk.

## DEFAULT: null/none - mandatory parameter

## @home

#WriterToSplunk.access.token = Splunk 4bccb251-43f7-413b-8a03-029040d2e330

## @SEK-test-env

WriterToSplunk.access.token = Splunk e3276d6d-1fd2-430b-9cca-9907ce34fa9a

## Should we format the JSON text as "pretty" before sending it to Splunk

## DEFAULT: false

WriterToSplunk.json.prettyPrint = USE\_DEFAULT

## For debugging, if we want to print the JSON message to the stdout   
## (to the console file) before sending it to Splunk.

## DEFAULT: false

WriterToSplunk.json.printBeforeSend = USE\_DEFAULT

## For debugging, if send fails we can specify how many charaters of the JSON Message we want to print

## DEFAULT: 512

WriterToSplunk.json.on.error.print.numOfChars = USE\_DEFAULT

## Max size of the HTTP Connection pool

## DEFAULT: 100

WriterToSplunk.http.conn.pool.maxSize = USE\_DEFAULT

## Max size per URL destination

## DEFAULT: 20

WriterToSplunk.http.conn.pool.maxSizePerRoute = USE\_DEFAULT

##------------------------------------------------------------

## ----- WriterToFile ----- can be used for debugging

##------------------------------------------------------------

## How many threads should we use

## DEFAULT: 1

WriterToFile.saveThreadCount = USE\_DEFAULT

## Directory to save files to

## DEFAULT: ${java.io.tmpdir}/SekAseAuditor/WriterToFile

WriterToFile.saveDir = USE\_DEFAULT

## Should we format tje JSON text as "pretty" before sending it to Splunk

## DEFAULT: true

WriterToFile.json.prettyPrint = USE\_DEFAULT

##----------------------------------------------------------------------------

## END: Writers

##----------------------------------------------------------------------------