IBM® Security Identity Governance and Intelligence

Version 5.2.6.1

Performance Tuning Guide

Note: Before using this information and the product it supports, read the information in "Notices".

2nd Edition notice

Note: This edition applies to Version 5.2.6.1 of IBM Security Identity Governance and Intelligence and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

About this publication

The *IBM® Security Identity Governance and Intelligence Performance Tuning Guide* provides information on tuning middleware for IBM Security Identity Governance and Intelligence (IGI) Version 5.2.6 and Version 5.2.6 FP1.

Access to publications and terminology

This section provides:

- A list of publications in the IBM Security Identity Governance and Intelligence library.
- Links to “Online publications.”
- A link to the “IBM Terminology website.”

Online publications

IBM posts product publications when the product is released and when the publications are

updated at the following locations:

IBM Security Identity Governance and Intelligence Information Center

The http://www-01.ibm.com/support/knowledgecenter/SSGHJR/welcome site displays the Knowledge Center welcome page for this product.

IBM Knowledge Center

The http://www-01.ibm.com/support/knowledgecenter site displays an alphabetical list of, and general information about, all IBM products. Use the Search dialogue box to navigate to *IBM Security Systems* for a list of IBM Security Products.

IBM Publications Center

The http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss site offers customized search functions to help you find all the IBM publications you need.

Support information

IBM Support Portal aids with code-related problems and routine, short duration installation or usage questions. The IBM Support Portal can be found at https://www.ibm.com/support/entry/portal/support.

1. Statistics Enablement for the Database

Tracking the behavior of the database can greatly improve the ability to tune this tier for optimal performance. The following DB2 commands invoked on the database environment enable the monitor switches. The database instance must be restarted to make these effective for new connections.

```
db2 update database manager configuration using DFT_MON_STMT ON
db2 update database manager configuration using DFT_MON_BUFPOOL ON
db2 update database manager configuration using DFT_MON_LOCK ON
db2 update database manager configuration using DFT_MON_SORT ON
db2 update database manager configuration using DFT_MON_TIMESTAMP ON
db2 update database manager configuration using DFT_MON_UOW ON
```

The switches above will report on the SQL statement, bufferpool(s), lock(s), sort behavior, timestamp information, and unit of work (UOW). Statistics related to these items will now be displayed in subsequent snapshots or output from *db2top*. The following command will report whether data collection is enabled for these statistics.

```
$> db2 get dbm cfg | grep DFT_MON

   Buffer pool                          (DFT_MON_BUFPOOL) = ON
   Lock                                    (DFT_MON_LOCK) = ON
   Sort                                    (DFT_MON_SORT) = ON
   Statement                               (DFT_MON_STMT) = ON
   Table                                  (DFT_MON_TABLE) = OFF
   Timestamp                         (DFT_MON_TIMESTAMP) = ON
   Unit of work                            (DFT_MON_UOW) = ON
```

In laboratory testing, table monitoring has been known to cause a slight performance impact when enabled. Enabling this statistic should be reserved for times when the statistic is collected, and then disabled afterwards. As stated before, the database instance will require a restart after this setting is changed.

Additionally, these monitors should be periodically reset to ensure the statistics do not become stale. The following command will reset the monitors on the DB2 database. This should be invoked on the running database and does not require an instance restart afterwards.

```
db2 reset monitor all
```

In the PostgreSQL implementation, the statistics collection engine is controlled by the settings found in the file postgresql.conf. For the Identity Governance system, the following are set in the standard configuration.

track_activities	on	Collects information about executing commands.	
track_activity_query_size	1024	Sets the size for pg_stat_activity.query, in bytes.	100 - 102400
track_commit_timestamp	off	Collects transaction commit time.	
track_counts	on	Collects statistics on database activity.	
track_functions	none	Collects function-level statistics on database activity.	
track_io_timing	off	Collects timing statistics for database I/O activity	

With these settings, the pg_stat and pg_statio views can be used to collect information at runtime.

To view all the settings, the DB administrator can use the Identity Governance virtual appliance (VA) command line interface (CLI). From the root menu of the CLI, the administrator will navigate to igi → postgres → postgres_cmd. The user will then be prompted to login. The following is a useful command to extract the current DB settings.

```
select name, setting, unit, short_desc, min_val, max_val from pg_settings;
```

There are several methods which can be used to access the PostgreSQL database remotely. The tool pgadmin (https://www.pgadmin.org) is a popular method, as well as the more common *psql* command. Here is an example of a remote access to the Identity Governance PostgreSQL DB from a Linux system using the *psql* command.

```
psql -q -d igidb -h hostname -U postgres -o outputfile
```

The user will login with the VA admin password. In this case, output from any query run during the *psql* session will be written to a local file on the Linux system named *outputfile.* Refer to the man page for *psql* for formatting techniques and methods to invoke commands for scripting. To view the settings in postgresql.conf, the user may download the support file.

The following are useful views to collect statistics for runtime behavior.

- pg_locks
- pg_stats
- pg_stat_activity
- pg_stat_database
- pg_stat_user_tables
- pg_stat_user_indexes
- pg_statio_user_tables
- pg_statio_user_indexes
- pg_stat_replication (For cluster environments)

To ensure the statistics are updated, it is advisable that tools such as "runstats" for DB2 or "analyze" for PostgreSQL be run periodically on long running DBs, or when the data set changes dramatically or frequently. After an upgrade from one firmware level to the next, the DB administrator should also follow best practices by running a *reorg* on the database tables and indexes, as well as the *runstats*.

In addition to the tuning provided by the *runstats* command, the DB2 administrator should also perform a *db2rbind*. This procedure should be performed each time the database changes dramatically (large bulk loads, many new applications, many new permissions or endpoints, for example), and at regular DB maintenance windows. It generally takes a few minutes to perform and should be executed on the running database. To avoid conflicts and errors, the VA application server should be stopped. The command for invoking this procedure is listed below, where -l (lower case 'L') specifies the log file.

```
db2rbind db_name -l log_file_name all
```

The table and index statistics are generated during table initialization, and for dynamic SQL, the *runstats* command will update the statistics for improved query plans. For static SQL code, statistics are not updated by *runstats*, and will become stale. The *db2rbind* command will update the statistics for static SQL, improving query plan performance. This is particularly important

when the database approaches enterprise levels. Laboratory tests have demonstrated significant performance improvements using this procedure on enterprise level DB2 deployments.

See DB2 Documentation in the IBM Knowledge Center for additional information regarding the *db2rbind* command.

In general, DB tuning and administration best practices should be used to maximize the data tier performance. Those recommendations will be specific to the customer's data contents, size, and usage patterns. Hints for applying indexes or setting individual tunable values can be found in DB statistical data or by studying the DB log. As an example, the DB2 tunable "Maxappls" can be tuned according to the number of application connection requests. This tuning recommendation is apparent by viewing the db2diag.log.

2. Tuning the Scan Rate of the Event Queues

By default, the Event queues will scan every 20 seconds looking for work to do. Upon completing a work item from the queue, the task will return to the queue to take the next work item. If the queue is empty, the task will wait 20 seconds to scan for newly arriving work. Navigate to Task Planner → Manage → Tasks and highlight an Event queue such as Event OUT. Under the *Scheduling* tab in the right frame, the frequency of the scans can be adjusted. Note that adjusting the frequency will require the task to be stopped, then started again after the change.

The default scan rate of 20 seconds makes the Identity Governance highly reactive to changes in the system.

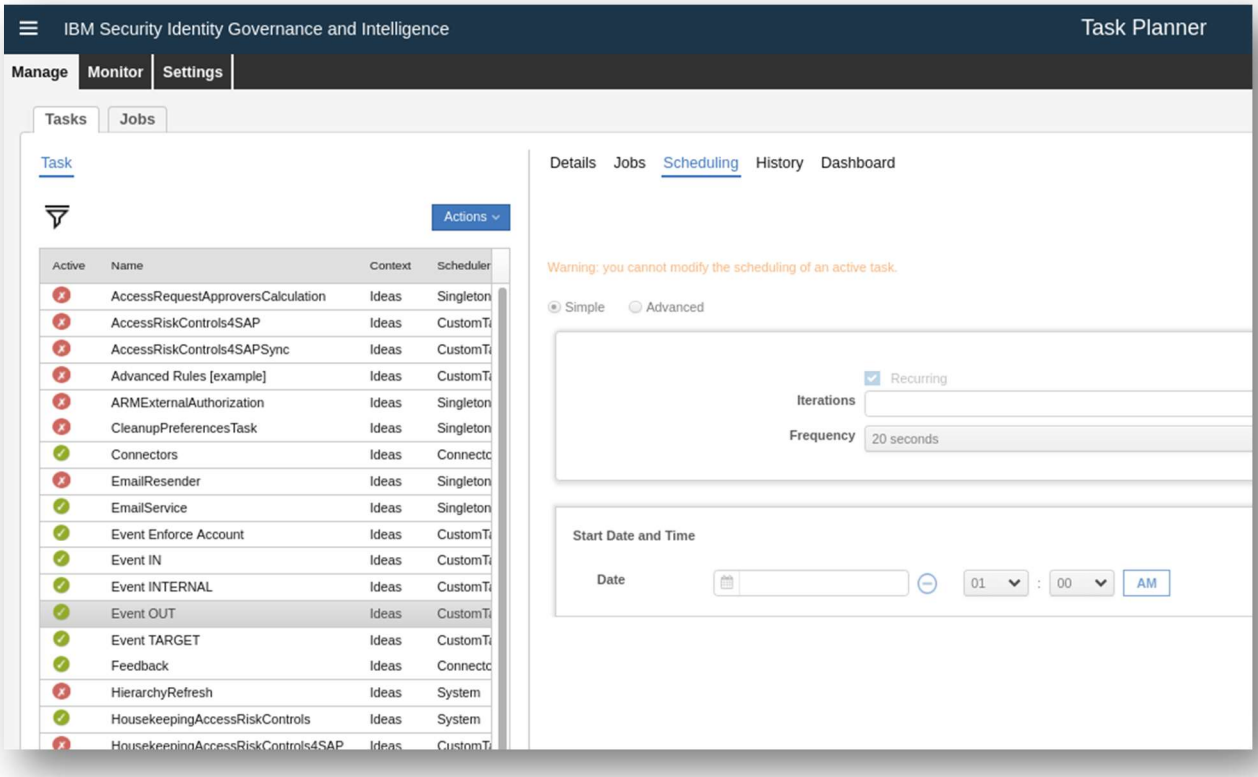


Figure 1: Modifying the Scan Rate

3. Tuning the Cache Time

In the Task Planner Module, the **cacheTime** value for a task can be enabled to improve event processing performance. In disabled mode (a value of 0), the rule flow is read from the database and compiled for each event processed. A value of 120 minutes is used by default when the cache is enabled. Navigate to Task Planner → Manage → Tasks and highlight an Event queue such as Event TARGET. Click on the Jobs tab and in the rightmost frame the value for the **cacheTime** will be displayed.

In a dynamic environment where the rules are modified frequently, a disabled cache ensures that the latest rules are always applied during event processing. However, there is a performance penalty to compile the rules for each event processed. In a stable, long running environment where the rules are established and remain unchanged for extended periods of time, caching can and should be used. Changing the **cacheTime** value to 600 minutes would result in a 10-hour cache of the current set of rules.

Using the cache results in fewer reads from the database, and fewer compiles of the rules, reducing CPU consumption on the VA and the database tier. Changing this parameter requires that the task be stopped, then started again after the change is made.

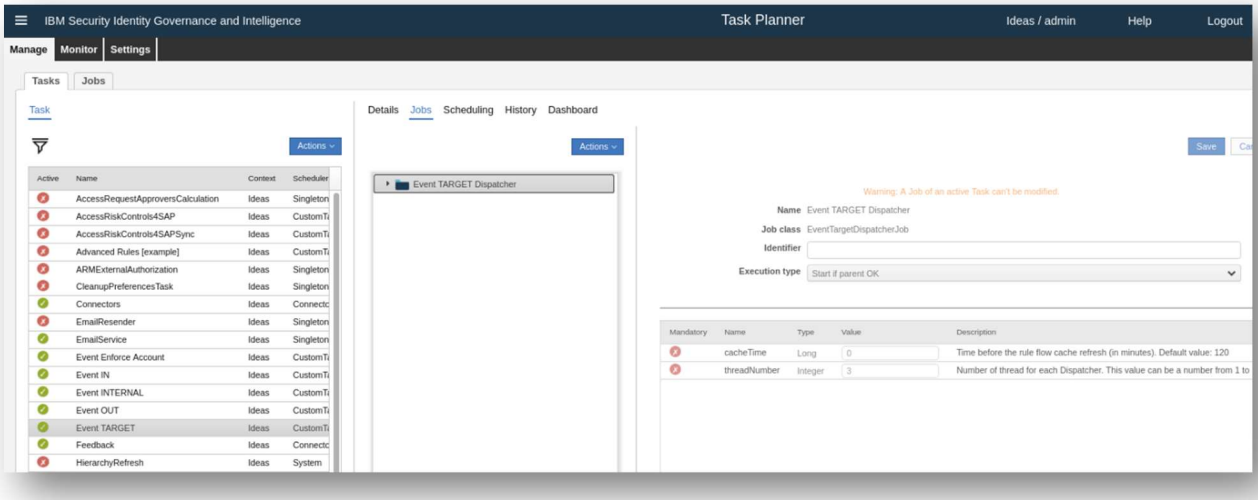


Figure 2: Modifying the Cache Time

4. Task Planner

The Task Planner is the internal scheduler for the Identity Governance product. This module is responsible for all batch and background jobs. The Task Planner has 7 schedulers.

- System Scheduler
- Reports Scheduler
- Connectors Scheduler
- Singleton Scheduler
- Custom Tasks Scheduler
- Role Mining Scheduler
- Internal Scheduler

The Internal Scheduler is new to Version 5.2.5 and is a private scheduler for the IGI system. It is responsible for handling hierarchy builds, bulk load operations, and for access risk activities such as separation of duty scans. The history is not viewable for this scheduler, and additional tasks cannot be added to it.

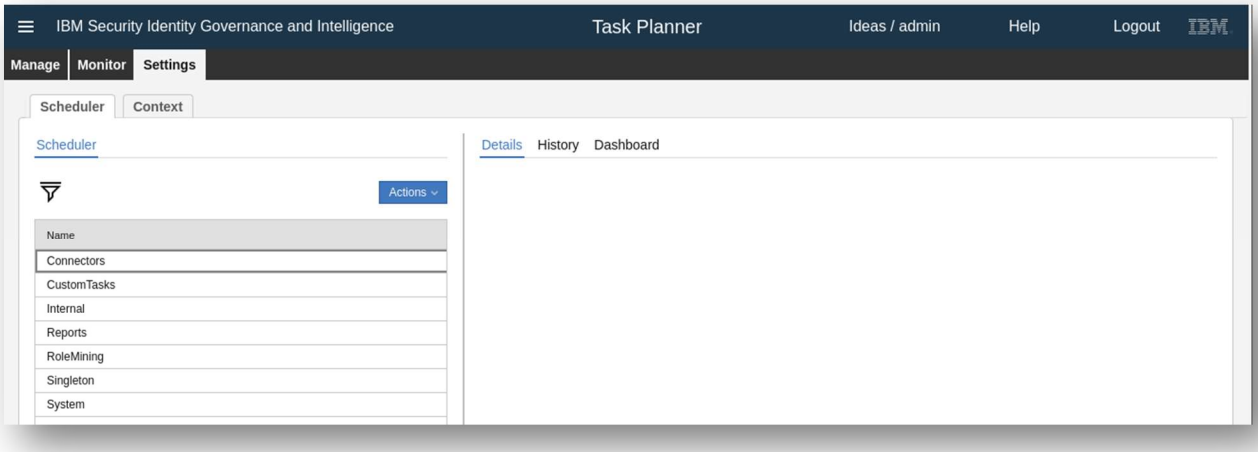


Figure 3: Task Planner Schedulers

Since Version 5.2.5, the Task Planner has introduced several new enhancements aimed at improving responsiveness. Except for the Singleton Scheduler, it is now possible to tune the number of dispatcher threads for all other Schedulers. The Singleton Scheduler, as the name implies, only has 1 thread and cannot be tuned.

Scheduler	Default No. of Dispatcher Threads	Configurable Range
Connectors	2	1-6
CustomTasks	2	1-6
Internal	5	1-15
Reports	2	1-6
RoleMining	2	1-6
Singleton	1	Not Configurable
System	2	1-6

Table 1: Scheduler Thread Properties

Great care should be taken when altering the number of dispatcher threads, especially on the Internal Scheduler. Workloads such as a hierarchy build and bulk load operations are CPU intensive. Adding threads should be done slowly, with a watchful eye on the CPU resources of both the DB and the IGI VA. Things to monitor include the CPU, the number of database connections in use, the heap usage on the VA, and the buffer pool statistics on the DB.

The number of threads for an eligible scheduler can be changed by navigating to Task Planner → Settings → Scheduler. Highlight the target scheduler in the left frame and in the Details tab in the right frame, look for the variable *org.quartz.threadPool.threadCount*. Change this to the desired thread count and click Save. You will be prompted with a caution statement and informed that the change will not take effect until the application server is restarted.

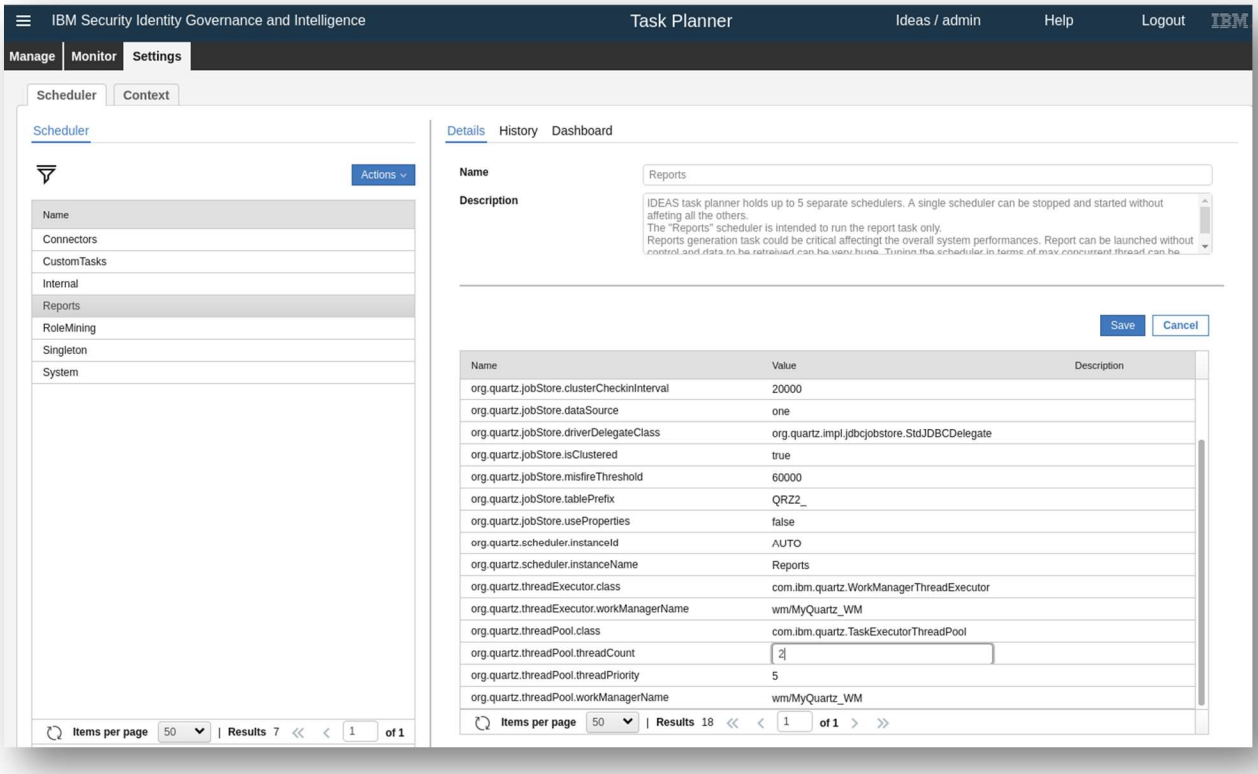


Figure 4: Changing the Scheduler Thread Count

In laboratory tests of event processing, increasing the System Scheduler threads shows a 3-4% increase in CPU per additional thread. Changing the number of scheduler threads will affect the number of simultaneous jobs but does not directly affect the number of database connections.

Depending on the type of job, the database connections may increase as a side effect. As more jobs are scheduled on the VA, the memory statistics of the VA and the DB may also be affected. Tools such as the garbage collection logs on the VA, and database snapshots on the DB can help monitor the heap consumption, the DB bufferpool behavior, and the DB connections.

It is also possible to manage how often each scheduler runs its jobs and monitor the efficiency of the threads. Each task found in the Task Planner is associated with a specific scheduler which can be seen in the fourth column of the left frame. Referring to the *Scheduling* tab in the right frame will provide information about the frequency for the given task.

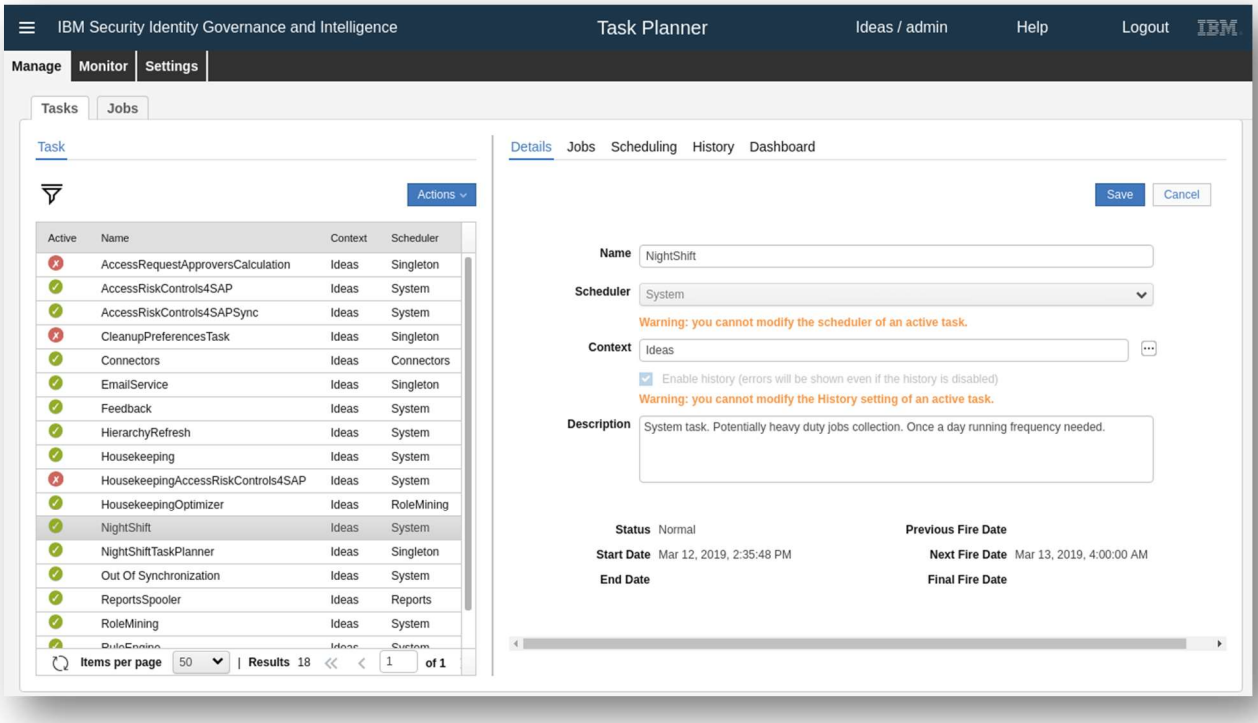


Figure 5: Mapping Task to Scheduler

Each task is a group of jobs. Referring to the *Jobs* tab will provide a list of the jobs associated with a specific task and the order they are executed. When the history has been enabled for a task, the runtime statistics are recorded in the *History* tab in the right frame. This can be used to track the elapsed time for the jobs and determine if the frequency is set appropriately. A task should not be scheduled to run in less time than it actually takes to complete the activity. This could result in blocked jobs in other tasks. Collecting history is not enabled by default. To enable it, stop the task. In the *Details* tab, click the check box next to Enable History, then Save. Start

the task in the left frame. It is not necessary to restart the Identity Governance service. When the history has been enabled for a task, all jobs associated with that task will also report their history. Navigate to Task Planner → Jobs → History. A consolidated view of all activities (whose history has been enabled) can be viewed on the Task Planner → Monitor tab. This view offers a glimpse of what IGI is running at any given moment.

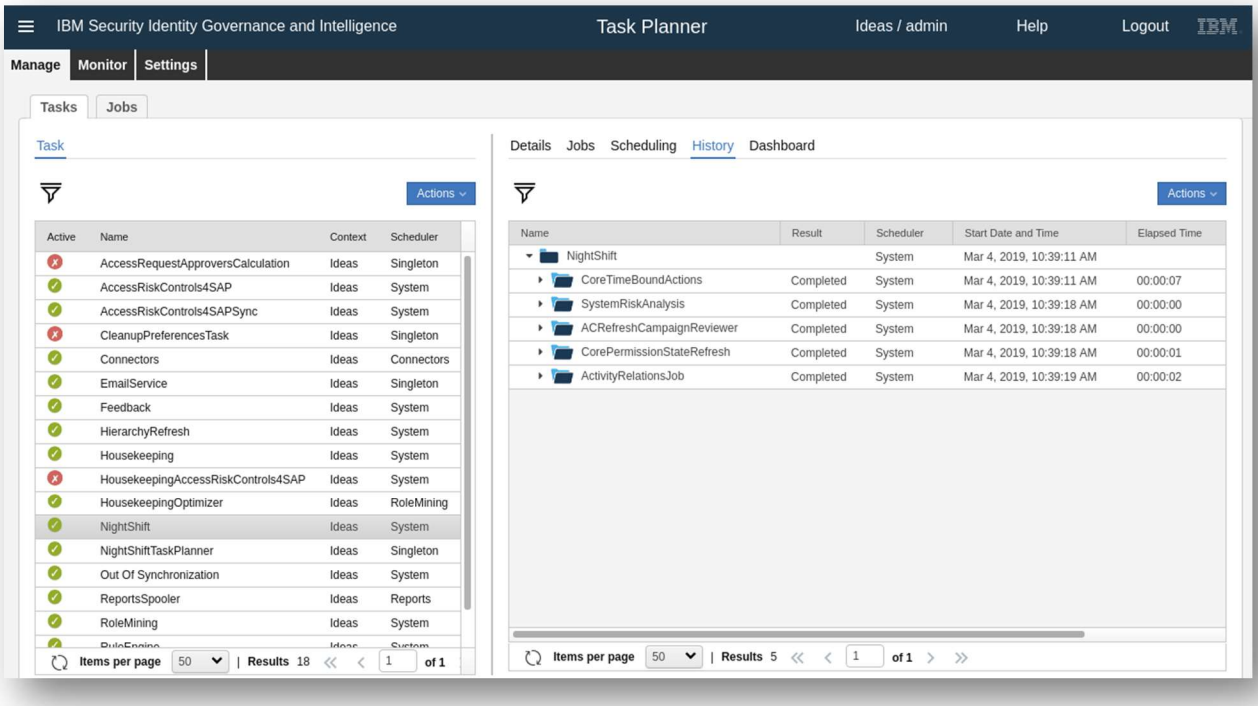


Figure 6: Job History

Prior to Version 5.2.5, the utilization at the task level could not be directly monitored. Since Version 5.2.5, a new component of the Task Planner is available that allows the user to view a chosen task. In the figure below, the Role Mining task was active for several hours. It is possible to use the quick view available with the Zoom function to choose a view for 1 hour, 1 day, 1 week, or all the runtime. Additionally, the user can choose a span of days to view by either using the date boxes above the graph, or the sliders on the timeline beneath the graph. Finally, as the mouse flies over the graph, a pop-up chart will appear with information regarding a specific time stamp.

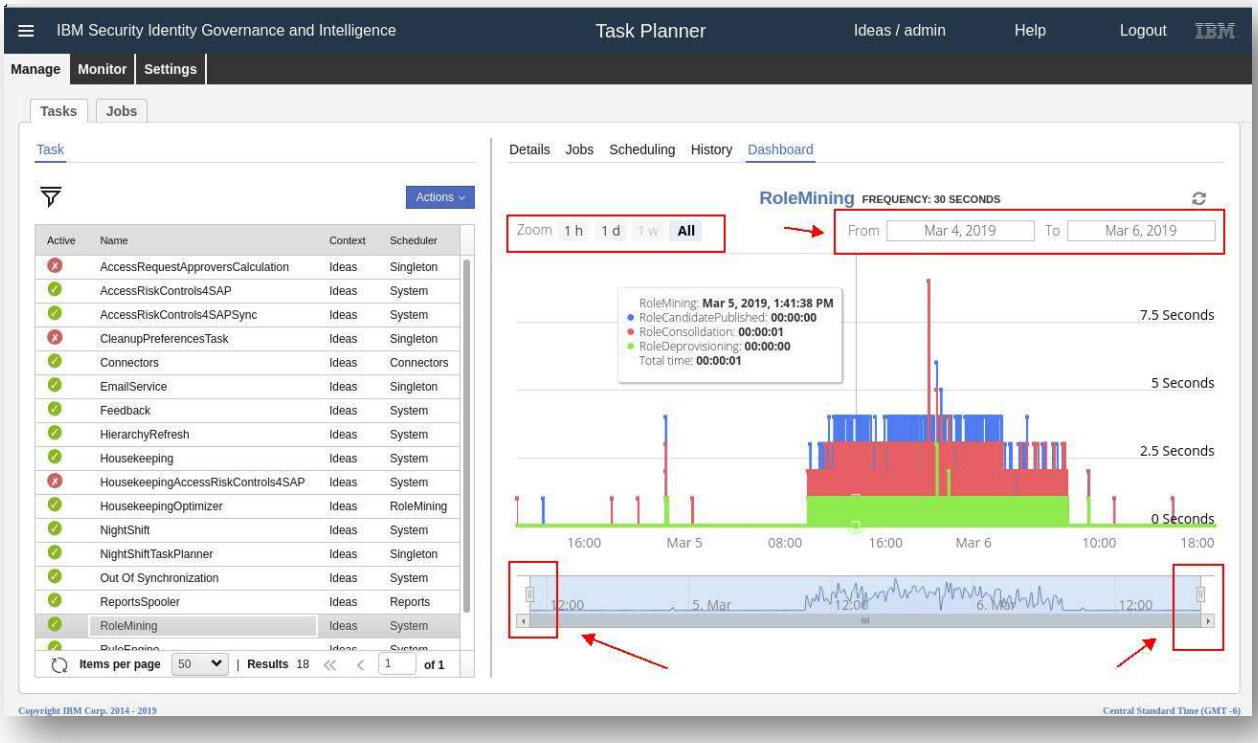


Figure 7: Dashboard Example

There is also a new view of the utilization of the threads within the Schedulers. The user can navigate to Task Planner → Settings → Scheduler → select a scheduler in the left frame → Dashboard. The right frame will display the activities of the thread behavior for the scheduler. The Internal Scheduler will not display a dashboard view as this is the private IGI scheduler.

In the figure below, the activity of the System Scheduler is shown for the last 7 hours. Dispatches shown in red indicate the task is scheduled to run at a frequency that is too high. That is, the current dispatch has not completed a prior task before the scheduler tried to dispatch it again.

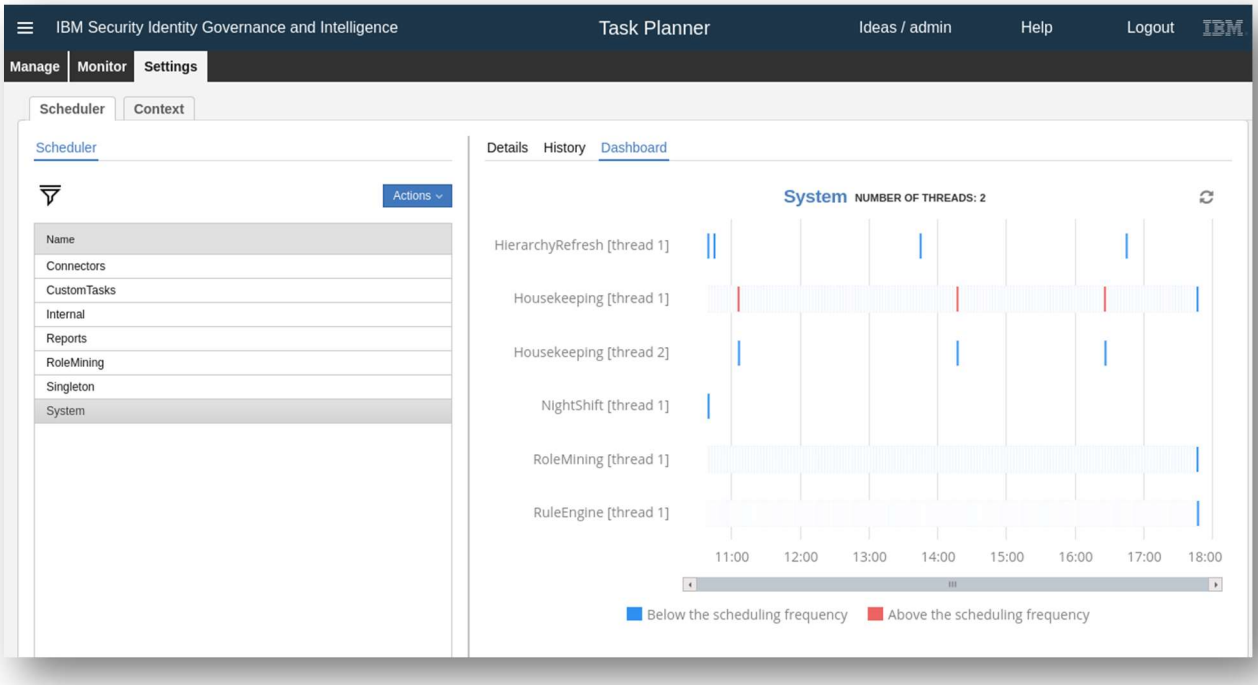


Figure 8: Scheduler Dashboard

When the mouse flies over this area, a pop-up will appear with a probable cause for the alert.

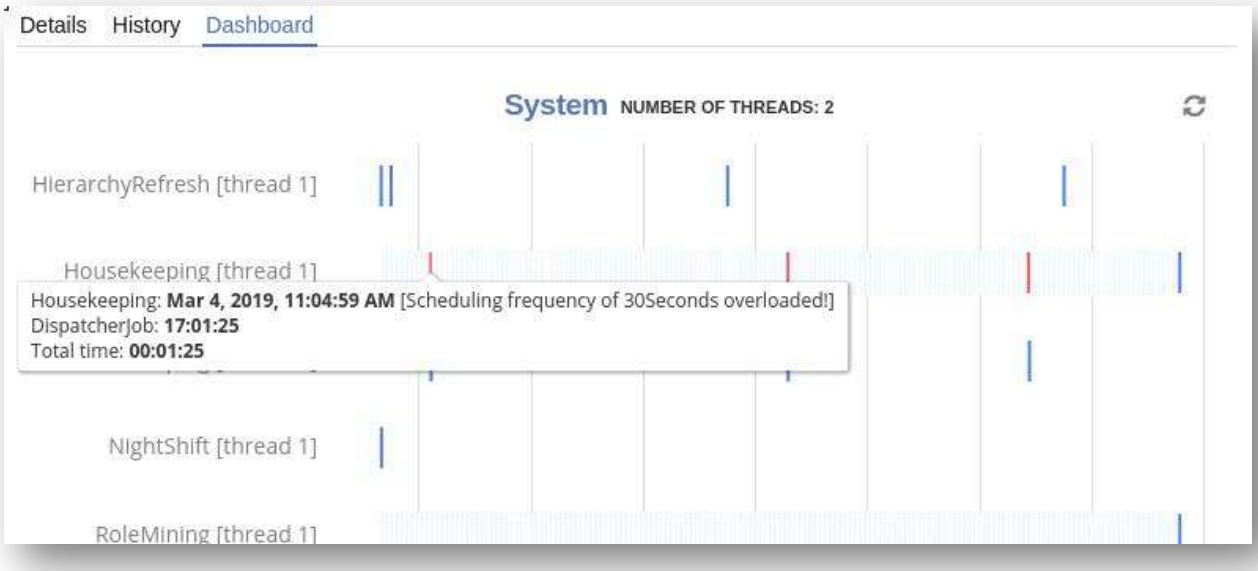


Figure 9: Scheduler Alert

Certain tasks, such as the SAP related tasks, should not be run more often than every 1-2 days. Separation of Duty analysis, for example, can take a long time and consume significant processing resources. By nature, the Separation of Duty analysis is computationally intensive. Using the history of a given task, and the built-in performance monitoring capabilities of the Administration Console, one can determine the time (from the history) and CPU requirements (from the Console) for an operation. This can help plan the frequency of such operations.

Although it is not possible to create additional schedulers beyond the 7 which are provided by default, the customer can use the Custom Tasks Scheduler to address specific needs. Such customization should be managed with care to avoid performance issues during runtime. It is recommended to enable the history of the Custom Tasks Scheduler when tasks are added to track the run time of the tasks within. The history can be disabled once the customer understands the run time and resource consumption for each task, adjusting the frequency if necessary.

In Version 5.2.5, the new Task Planner view will only be seen for new installations. For those customers who are migrating, the original task names will be maintained. To cross reference the old names with the new, a table has been included in the Knowledge Center. At the link below, refer to Table 4: *Job-to-task grouping in V5.2.5 and older versions*.

https://www.ibm.com/support/knowledgecenter/SSGHJR_5.2.5/com.ibm.igi.doc/CrossIdeas_Topics/TSKP/job_descriptions.html

5. Improving Event Processing Concurrency and Performance

There are four Tasks representing the multi-threaded event processing queues of the Access Governance Core (AGC). The four tasks are Event IN, Event INTERNAL, Event OUT, and Event TARGET. Prior to Version 5.2.5, these queues were part of the Rule Engine Task. The number of worker threads for each queue is configurable by setting the value of **threadNumber** from 1 to 10 threads. The task must be stopped for this value to be adjusted, but it is not necessary to restart the appliance. As mentioned before, closely monitor the CPU utilization when the number of threads is adjusted and increase this number slowly to avoid over-committing the CPU resources.

Increasing the number of threads in the TARGET queue can, for example, improve the processing time of events in the queue. In laboratory tests, increasing this value 1 thread at a time demonstrates an improved throughput rate for simple reconciliation events. However, alongside

the improvement in throughput, the required CPU resources also go up, and the cost per transaction increases. The best performance (highest throughput with lowest cost) for this reconciliation test on a 4-core VA was seen with 6 TARGET dispatcher threads. In the graph below, the cost is a measure of the amount of CPU resource required to support the transaction rate.

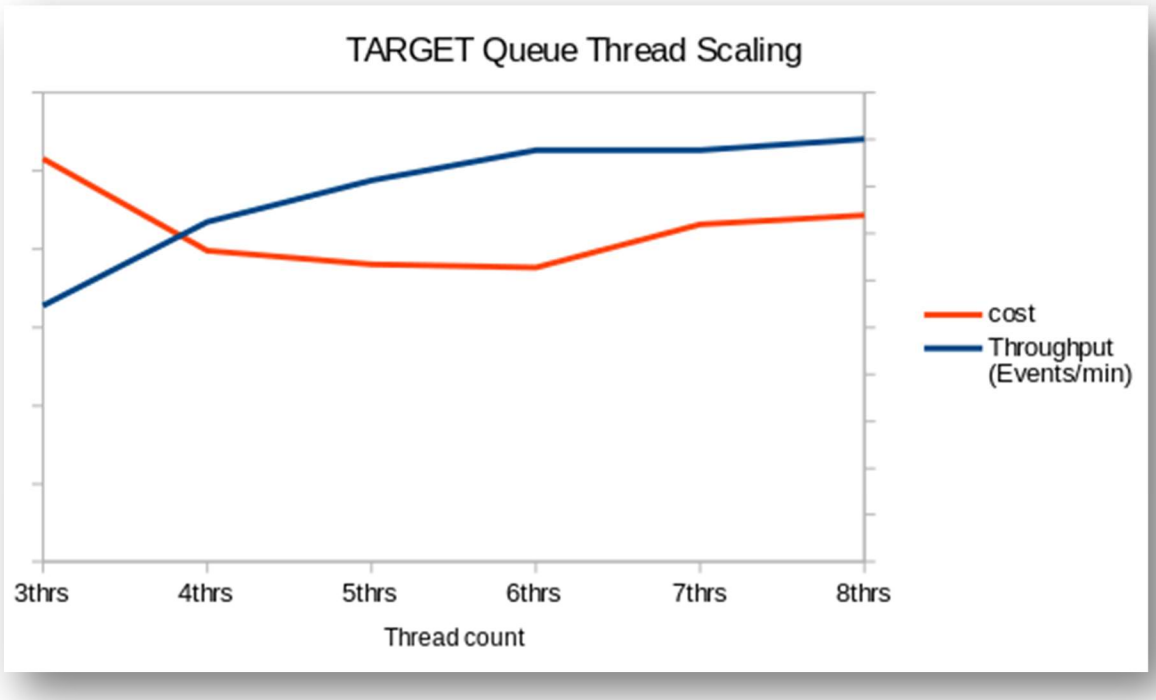


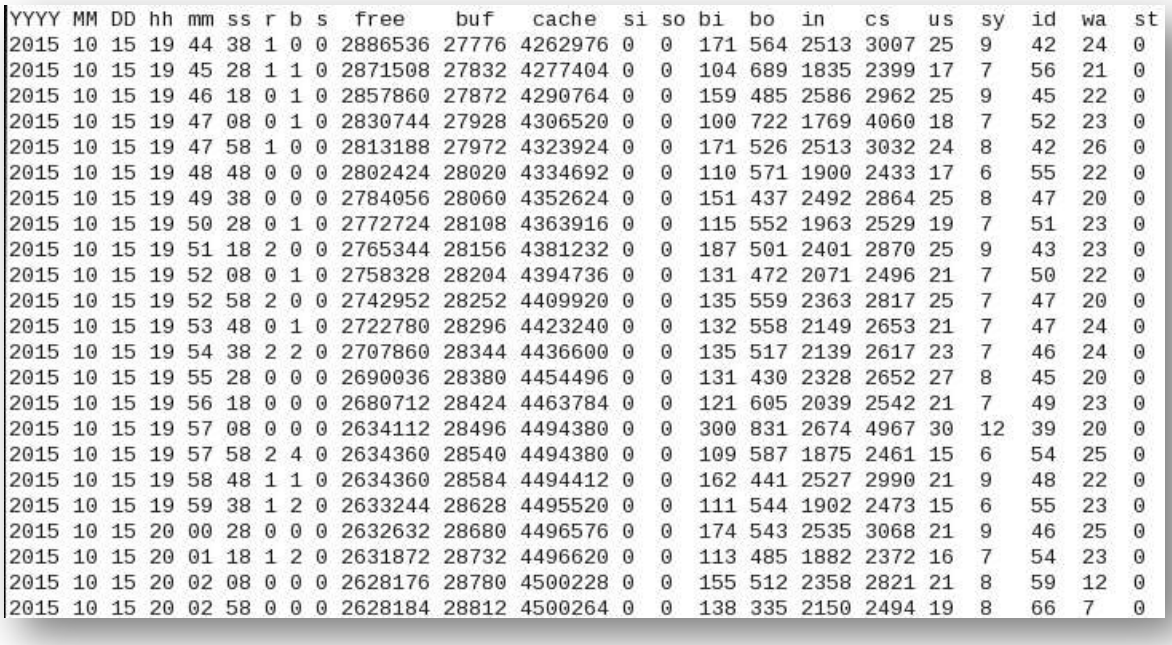
Figure 10: Dispatcher Thread Scaling Example

For a similar scaling test using provisioning (also known as 'fulfillment') on the 4-core VA, the best performance was found to be 4 OUT Dispatcher threads, and 4 Enterprise Connector threads (see Multi-threaded Enterprise Connector) configured on the target. This result is not surprising as the number of threads matches the core count. For a provisioning exercise, adding more threads than the available number of cores resulted in higher CPU utilization costs and lower throughput rates.

6. Reducing I/O Wait Time

I/O wait time can significantly reduce the performance of the Identity Governance data tier. Managing this wait time in the environment can translate to improved performance, higher transaction rates, and lower response times.

A simple *vmstat* output on the data tier can alert the system administrator to the need to tune the system to alleviate I/O wait time.



YYYY	MM	DD	hh	mm	ss	r	b	s	free	buf	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
2015	10	15	19	44	38	1	0	0	2886536	27776	4262976	0	0	171	564	2513	3007	25	9	42	24	0
2015	10	15	19	45	28	1	1	0	2871508	27832	4277404	0	0	104	689	1835	2399	17	7	56	21	0
2015	10	15	19	46	18	0	1	0	2857860	27872	4290764	0	0	159	485	2586	2962	25	9	45	22	0
2015	10	15	19	47	08	0	1	0	2830744	27928	4306520	0	0	100	722	1769	4060	18	7	52	23	0
2015	10	15	19	47	58	1	0	0	2813188	27972	4323924	0	0	171	526	2513	3032	24	8	42	26	0
2015	10	15	19	48	48	0	0	0	2802424	28020	4334692	0	0	110	571	1900	2433	17	6	55	22	0
2015	10	15	19	49	38	0	0	0	2784056	28060	4352624	0	0	151	437	2492	2864	25	8	47	20	0
2015	10	15	19	50	28	0	1	0	2772724	28108	4363916	0	0	115	552	1963	2529	19	7	51	23	0
2015	10	15	19	51	18	2	0	0	2765344	28156	4381232	0	0	187	501	2401	2870	25	9	43	23	0
2015	10	15	19	52	08	0	1	0	2758328	28204	4394736	0	0	131	472	2071	2496	21	7	50	22	0
2015	10	15	19	52	58	2	0	0	2742952	28252	4409920	0	0	135	559	2363	2817	25	7	47	20	0
2015	10	15	19	53	48	0	1	0	2722780	28296	4423240	0	0	132	558	2149	2653	21	7	47	24	0
2015	10	15	19	54	38	2	2	0	2707860	28344	4436600	0	0	135	517	2139	2617	23	7	46	24	0
2015	10	15	19	55	28	0	0	0	2690036	28380	4454496	0	0	131	430	2328	2652	27	8	45	20	0
2015	10	15	19	56	18	0	0	0	2680712	28424	4463784	0	0	121	605	2039	2542	21	7	49	23	0
2015	10	15	19	57	08	0	0	0	2634112	28496	4494380	0	0	300	831	2674	4967	30	12	39	20	0
2015	10	15	19	57	58	2	4	0	2634360	28540	4494380	0	0	109	587	1875	2461	15	6	54	25	0
2015	10	15	19	58	48	1	1	0	2634360	28584	4494412	0	0	162	441	2527	2990	21	9	48	22	0
2015	10	15	19	59	38	1	2	0	2633244	28628	4495520	0	0	111	544	1902	2473	15	6	55	23	0
2015	10	15	20	00	28	0	0	0	2632632	28680	4496576	0	0	174	543	2535	3068	21	9	46	25	0
2015	10	15	20	01	18	1	2	0	2631872	28732	4496620	0	0	113	485	1882	2372	16	7	54	23	0
2015	10	15	20	02	08	0	0	0	2628176	28780	4500228	0	0	155	512	2358	2821	21	8	59	12	0
2015	10	15	20	02	58	0	0	0	2628184	28812	4500264	0	0	138	335	2150	2494	19	8	66	7	0

Figure 11: vmstat output

In the previous figure, the database is reading and writing, although the blocks being written out dominate the I/O activity.

With buffer pool monitoring turned on, the database snapshots will reveal if the database is waiting on buffers to satisfy its I/O requirements. The hit ratio can provide information about the efficiency of the buffer pool usage. To calculate the hit ratio, refer to the snapshot for the values of logical to physical reads. In the figure below, the buffer pool data is being compared.

```
Buffer pool data logical reads              = 2294713
Buffer pool data physical reads             = 3202
```

Figure 12: Buffer pool information in DB snapshot

$$\frac{Logical\ reads - Physical\ reads}{Logical\ reads} * 100$$

Ideally, this hit ratio should be very close to 100%.

High write I/O, as seen through *iostat*, can also indicate the sort heap setting may be too small for the temporary space that is needed to sort a table or performing database functions such as hash joins. Check for write I/O at the operating system level (using *iostat)* to determine the location of the write performance issue. If hash joins occur frequently, consider increasing the value of the sort heap parameter. The settings recommended in the Identity Governance documentation are suitable for the minimum VA (4-core). A more robust appliance may require more database resource.

High I/O wait time can also be caused by a very active DB transaction log on the data tier. To mitigate this situation, the transaction log should be moved to a dedicated disk, separated from the DB data.

Although not entirely related to I/O wait time, the latency of the DB connection can also impact the performance of the system. It is recommended that the DB tier exist on the same subnet as the Identity Governance VA. The next best configuration is to house the DB in the same data center as the Identity Governance VA. In laboratory testing with a remote database in a separate data center, the increased latency has been observed to have a significant impact on UI responsiveness.

7. Bulk Load

For all but a few (rarely used) bulk load operations, the bulk load engine is multi-threaded. A single bulk load file is now split into smaller segments (called micro-jobs) and processed in parallel. With the improved concurrency since Version 5.2.2, customers should no longer need to

run two concurrent bulk loads, although this option is still available. A new bulk load can be started for every available thread in the scheduler. Since Version 5.2.5, bulk load operations are handled in the Internal Scheduler. By increasing the number of scheduler threads, one can increase the number of simultaneous bulk load operations. Care should be taken to ensure the CPU is not overcommitted when issuing more than one bulk load.

Since Version 5.2.5, a new enhancement to the bulk load operation reduces the time and memory consumption during the XLS file import. The inefficient DOM parsing code has been replaced with a custom SAX parser that ignores numerous aspects of the XLS spreadsheet, which are not consumable in IGI. This enhancement produces a memory image that is lean and compact. The more efficient bulk load will drive higher CPU utilization, so the user is advised to monitor the CPU consumption of the first bulk load before starting a second.

New error checking in this module will also provide a warning to the customer if the bulk load file is too large and should be split into smaller files. This warning message is meant to ensure the heap is not exhausted as the file is read into the temporary tables to be processed.

For long running bulk load jobs, it is advisable to disable data analysis tasks until the load is complete. As an example, the user can disable Role Mining, Data Exploration, System Hierarchy Refresh, or Separation of Duty jobs until the load is complete. This will reduce pressure on the CPUs of the Identity Governance system and avoid data analysis on an incomplete data set.

In previous versions of IGI, a performance enhancement involved moving the Housekeeping task (where Bulk Load operations were previously dispatched) to the Custom Tasks Scheduler. This allowed the housekeeping task, and thereby the Bulk Load engine, to have exclusive use of the two threads for that scheduler. This is no longer necessary in Version 5.2.2 or later versions. In fact, if the Housekeeping task had been moved to the Custom Scheduler, it should be moved back to the System Scheduler to avoid problems with future upgrades. **The tasks must be on the original schedulers to ensure that upgrades and migrations process successfully.**

Lastly, dynamic generation of bulk load data using scripting mechanisms in Microsoft Excel have shown very long load times. An example is building a bulk load spreadsheet using formulas to generate “unique names” and assigning the correct OUs and managers. The recommended method of using Excel dynamic population is to let the spreadsheet calculate all the values. One can then cut and paste all the values to another spreadsheet as values only or save as a csv file, then re-save as an XLS to strip the formulas.

8. Collecting Java Core Dumps

Since Version 5.2.5, verbose garbage collection is turned on by default in the VA but can be disabled via the CLI. From the root menu of the CLI, the administrator must navigate to igi → verbose_gc. The Identity server must be restarted from the Administration Dashboard if the garbage collection state is changed. Verbose garbage collection output is written to the verbosegc logs and does not affect the javacore. The javacore will list garbage collection statistics for a snapshot of time independent of whether verbose garbage collection is enabled. Enabling verbose garbage collection allows a complete record of all garbage collection activity to be captured, instead of a random short snapshot.

The customer can download the support file from the VA which will include a copy of the verbosegc.log in *download_dir*/opt/ibm/wlp/usr/servers/igi/logs. Using the CLI, it is possible to enable, disable, and clear the log. To clear the log, the VA admin should enter the CLI path igi → logs → clear. An index will be presented. Option 4 will list the VA server logs. Choose the options to clear the verbosegc logs. Clearing the log does not require restarting the VA server. The same procedure can also be used for the Broker Application logs.

Figure 13: Clear VA Logs via CLI

To collect a java core file, the Administrator must navigate to Manage → Maintenance → Core/Heap Dumps on the Administration Dashboard. Choose the Generate option, then select which application server to collect.

The user will then click the check box to choose core dump, heap dump, or both. The heap dump will cause significant I/O for a short period of time, a condition which is most dramatic on the PostgreSQL environment where the DB is embedded. The customer should avoid collecting heap dumps unless instructed to do so by a field support technician. Additionally, such activities should be planned for a time when the VA is not busy with other CPU intensive activities.

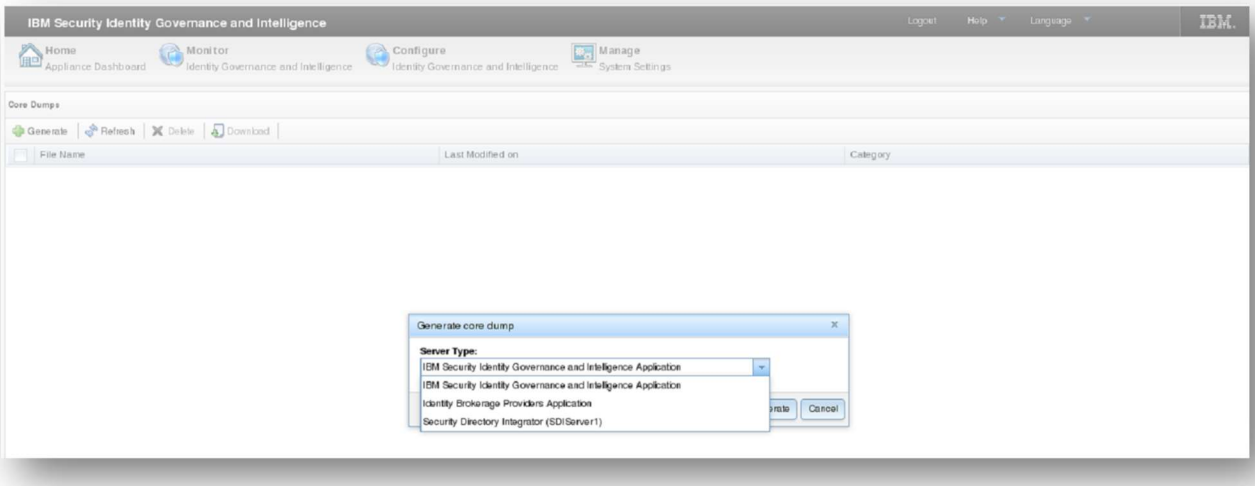


Figure 14: Generating a Java Core File

The appliance also includes an API method to generate a javacore, core dump, and/or heap dump.

Request

URL:

```
https://{appliance_hostname}:9443/v1/dmp_mgmt
```

Method:

```
POST
```

Parameters

Parameter	Description
appliance_hostname	Host name of the appliance.

Headers

Header	Description
`Content-Type:application/json`	Required for requests to the service.
`Authorization`	Basic Authentication header.
`Accept:application/json`	Required for requests to the service.

Request Example

```
POST https://{appliance_hostname}:9443/v1/dmp_mgmt
 POST_DATA:
 {
                generate_core_dump: "false",
                generate_heap_dump: "false",
                server_name: "igi"
 }
```

Response

Code: 201

Created

Figure 15: API to Generate Java Cores, Core Dumps, and Heap Dumps

The following remote command will cause a javacore to be created on the appliance for the IGI application server:

```
curl -s -k --user admin:admin_password  -H "Accept: application/json" --data
'{generate_core_dump: "false", generate_heap_dump: "false", server_name:
"igi"}' POST https://appliance_hostname:9443/v1/dmp_mgmt
```

The javacore is created by default, and if the user requires a core dump or heap dump, the value can be set to "true". Once created, the user can then use the Administration Dashboard to list/download the core or heap dumps or use another API. The API definitions can be found from the Administration Dashboard by navigating to Help → Web Services.

9. PostgreSQL Database

Since Version 5.2.2, Identity Governance introduced support for PostgreSQL. If the database type is set to PostgreSQL during the appliance installation, the database configuration is created automatically within the VA. The actual PostgreSQL database can either be truly embedded within the VA, or in an external tier where the database data is accessible via NFS mount. Both configurations have performance implications, but in both cases the actual DB administration runs inside the VA.

9.1 Embedded PostgreSQL Database

An embedded database is a desirable configuration for simple Proof of Concept or demo situations, because administration is low and a hardware tier is eliminated. The PostgreSQL infrastructure itself has a low footprint requirement which makes it ideal for embedding in the VA. An embedded PostgreSQL database environment requires higher resource consumption than the standard external DB2 database, making it critical to increase memory and CPU allocation to ensure a stable operation of the environment. When the database is co-resident in the VA, the CPU and memory resources will be taxed additionally to provide services to the Identity Governance processes, as well as the database management processes. In laboratory tests, the CPU requirements on the VA are 2 to 3 times higher when running with PostgreSQL, versus the combined requirements of a VA and DB running with DB2. The additional memory and CPU requirements are most important in the PostgreSQL cluster scenario when data replication is enabled. Despite additional memory and CPU, the performance of this environment also falls behind that of DB2. At this time, PostgreSQL is not recommended for mission-critical environments, production, or deployments where performance requirements are high.

In laboratory tests of a single VA configuration (not clustered), the mix of CPU utilization for Identity Governance processes and DB processes varies by operation. For reconciliation

activities, the CPU utilization is evenly split between the Identity Governance processes (45%) and the PostgreSQL database processes (45%). For provisioning activities, the Identity Governance consumes more than half of the available CPU resources (56%) compared to the PostgreSQL DB processes (which consumes around 35%). It is clear that if the VA configuration includes an embedded database, the CPU resources of the VA will need to be increased by at least 45% to avoid CPU exhaustion.

Beyond simplified administration, another advantage of the embedded PostgreSQL database can be found in the cluster configuration. When the VA is configured in a cluster, database replication can be enabled to provide an automatic High Availability (HA) scenario as a secondary node will house a copy of the database. Automatic failover capabilities are not supported, but the Administrator can use the Administration Console to manually promote the secondary node. To ensure failover capabilities are preserved, all nodes need to be configured with memory and CPU resources that match the primary node. As stated previously, additional resources are required to handle the processing requirements. Although data replication is a desirable feature, it comes at a cost. When data replication is enabled, scaling is problematic on a 2-node cluster. As an example, in laboratory tests of a reconciliation in a cluster, the data replication features require 1.4X the processing power compared to the same operation without data replication. If data replication is enabled, it is especially important that the nodes participating in the cluster be on the same subnet to avoid latency concerns.

9.2 NFS Mounted PostgreSQL Database

The first thing to know is Identity Governance data replication services are not available if the PostgreSQL database is moved to an external mount point on an NFS server. In this case, it is left to the customer to implement HA on the NFS environment. As previously mentioned, best practices for a data tier of any type recommend the database be kept close to the Identity Governance server, within the same subnet if possible. During initial deployment of the VA, the database should be moved to the NFS prior to loading the data. As the database is being moved to the NFS, the database server is not running. Any Identity Governance transactions requiring database access will be stalled. In laboratory tests, moving an empty PostgreSQL database takes 2-3 minutes. A database of 120K users and 160K entitlements took 30-40 minutes.

If a cluster is moved to an NFS mount configuration, the slave database will disappear, and the cluster synchronization will instruct all member nodes to begin using the NFS definition. There is currently no automated method to migrate back to an embedded database configuration, once the database has been moved to an NFS mount. To return to an internal version of the PostgreSQL database, the administrator must use backup/restore procedures. Refer the IGI product documentation for instructions.

The second important thing to know about an NFS mounted PostgreSQL DB is that the database processes are still running on the Identity Governance VA. Although the data is now housed externally, the resources required to manage the data (memory and CPU) will remain high on the VA.

The Identity Governance VA uses NFS3. The default options are seen below. Additional options can be set when the mount point is created through the Administration Console.

```
(rw,relatime,vers=3,rsize=32768,wsize=32768,namlen=255,hard,nolock,proto=udp,
port=65535,timeo=7,retrans=3,sec=sys,mountport=65535,mountproto=,local_lock=a
ll,addr=9.xxx.xxx.xxx)
```

The VA network interface will rely on the settings of the hardware on which it is running. The administrator is advised to inspect the Hypervisor of the underlying hardware to determine the NFS settings for Auto-Negotiation and the adapter speed. Auto negotiation should be turned on if the transmission rates are not known for the adapters on the end points, or the intermediate routers, switches, and hubs.

10. User Interface Dashboards

The User Interface (UI) has been updated in Version 5.2.2 with a new framework and new default options. One of those default options is an improved dashboard design for three personas: Employee, User Manager, and Application Manager. If a user logs into the Service Center with any of these personas, the dashboard presented to him will include panels representing content loaded especially for him. User and Application managers might see access requests awaiting approval, delegation requests in process, or alerts from various applications. An employee user might see the status of a request for access, for example. This dashboard can be further customized to include additional content, swap out content, or remove content altogether. While these real time dashboards are desirable, they generate load on the VA server and the database. When users navigate to the various views, default queries run to populate those views. At times when the VA is lightly utilized, this additional CPU consumption may go unnoticed. However, at peak hours, the login load may interfere with other jobs, or the user may experience delayed login times, page load times, or intermittent failures.

To mitigate the pressure on the VA and the database, the Administrator can disable the dashboard panels for the users with the following procedure. There are two steps to disabling the dashboards. In the first step, the Administrator will login to the Administration Console and

navigate to the Access Governance Core → Settings. The Administrator will need to check the boxes next to “Disable certification dashboards” and “Disable Access Requests dashboards”. Save the changes. This change will remove the selected panel from the dashboard at login time.

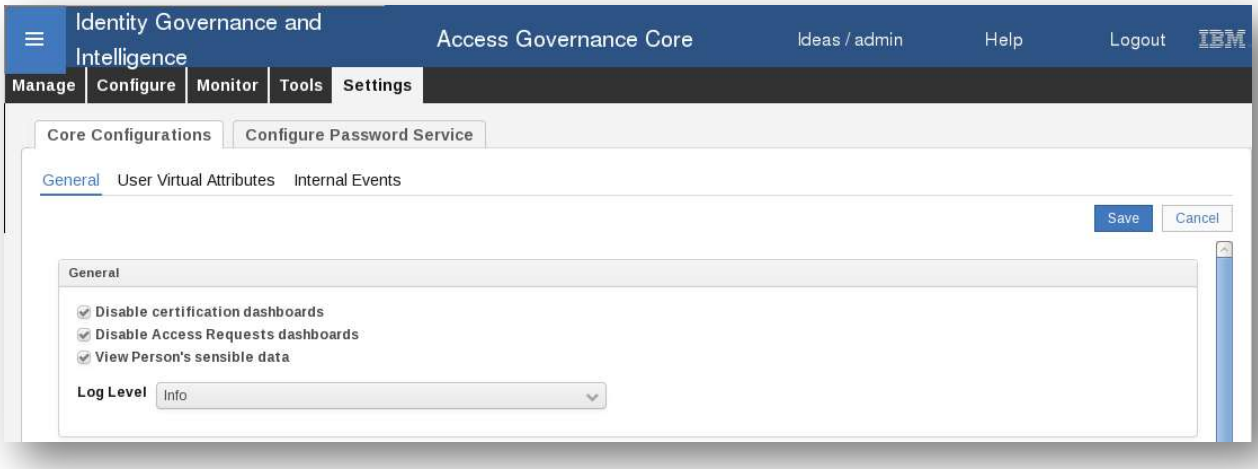


Figure 16: Disable dashboards

In the second step, the Administrator will navigate to the Report Designer Module → Configure → Assignment → Entitlement → Report Dashboard. In the left frame, set the Type to *Business Role* and click Search. In the list of roles which appear in the left frame, check the box next to User Manager. The right frame will be updated with the panels which will be displayed for this user persona. Check the boxes next to the dashboards to remove, then select Actions → Remove. This action will ensure the reports associated with the User Manager persona are not automatically generated each time that persona logs in.

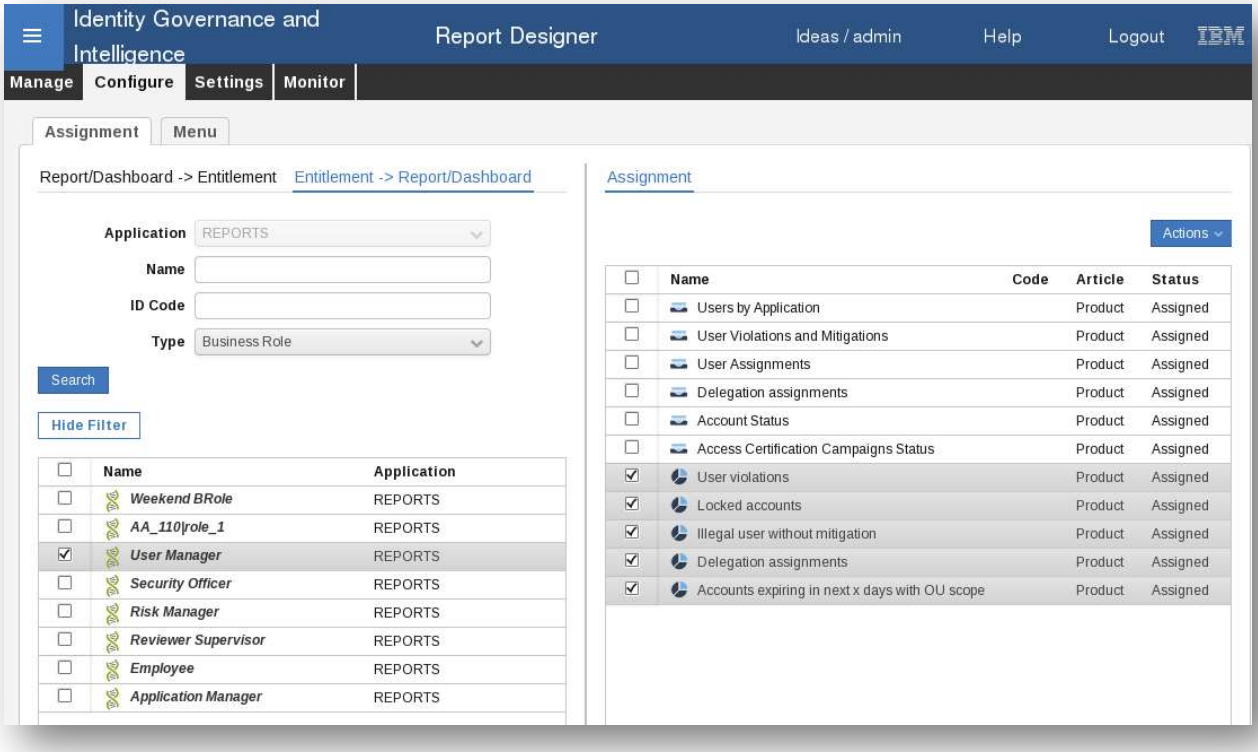


Figure 17: Remove Reports

Another method to improve the scaling of the UI is related to general best practices for a long running DB of any type. After sustained utilization of the UI, the table statistics and data access statistics should be updated. This reorganization should translate to reduced response times for frequently used data.

11. Improving Access Request Module Response Time

When the user logs into the Access Request Module, the query to populate the first landing page could delay the load of the page if there are many entries in the View Requests tab.

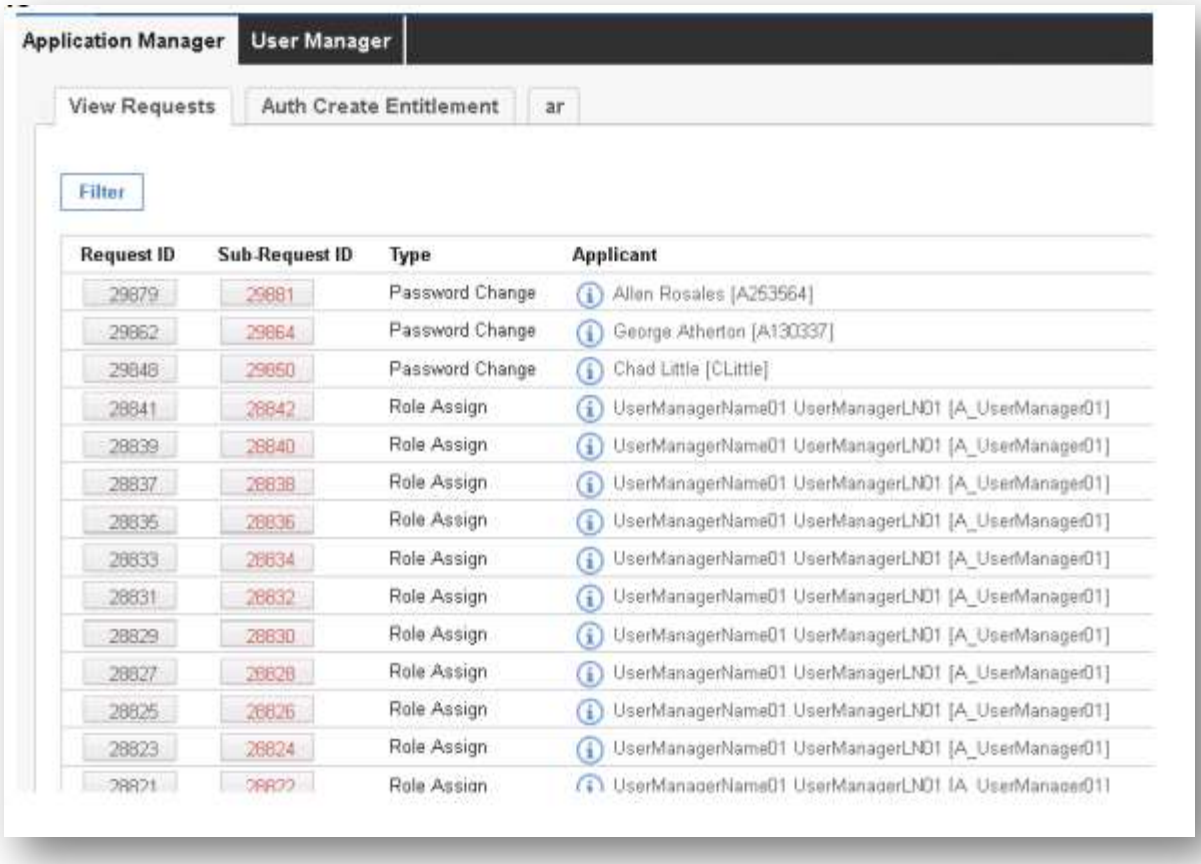


Figure 18: Default Login to ARM

These entries may, or may not, be of interest to the user upon first access. To mitigate the performance of first access, the first tab containing this information can be moved. That is, the administrator can customize the order in which the tabs appear to the users, delaying the query to populate the panel until the user actually chooses to see this data.

To reorder the tabs, the administrator can navigate to Process Designer → Manage → Process. Select the workflow in the left panel, then Actions → Maintenance.

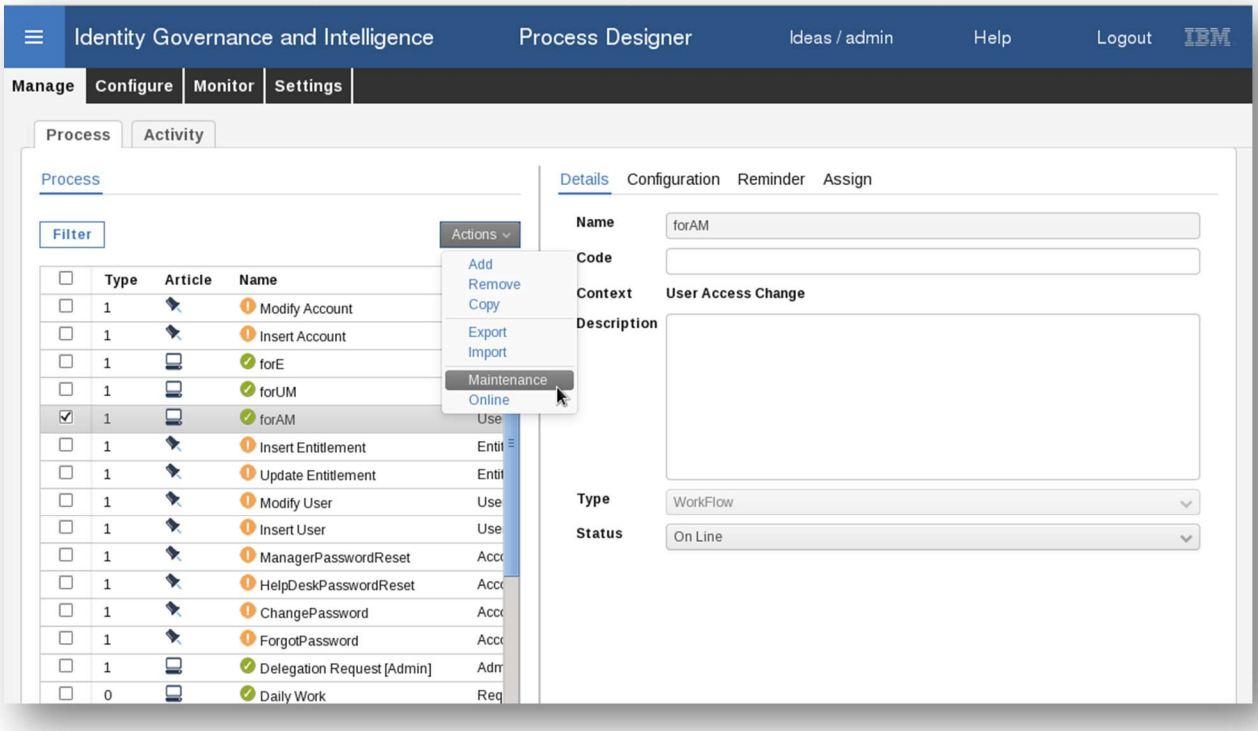


Figure 19: Select Workflow

The right panel will update for the given workflow. Navigate to Assign and select the Application. A new panel will appear on the right. Highlight a tab and then move up or down. The recommendation is to move the "View Requests" tab away from the front of the list by moving another tab up (and to the left on the actual login page as the first tab). This results in a panel that shows less data when the user initially navigates to this screen.

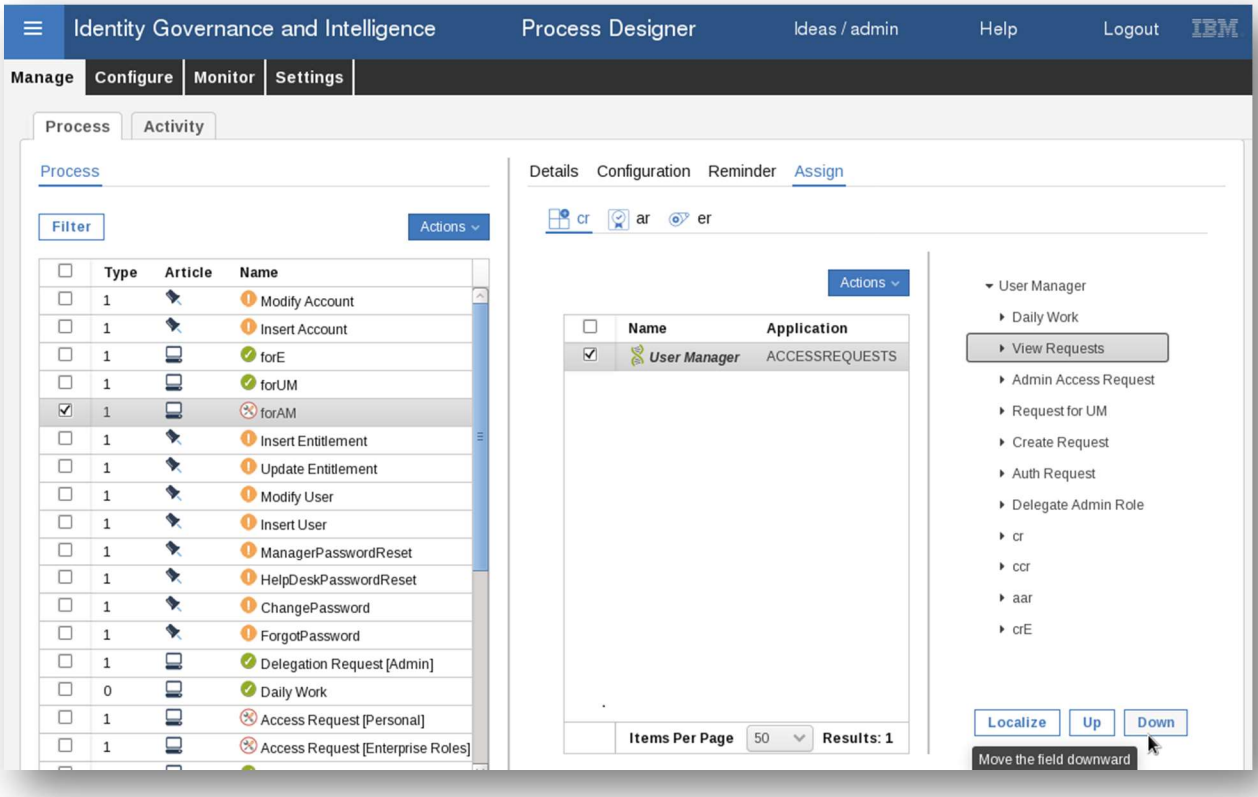


Figure 20: Move 'View Requests' Tab Down

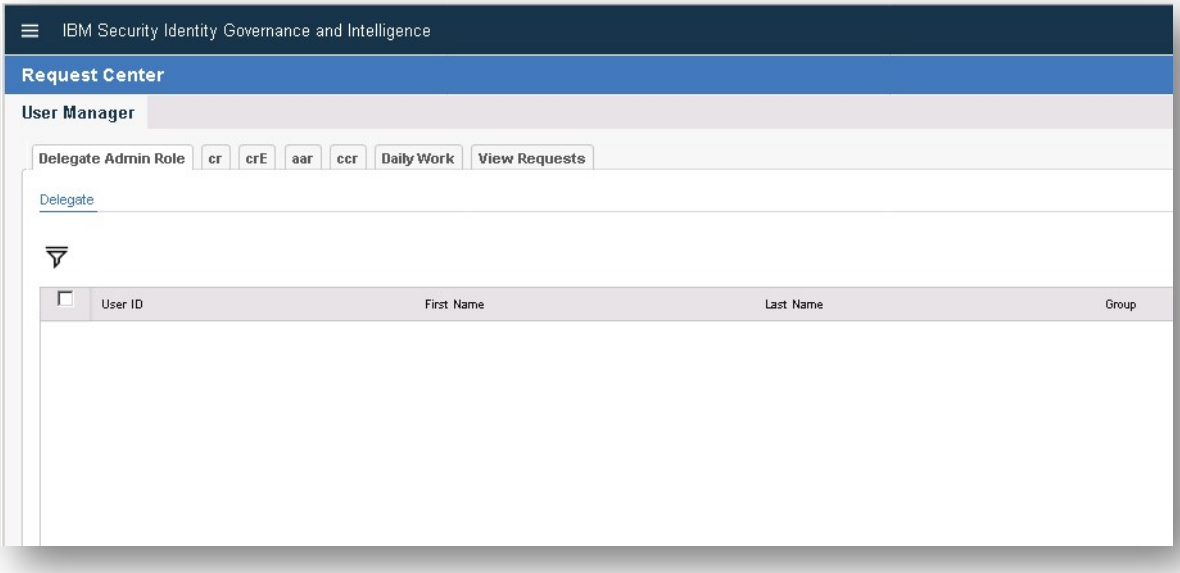
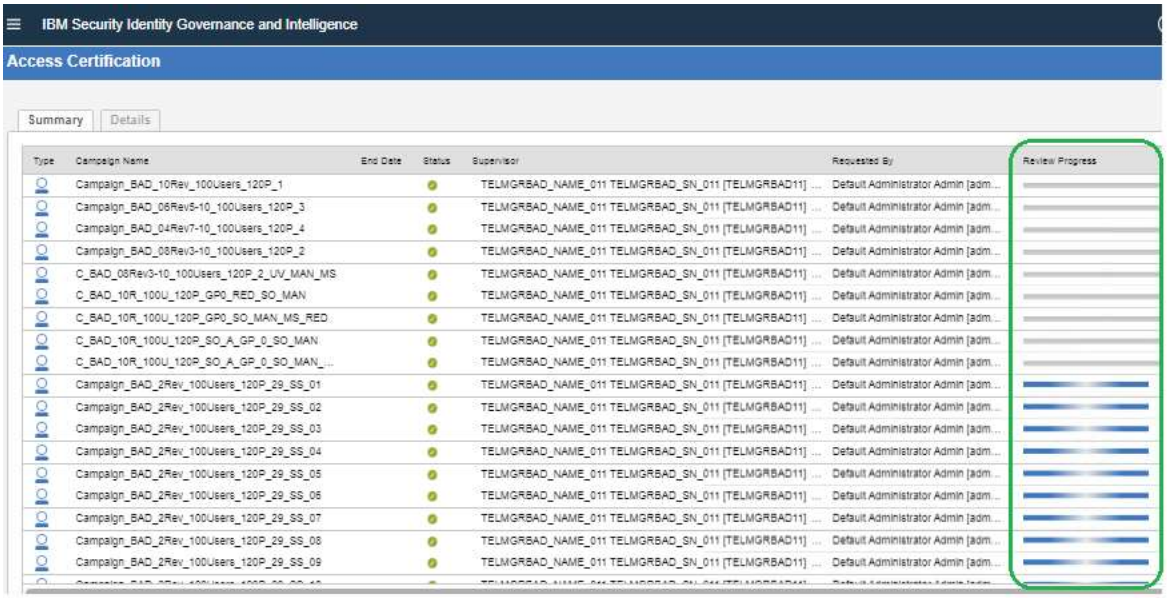


Figure 21: Example of Initial Page with No Data

12. Statistics Progress Bar on Campaign Certification

Beginning in Version 5.2.6, the option to disable the “View Statistics for Access Certification Campaigns” is dropped. The Progress Bar statistics for Campaigns Certification are solved progressively and allow the reviewer to navigate in details of campaigns without to wait that all progresses bar are solved.



13. Disable Statistics Data for Permissions of a Single Campaign

Beginning in Version 5.2.5 FP1 IF001, it is possible to disable the statistics progress bar for all permissions of a campaign certification. This improves the response time when navigating to the permissions detail of a campaign by avoiding the statistics calculation when viewing the details of the campaign. To disable statistics on the specific campaign certification, navigate to Access Governance Core → Configure → Certification Campaigns, select the campaign, in the Details tab in the right frame, and uncheck the option “Show Certification Progress Bar”.

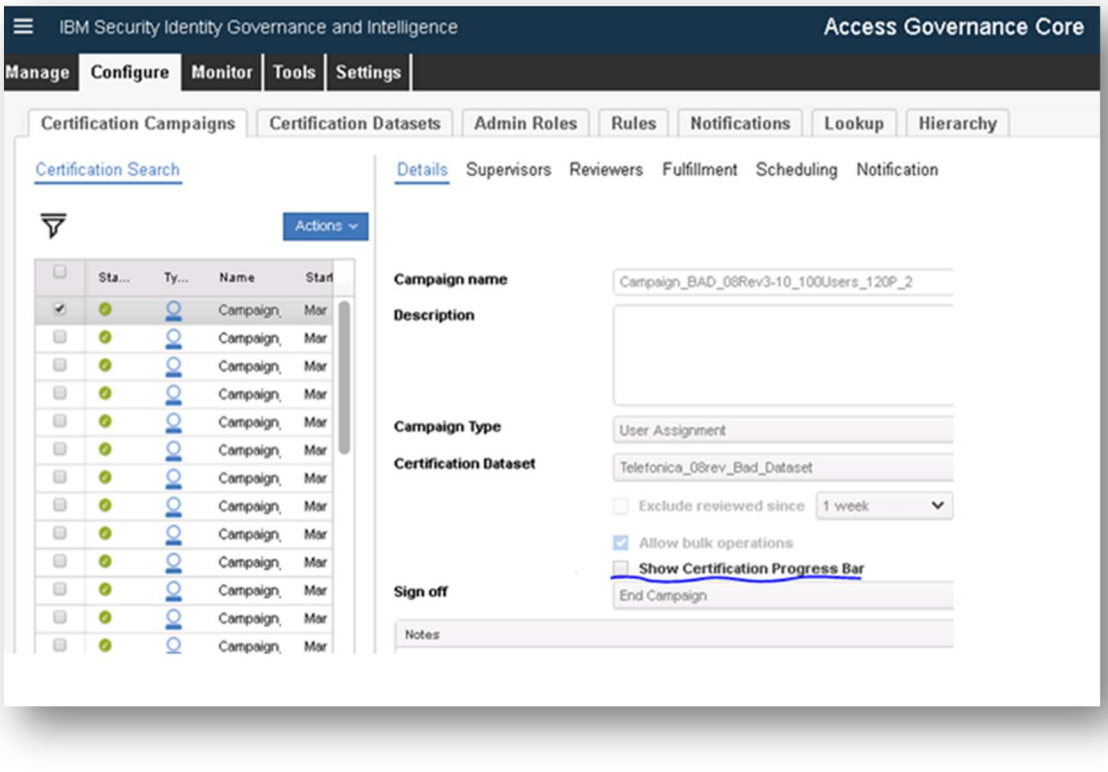


Figure 22: Disable Progress Bar

14. UI Response Time at Application Server Restart

In laboratory testing, connections to the IGI User Interfaces are slow after a WAS restart and may be due to initial caching. The response time to load dashboards or to connect to a module, such as Access Request Module or Access Certifier, suffer on the connection following the restart. Avoid applying stressful loads immediately after the WAS restart, then use an initial session login to the IGI application to repopulate the cache.

15. The Internal Security Directory Integrator

Beginning with Version 5.2.2, there are several important updates to the internal Security Directory Integrator (SDI) service. Firstly, the customer can now create multiple SDI instances and secondly, the internal SDI can be used for general Directory Integrator services with the Identity Life Cycle Management and the Identity Brokerage. That is, there is no longer a requirement to use an external Tivoli Directory Integrator for operations which require Brokerage services.

The customer will use multiple SDIs if there are adapter conflicts amongst the IBM Security Identity Adapters resident on the Identity Governance configuration. These SDI instances also provide high availability services in cluster environments. Up to a maximum of 10 SDI instances can be configured. Each instance will have a heap associated with it which is configurable at creation time. Laboratory tests for multiple SDI instances indicate that a max heap setting of 1GB is sufficient to achieve performance and resource requirements, without putting too much pressure on the memory subsystem. In laboratory tests, each additional SDI translates to 6% additional CPU consumption. In a system where there is abundant CPU available, this will not translate to a performance impact. However, a system with high CPU demands will likely see an impact for multiple SDI instances.

In laboratory testing, using the single internal SDI rather than an external TDI server for Identity Life Cycle Management and the Identity Brokerage operations, results in an additional 2% CPU consumption on the database tier.

16. Hierarchy Build and Hierarchy Refresh

Version 5.2.2 introduced significant improvements to the hierarchy build calculations. In Version 5.2.5, additional enhancements have been made to the hierarchy build, and the hierarchy refresh engine has been restructured. The result is a more powerful engine with higher calculation rates. Previous settings for hierarchy scheduling should be reexamined when migrating to Version 5.2.5 to ensure the CPU is not overcommitted.

A general rule of thumb for any hierarchy is to ensure the attribute upon which the hierarchy is built has an index. For example, when building a simple manager hierarchy, the attribute in the user's data which contains the manager's name or serial number should be indexed in the **igacore.user_erc** schema and table.

17. Clearing the Event Queues

The Event Tasks are responsible for processing events in the work queues. Searching the queues is a normal part of the operation of these tasks, as well as updating and inserting events. If the queues contain many entries, the time it takes to search, update, or insert into the queue will cause delays in event processing. While 100, 10,000, or even half a million events may not affect event processing performance, letting the queues grow beyond this might. A useful maintenance strategy is to clear the event queues periodically to reduce the search time. How often to clear the

queues is dependent upon the activities of the Identity Governance environment and how often events are created. The customer can use the filter option to list only those events which are successfully completed and remove them.

For example, events in the Access Governance Core queues should be inspected, AGC → Monitor.

1. Scheduled Tasks
2. TARGET inbound – Account events
3. TARGET inbound – Access events
4. OUT events
5. IN – User events
6. IN – Org. Unit events
7. INTERNAL events

Clearing the event queue may not be an option for those customers who need records of event processing for auditing purposes. In this case, archiving might be a compromise for auditing vs. performance. See the Appendix for an example of archiving event records.

18. Enabling SNMP for Performance Monitoring

The Identity Governance product offers monitoring from the Administration Dashboard. The user may navigate to Monitor → Monitoring. There are three options to view common performance statistics: Memory, CPU, and Storage. The Memory graph will provide a view of memory statistics for 1, 3, 7 or 30 days. In the same way, the CPU utilization and VA Storage can be viewed in graphical format.

There is an additional option for performance monitoring available. Performance of the Identity Governance product can be monitored via the Simple Network Management Protocol (SNMP). Using this popular tool, one can query the VA for standard performance statistics. To configure the VA for an SNMP connection, log onto the Administration Console → Monitor → Monitoring → SNMP Monitoring. Check the radio button next to SNMP Monitoring and click Reconfigure.

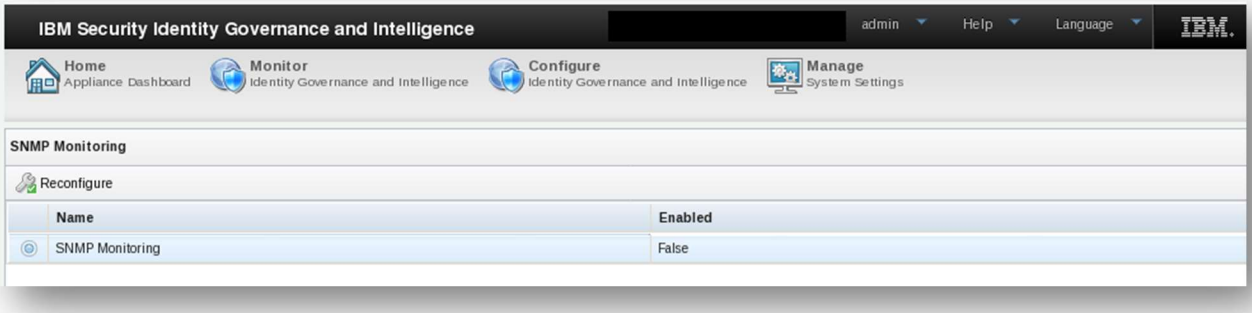


Figure 23: Enable Monitoring

A pop-up window will appear to specify the SNMP version, port, and the community name. Fill in the values and click Save Configuration.

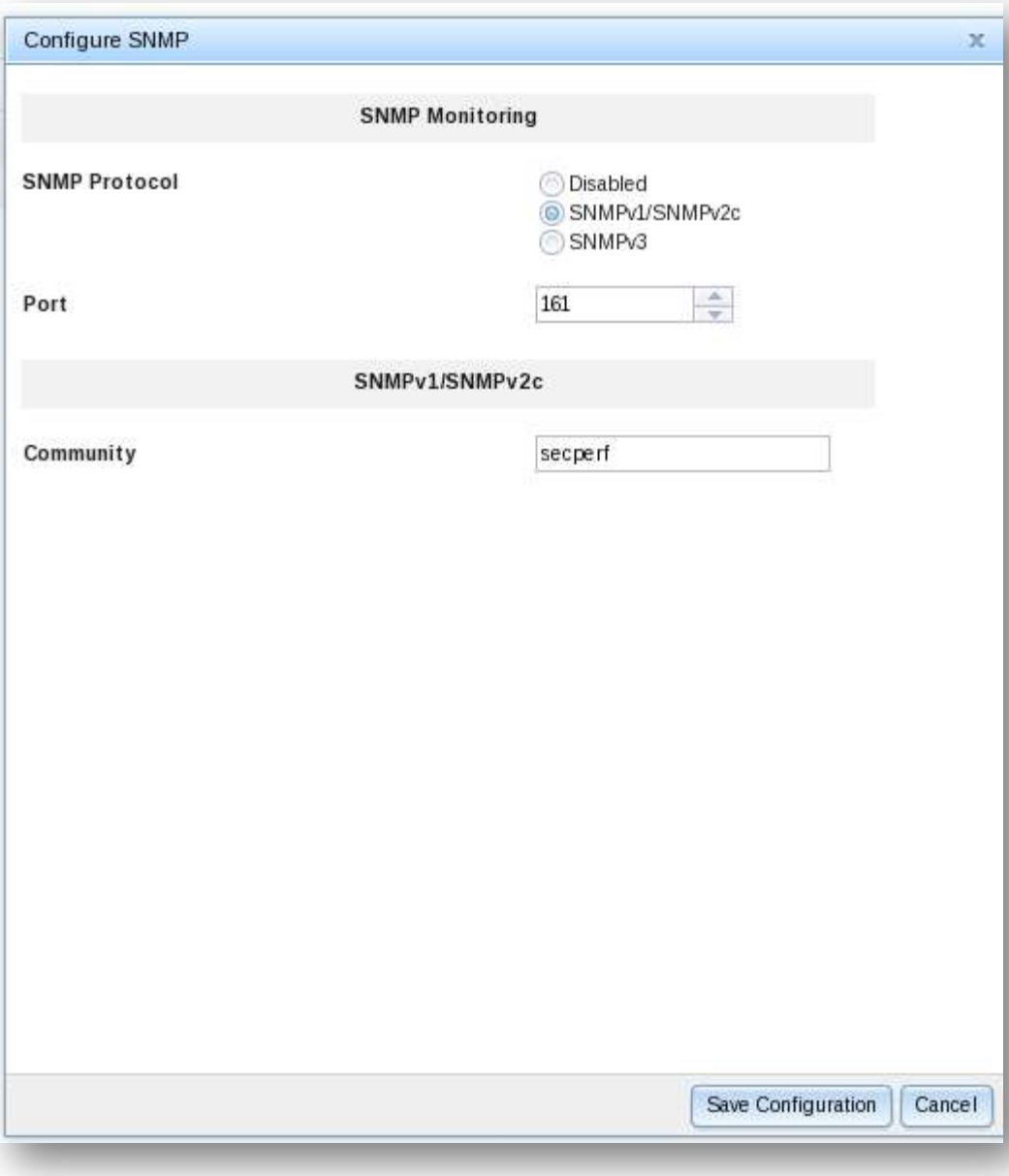


Figure 24: Set SNMP Version and Community

The SNMP processes will be started, but the Liberty processes for the VA do not require a restart. To enable alerts for SNMP, navigate to Manage → System Settings → System Audit Events.

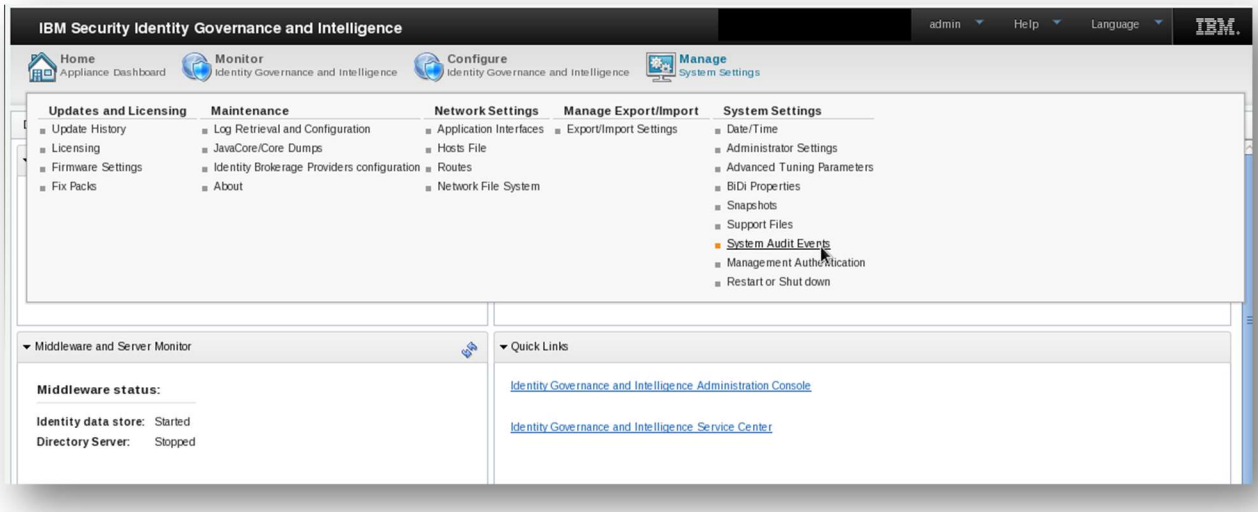


Figure 25: System Audit Events

Add an SNMP object by clicking New → SNMP.

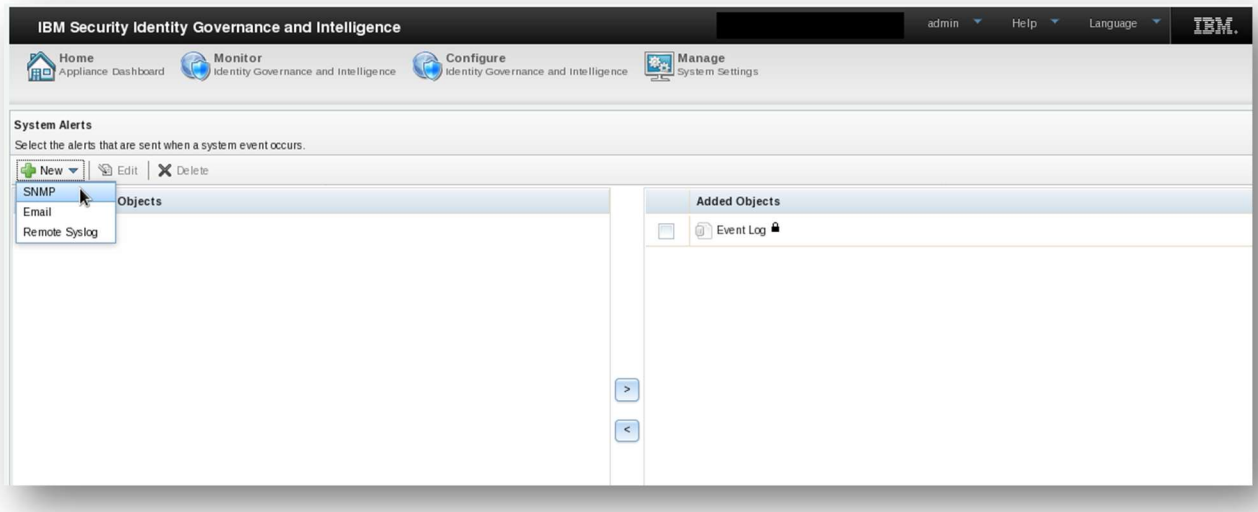


Figure 26: Add SNMP Object

A pop-up window will appear. Fill in the information to name the object, along with the version, the IP address, the port, and the community. Save the configuration.

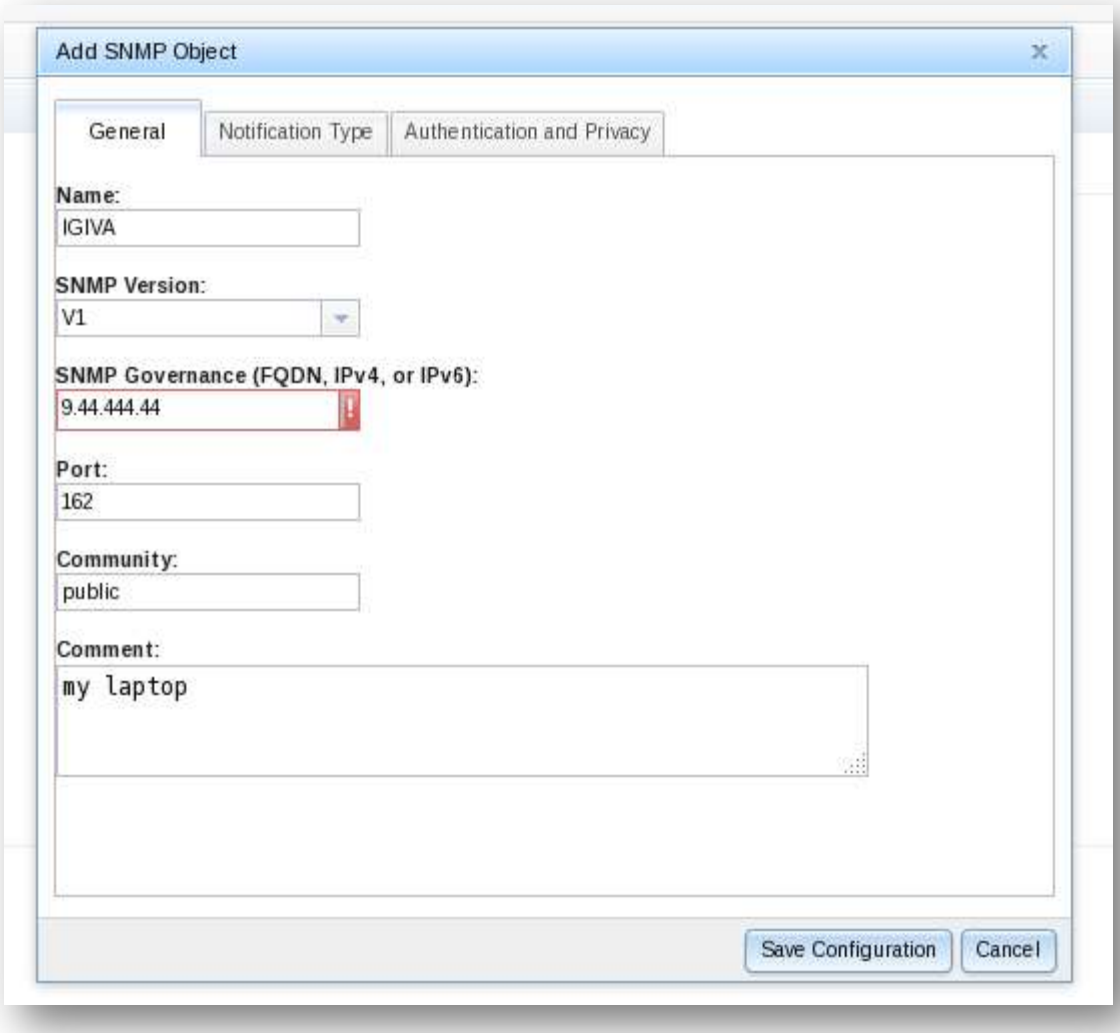


Figure 27: Configure Alerts

Use “>” button in the center of the window to add this to the Added Objects panel on the right side.

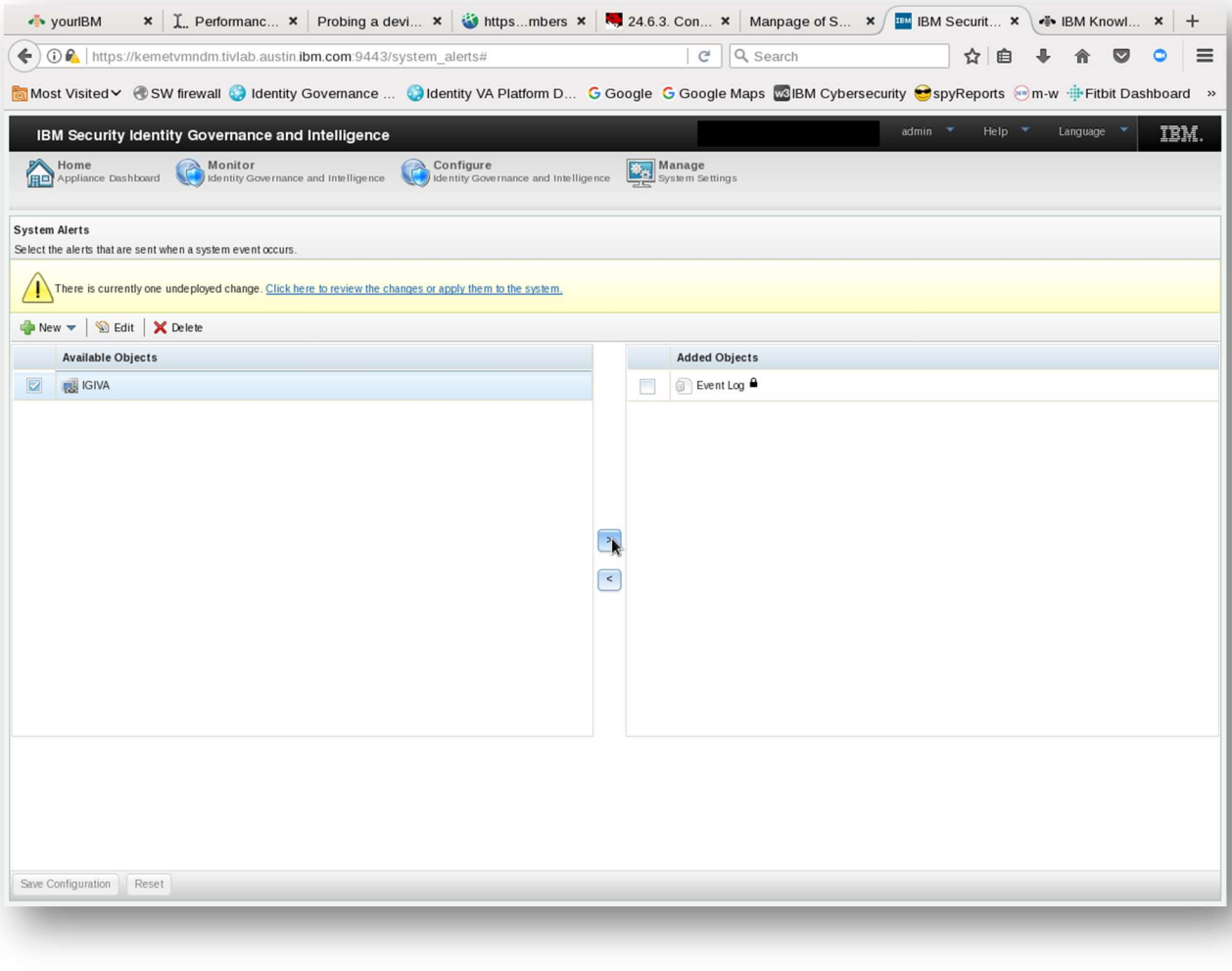


Figure 28: Deploy Changes to Alerts

Click Save Configuration. Deploy the changes.

Once the system is configured, it can be tested/accessed with a simple command such as

```
snmpwalk -v1 -c community hostname
```

To begin with, the *snmpwalk* is a good way to determine the object labels available in your VA. Redirect the output from the command above to a local file and reference it to determine the labels for polling. The values in the MIB variables are counters, so to determine input or output rates (for example), one will need two poll cycles to calculate the difference between them.

With these labels, one can then write custom scripts to poll this information directly from the system. As an example, the following command will collect CPU utilization information on the VA every 10 seconds.

```
while ((1)); do snmpwalk –v1 –c community hostname HOST-RESOURCES-
MIB::hrProcessorLoad ; sleep 10;
```

Online man pages for SNMP caution against polling too often (1 second) because this can artificially inflate the CPU utilization.

IO behavior can be collected. In the example below, inbound traffic on the network object at index 1 is polled several seconds apart with the following command.

```
snmpwalk -v1 -c community hostname | grep ifInOctets | grep "\.1 "
```

Response #1: 489831639

Response #2: 490500051

The customer will need to do the appropriate math to determine the rate of network traffic based on the polling period, the duplex setting of the network, and the speed of the network interface.

Users can also collect per-process CPU utilization statistics from the *snmpwalk* information. The MIB data will report the process index. The process almost always found at index 1 on a Unix system is **init**. Looking at a sample of the *snmpwalk* output, one can see the index of processes.

```
HOST-RESOURCES-MIB::hrSWRunIndex.1 = INTEGER: 1
HOST-RESOURCES-MIB::hrSWRunIndex.2 = INTEGER: 2
HOST-RESOURCES-MIB::hrSWRunIndex.3 = INTEGER: 3
HOST-RESOURCES-MIB::hrSWRunIndex.4 = INTEGER: 4
HOST-RESOURCES-MIB::hrSWRunIndex.5 = INTEGER: 5
HOST-RESOURCES-MIB::hrSWRunIndex.6 = INTEGER: 6
```

```
HOST-RESOURCES-MIB::hrSWRunIndex.7 = INTEGER: 7
HOST-RESOURCES-MIB::hrSWRunIndex.8 = INTEGER: 8
HOST-RESOURCES-MIB::hrSWRunIndex.9 = INTEGER: 9
HOST-RESOURCES-MIB::hrSWRunIndex.10 = INTEGER: 10
HOST-RESOURCES-MIB::hrSWRunIndex.11 = INTEGER: 11
HOST-RESOURCES-MIB::hrSWRunIndex.12 = INTEGER: 12
```

The index can be used to find the name of the process by looking at the hrSWRunName object.

```
HOST-RESOURCES-MIB::hrSWRunName.1 = STRING: "init"
HOST-RESOURCES-MIB::hrSWRunName.2 = STRING: "kthreadd"
HOST-RESOURCES-MIB::hrSWRunName.3 = STRING: "migration/0"
HOST-RESOURCES-MIB::hrSWRunName.4 = STRING: "ksoftirqd/0"
HOST-RESOURCES-MIB::hrSWRunName.5 = STRING: "migration/0"
HOST-RESOURCES-MIB::hrSWRunName.6 = STRING: "watchdog/0"
HOST-RESOURCES-MIB::hrSWRunName.7 = STRING: "migration/1"
HOST-RESOURCES-MIB::hrSWRunName.8 = STRING: "migration/1"
HOST-RESOURCES-MIB::hrSWRunName.9 = STRING: "ksoftirqd/1"
HOST-RESOURCES-MIB::hrSWRunName.10 = STRING: "watchdog/1"
HOST-RESOURCES-MIB::hrSWRunName.11 = STRING: "migration/2"
HOST-RESOURCES-MIB::hrSWRunName.12 = STRING: "migration/2"
```

The following command can be run a loop to collect the utilization of a single process over time. In this example, the process being examined is the **init** process (index 1).

```
snmpwalk -v1 -c community hostname | grep HOST-RESOURCES-MIB::hrSW | grep
"\.1 "
```

This is the output.

```
HOST-RESOURCES-MIB::hrSWRunIndex.1 = INTEGER: 1
HOST-RESOURCES-MIB::hrSWRunName.1 = STRING: "init"
```

```
HOST-RESOURCES-MIB::hrSWRunID.1 = OID: SNMPv2-SMI::zeroDotZero
HOST-RESOURCES-MIB::hrSWRunPath.1 = STRING: "init"
HOST-RESOURCES-MIB::hrSWRunParameters.1 = ""
HOST-RESOURCES-MIB::hrSWRunType.1 = INTEGER: application(4)
HOST-RESOURCES-MIB::hrSWRunStatus.1 = INTEGER: runnable(2)
HOST-RESOURCES-MIB::hrSWRunPerfCPU.1 = INTEGER: 111
HOST-RESOURCES-MIB::hrSWRunPerfMem.1 = INTEGER: 856 KBytes
```

In this example, the **init** process took 111 centi-seconds of the total system's CPU resources. Although memory utilization is available through the VA Administration Dashboard, the user might find the SNMP information useful for per-process memory utilization statistics as well.

Refer to the online documentation and man page(s) for definitions of the fields and labels, additional examples, and help for SNMP.

19. DB Connection Pool

The default number of connections per resource for the Identity Governance product is 30. For Enterprise Environments, medium sized environments with many targets, or environments expecting many concurrent user logins or change password by self care, this default setting may not be adequate. The customer may experience gradually deteriorating UI performance and/or sluggish response times. To determine if these problems are due to a low connection setting, the user should look at the WebSphere Liberty logs for the Identity Governance application server. Download the support files from Administration Console → Manage → System Settings → Support Files.

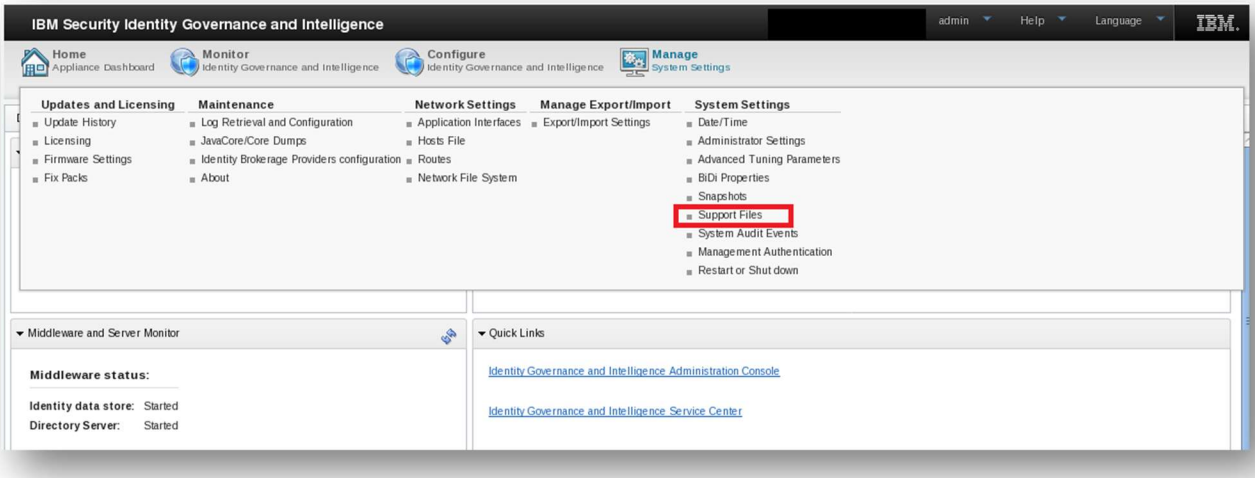


Figure 29: Downloading Support Files

In the log file at *download_dir*/opt/ibm/wlp/usr/servers/igi/logs/messages.log, look for errors associated with connections not being available.

```
00000066 SystemErr          R       java.sql.SQLTransientConnectionException:
Connection not available, Timed out waiting for 180000
```

Such messages can also be found in *download_dir*/opt/ibm/wlp/usr/servers/igi/logs/ffdc in the exception summaries.

```
JST com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException Max connections
reached 869
```

When there are no connections available, the Identity Governance core operations will also see errors. For example, event processing may encounter an error such as the one below.

```
ERROR AGC:? - openConnection failed

org.hibernate.exception.JDBCConnectionException: Cannot open connection
```

There are several ways to monitor the DB connections on the database tier. A DB snapshot is one method to monitor the connections and the status. As an example, in a DB2 snapshot search for the phrase “High water mark for connections”

```
# grep "High water mark for connections" ISIG-snapshot*.out
ISIG-snapshot1.out:High water mark for connections            = 31
ISIG-snapshot2.out:High water mark for connections            = 50
```

A new API has been introduced to Version 5.2.5 that allows Liberty connection monitoring. This is enabled by default in the application server and in the Brokerage server, and laboratory tests show there is no performance impact. While there is no API to enable/disable, the customer can use the CLI to modify or check the status, igi → monitoring → connection pool. This is a node specific feature and needs to be enabled separately for each node in a cluster. This value will not be synced during a cluster node sync operation. When a VA is upgraded, the node will retain the old connection monitoring state.

Figure 30: Connection Pool Monitoring API

To avoid connection failures, the DB connection pool can be increased. From the Administration Dashboard → Configure → Database Server Connection. Click the radio button next to the current Identity data store and click Reconfigure. In the pop-up window, go to the Connection Pool tab and edit the maximum number of connections. The Identity Governance application server must be restarted for the new connection setting to go into effect. A setting of 50 is a good first step to reducing connection errors.

20. Multi-threaded Enterprise Connector

The ability to create multi-threaded enterprise connectors was added to the Governance product in Version 5.2.3. When an enterprise connector is created, the administrator may now specify the number of threads which will process the events for a given endpoint target. The default number of threads is 3, but this value can be tuned to a value in the range 1 – 10. To set this value, the administrator can edit the existing configuration of an enterprise connector or set it when the connector is newly created.

To set this value for an existing connector, the administrator must stop the connector, Enterprise Connectors (EC) → Manage → Connectors. Choose the connector from the left panel, then open the Global Config section in the right panel.

The attribute is called the “WRITE_TO_THREADS_NUMBER”. When a connector is created, this value is not surfaced in the configuration.

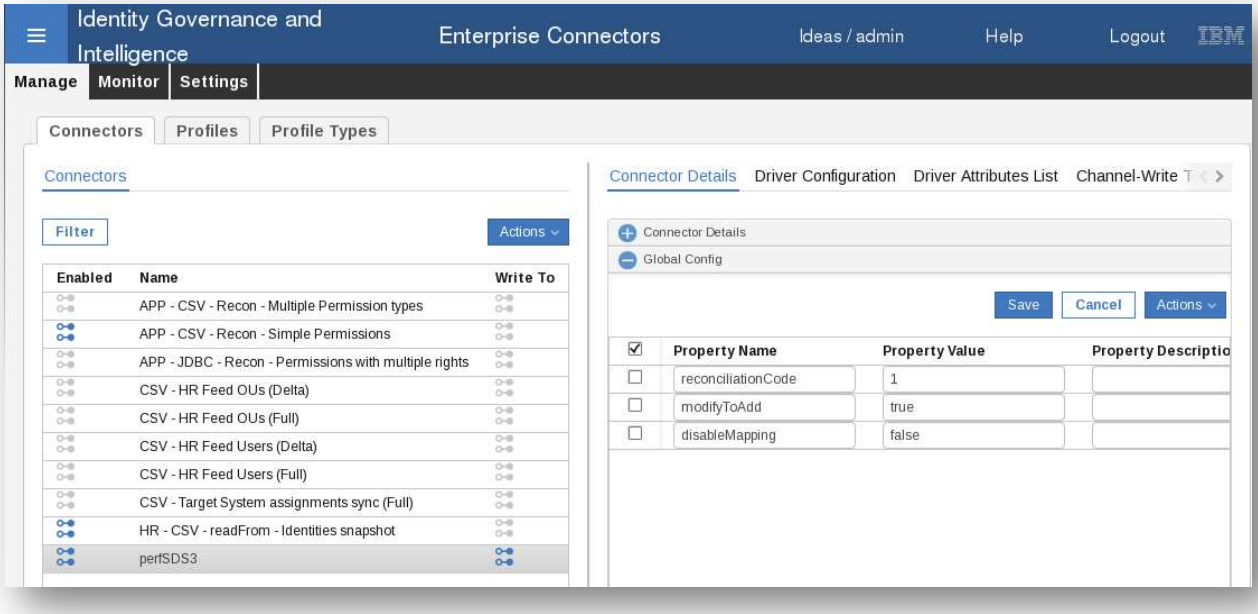


Figure 31: Global Configuration of Enterprise Connector

If the administrator wishes to alter this property, choose Actions → Add in the right panel. An empty field will be added to the panel. Enter WRITE_TO_THREADS_NUMBER in the *Property Name* field, the number of threads to set in the *Property Value* field, and an optional description in the *Property Description* field. Click Save.

In the same way, when a connector is newly created, this value can be set in the initial configuration. There is no need to stop/start/restart the Governance application server.

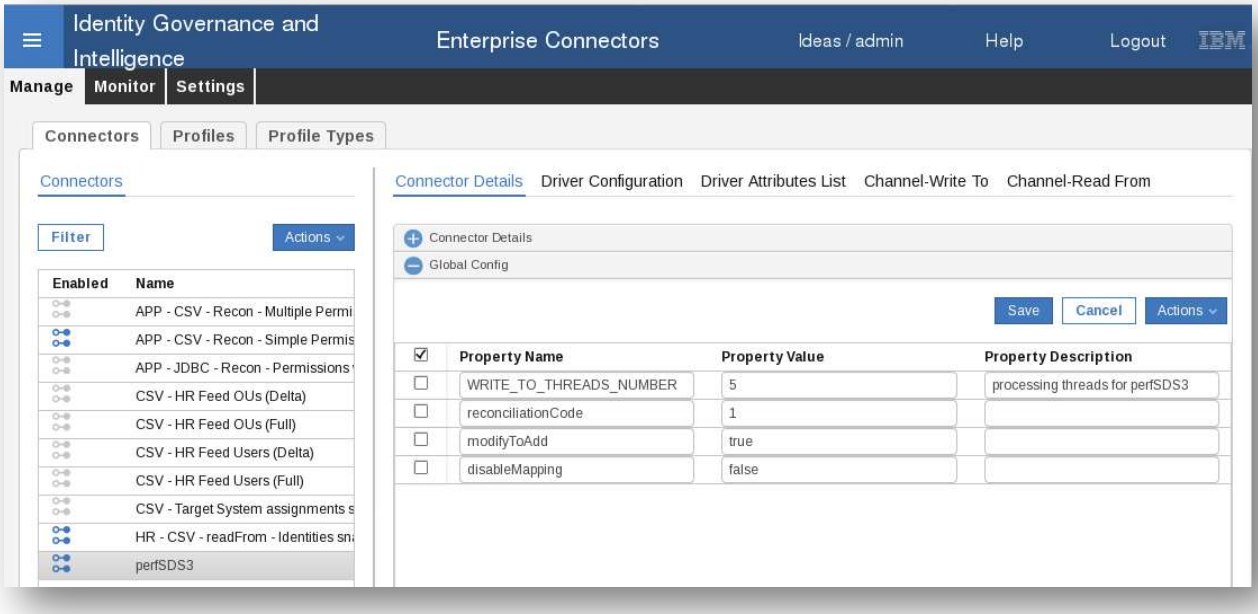


Figure 32: Setting WRITE_TO_THREADS_NUMBER

Although the default configuration already contains multi-threading (3), the concurrency can be increased for the processing of the events in the OUT queue. The tests used to demonstrate this feature separated the event creation from the event processing. Under ordinary (default) circumstances, events are created in the OUT queue while the Enterprise Connector threads are processing them (fulfillment). For the purposes of this test, and to demonstrate the precise effect of this feature, the two phases were separated into event creation and event processing. In laboratory tests with 3, 5, 7, and 9 threads, the event processing showed increasing raw throughput, and increased overall system efficiency (DB + Governance VA). For the system under test (laboratory environment), the peak for both throughput and efficiency was 9 threads.

Obviously, the DB and VA will see increased CPU utilization when the thread concurrency is increased. In the laboratory, going from 3 to 5 threads increased the both DB and VA CPU utilization by 3-4%. Going from 5 to 7 threads increased the DB CPU utilization by an additional 20+%. The VA CPU increased by only 7%. The utilization is effective flat for both the DB and VA going from 7 to 9 threads, with a slight raw throughput increase at 9 threads.

When invoking this feature in a production environment, there are three things to remember. The first consideration is a default processing environment will involve event creation and event processing occurring at the same time. CPU resources will need to be split across the two. Setting WRITE_TO_THREADS_NUMBER too high may choke the CPU or overrun the event

creation process. If there are no events to process, the multiple threads of the connector will go to waste. The second consideration is for the other background tasks that may need CPU resources. Care should be taken; tune this setting slowly to avoid CPU starvation for other tasks within the VA and those which need DB services. Lastly, faster event processing will cause the endpoint target and the TDI (if an external TDI is used) to see increased CPU utilization. Although these increases were small in the laboratory environment, this may not be the case in the production environment where those machines/services might be housed on shared hardware where CPU is a premium resource.

21. Tcpdump

Support was added in Version 5.2.4 for collecting tcpdump results from the Governance appliance. The tool is available via the CLI tools → packet_tracing and is not enabled by default. Below is an example of accessing the tool and running a trace.

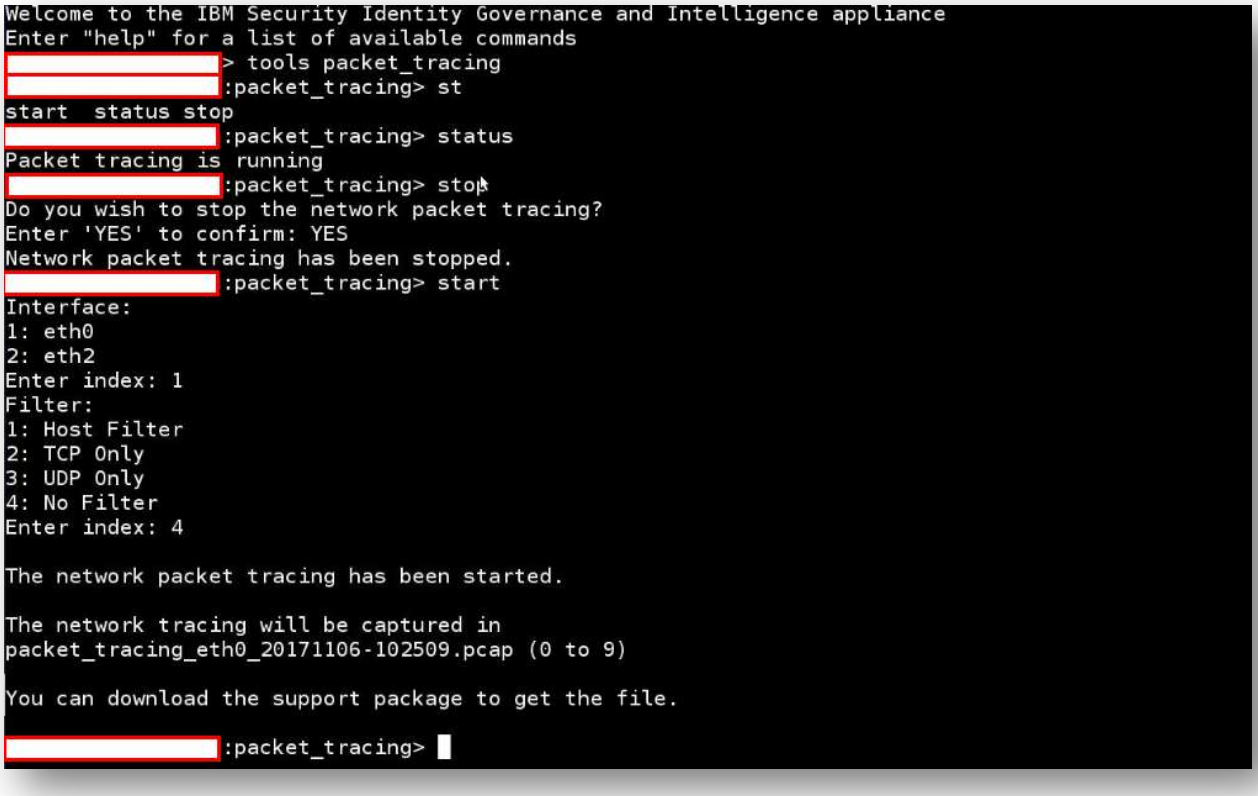


Figure 33: Tcpdump Example

22. Increasing the Heap Size

Version 5.2.4 provides a mechanism to increase the heap size to a max of 8GB. This step might be necessary when running large bulk loads, or launching many or particularly large hierarchies at the same time, etc. If the VA reports an error message that the heap has been exhausted, the following procedure can be used to increase the heap. From the CLI navigate to igi → jvm_heapsize → set_max_heapsize. The menu will offer an option to change the IGI application server heap or the Broker application server heap. The user may choose any size between 4096 MB and 8192 MB. A restart of the application server is necessary. Up to 5.2.6 Version, for configurations with large numbers of users, 500K+, it is advisable to set the heap to 8GB to ensure the heap is not exhausted during report generation. In 5.2.6.1 this mechanism has improved and in configurations with about 700K users , the default 4092MB has safe to run reports.

23. Resetting a Connector and Clearing Brokerage Data

As of Version 5.2.4, the Governance product offers a mechanism to reset a connector and clear the brokerage data for the target application. Refer to the Knowledge Center for prerequisites and instructions. As stated in the Knowledge Center, this operation should be a last resort, and there are performance implications of using this procedure. The coherency between the Governance VA and the target must undergo a synchronization to align their states. This may result in a repopulation of the cached data which means CPU consumption on both the VA and the DB tier. Depending on the amount of data to be repopulated, this resynchronization could take a significant amount of time and should be scheduled during periods of low activity.

Prior to starting this procedure, data analytics should be turned off (Risk Scans, Role Mining, etc.) and resumed after the operation is complete. Additionally, it is advisable that database optimization techniques like *runstats* and *db2rbind* be executed when the operation is complete to ensure statistics are updated.

24. Deadlocking on Foreign Key Constraints

During testing of Version 5.2.4, the laboratory environment encountered a database deadlock during a performance test of a hierarchy build. Analysis of the condition revealed the cause to be foreign key constraint definitions that had no obvious function. To avoid this situation, the performance labs dropped all foreign key constraints using the following DB commands. Since 5.2.5 onwards it’s no longer needed to do these actions

```
ALTER TABLE IGAQRZ.QRZ1_BLOB_TRIGGERS DROP FOREIGN KEY QRZ1_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ1_CRON_TRIGGERS DROP FOREIGN KEY QRZ1_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ1_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ1_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ1_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ1_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ1_TRIGGERS DROP FOREIGN KEY QRZ1_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ2_BLOB_TRIGGERS DROP FOREIGN KEY QRZ2_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ2_CRON_TRIGGERS DROP FOREIGN KEY QRZ2_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ2_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ2_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ2_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ2_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ2_TRIGGERS DROP FOREIGN KEY QRZ2_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ3_BLOB_TRIGGERS DROP FOREIGN KEY QRZ3_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ3_CRON_TRIGGERS DROP FOREIGN KEY QRZ3_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ3_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ3_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ3_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ3_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ3_TRIGGERS DROP FOREIGN KEY QRZ3_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ4_BLOB_TRIGGERS DROP FOREIGN KEY QRZ4_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ4_CRON_TRIGGERS DROP FOREIGN KEY QRZ4_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ4_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ4_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ4_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ4_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ4_TRIGGERS DROP FOREIGN KEY QRZ4_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ5_BLOB_TRIGGERS DROP FOREIGN KEY QRZ5_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ5_CRON_TRIGGERS DROP FOREIGN KEY QRZ5_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ5_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ5_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ5_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ5_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ5_TRIGGERS DROP FOREIGN KEY QRZ5_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ6_BLOB_TRIGGERS DROP FOREIGN KEY QRZ6_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ6_CRON_TRIGGERS DROP FOREIGN KEY QRZ6_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ6_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ6_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ6_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ6_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ6_TRIGGERS DROP FOREIGN KEY QRZ6_TRIGGER_TO_JOBS_FK;
ALTER TABLE IGAQRZ.QRZ7_BLOB_TRIGGERS DROP FOREIGN KEY QRZ7_BLOB_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ7_CRON_TRIGGERS DROP FOREIGN KEY QRZ7_CRON_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ7_SIMPLE_TRIGGERS DROP FOREIGN KEY QRZ7_SIMPLE_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ7_SIMPROP_TRIGGERS DROP FOREIGN KEY QRZ7_SIMPROP_TRIG_TO_TRIG_FK;
ALTER TABLE IGAQRZ.QRZ7_TRIGGERS DROP FOREIGN KEY QRZ7_TRIGGER_TO_JOBS_FK;
```

Performance is not affected by dropping the constraints, but the deadlocks disappeared from the hierarchy tests.

25. Admin Dashboard Monitoring of CPU and Memory

In Version 5.2.5, the Administration Dashboard contains an enhanced view for determining which part of the CPU or memory consumption is due to the Identity Governance processes versus the Identity Brokerage processes. Navigating to Monitor → Monitoring → CPU, for example, the administrator is presented with a view of the CPU components for a selected Date Range. The choices for the Date Range are 1, 3, 7, and 30 days.

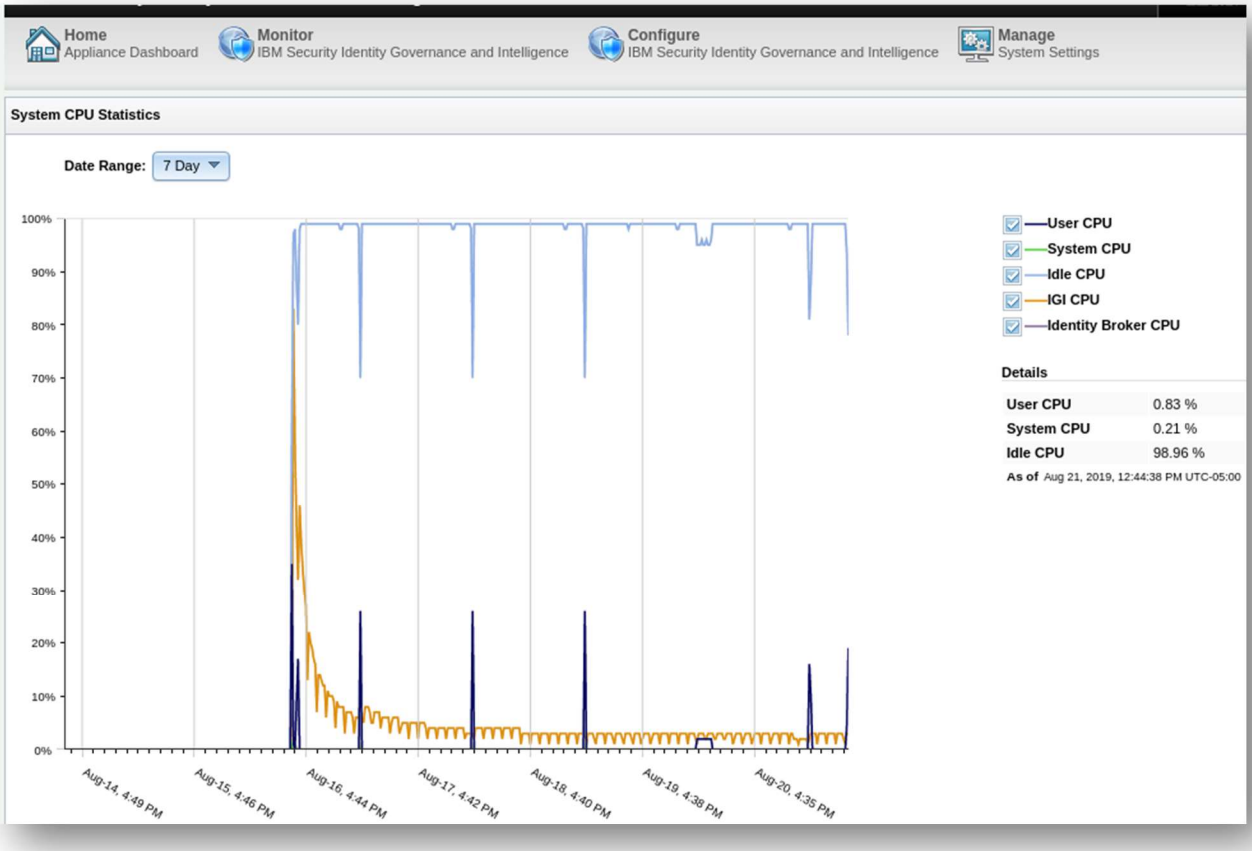


Figure 34: CPU Monitoring of VA

The memory view also includes information about the current max heap size for each application server.

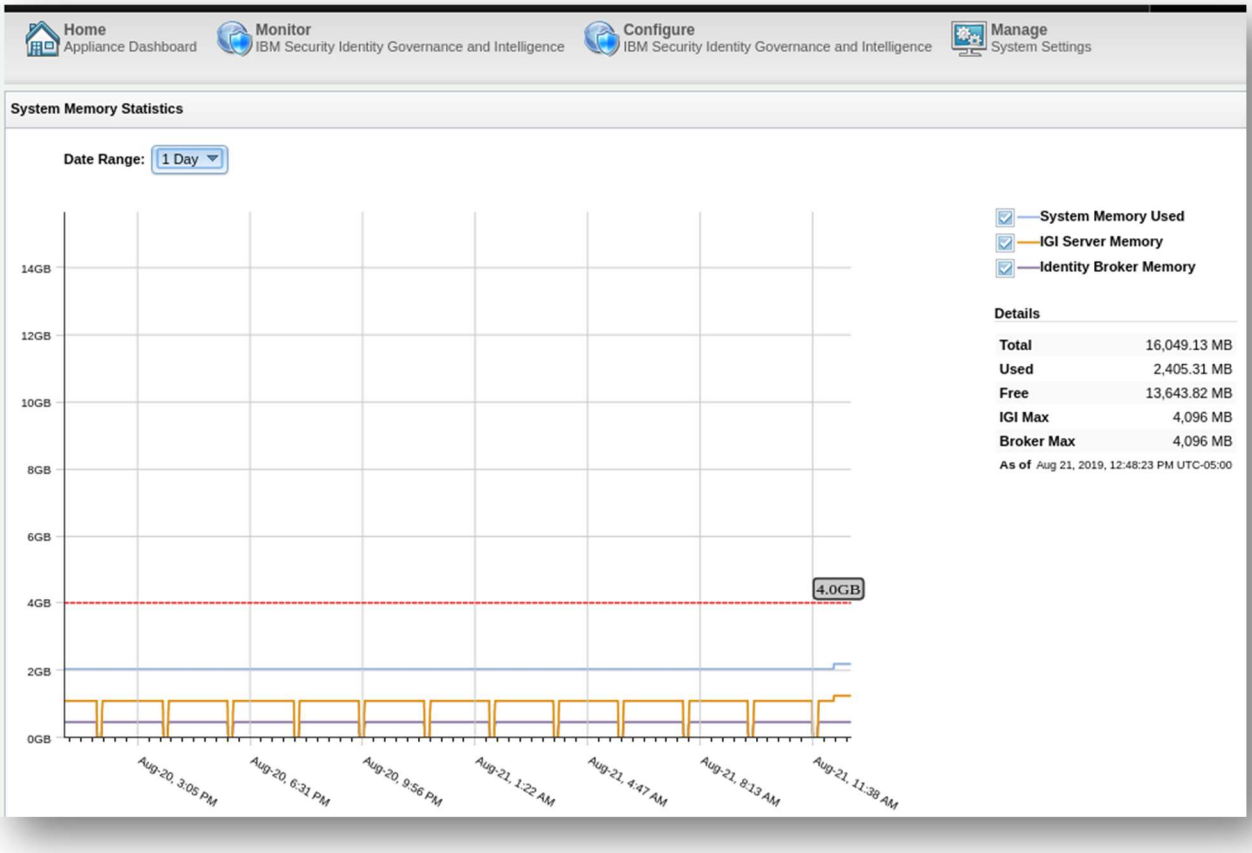


Figure 35: Memory Monitoring of VA

26. Indexes on Foreign Keys

Indexes on foreign keys improve performance during removal operations and result in reduced locking on the DB.

A new set of indexes have been identified and can be created up to 5.2.6 GA version on the IGACORE schema using the scripts reported in the attached file. These indexed can be create by connecting to the DB with the IGACORE user. Since 5.2.6 fp1 onwards it's no longer needed to do these actions

CREATE_INDEXES_IG
ACORE.sql

The script contains the following indexes:

```
CREATE INDEX IGACORE.IDX_AR_REV_PWDCFG_FK_FPIDX ON IGACORE.ACCOUNT_REVIEW ("PWDCFG")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_AR_REV_REVIEWED_BY_FK ON IGACORE.ACCOUNT_REVIEW ("REVIEWED_BY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_AR_REVIEWER_FIRSTOWNER_FK ON IGACORE.ACCOUNT_REVIEWER
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_AR_REVIEWER_SECONDOWNER_FK ON IGACORE.ACCOUNT_REVIEWER
("CERT_SECOND_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ACCOUNT_REV_HIS_PERSON_FK ON IGACORE.ACCOUNT_REVIEW_H ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ACCOUNT_REV_HIS_PWDCFG_FK ON IGACORE.ACCOUNT_REVIEW_H ("PWDCFG")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ACCOUNT_REV_HIS_REVIEWED_BY_FK ON IGACORE.ACCOUNT_REVIEW_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARNOTE_ACCOUNT_REVIEWER_FK ON IGACORE.ACCOUNT_REVIEW_NOTE
("ACCOUNT_REVIEWER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARNOTE_PERSON_FROM_FK ON IGACORE.ACCOUNT_REVIEW_NOTE ("PERSON_FROM")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARNOTE_TO_FK ON IGACORE.ACCOUNT_REVIEW_NOTE ("PERSON_TO")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ADDITIONAL_DATA_REQUEST_FK ON IGACORE.ADDITIONAL_DATA ("REQUEST")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_APPLICATION_ENT_EXT_TYPE_FK ON IGACORE.APPLICATION ("SOD_EXT_LEVEL")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_APPLICATION_FLOW_FK ON IGACORE.APPLICATION ("FLOW")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_APPLICATIONS_REQUEST_FK ON IGACORE.APPLICATIONS ("REQUEST")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARM_AUTHORIZATIONS_REQUEST_FK ON IGACORE.ARM_AUTHORIZATIONS ("REQUEST")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARM_AUTH_SU_RED_FK ON IGACORE.ARM_AUTH_S_USER ("REDIRECTED_BY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARM_AUTH_SU_RED_REP_FK ON IGACORE.ARM_AUTH_S_USER ("RED_REPRESENTED_BY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARM_AUTH_SU_REP_FK ON IGACORE.ARM_AUTH_S_USER ("REPRESENTED_BY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ARM_I18N_MESSAGE_LOC_CODE_FK ON IGACORE.ARM_I18N_MESSAGE
("LOCALIZATION_CODE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATT_DEF_REVIEWER_FK ON IGACORE.ATTESTATION ("DEF_REVIEWER")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_ATT_SUPERVISOR_FK ON IGACORE.ATTESTATION ("SUPERVISOR")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATTBL_USER_ATTESTATION_FK ON IGACORE.ATTESTATION_BLACKLIST
("ATTESTATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATTBL_USER_PERSON_FK ON IGACORE.ATTESTATION_BLACKLIST ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATTPROP_ATTESTATION_FK ON IGACORE.ATTESTATION_PROP ("ATTESTATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATTVIEW_ATTESTATION_FK ON IGACORE.ATTESTATION_VIEW ("ATTESTATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ATTR_PERM_MAPPING_TARGET_ID_FK ON IGACORE.ATTR_PERMISSION_MAPPING
("TARGET_ID")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_FIX_APP_SCOPE_FK ON IGACORE.CFG_FIXED_APPLICATION_SCOPE
("ACTIVITY_ID")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FIXED_REPRSCOPE_FK ON IGACORE.CFG_FIXED_REPRESENTED_SCOPE
("ACTIVITY_ID")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_USR_SCOPE_ACTID_FK ON IGACORE.CFG_FIXED_USER_SCOPE ("ACTIVITY_ID")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_NOTIF_EMS_TEMPLATE_FK ON IGACORE.CFG_NOTIFICATION ("EMS_TEMPLATE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PRIORITY_NOTIF_EMAIL_FK ON IGACORE.CFG_PRIORITY
("CFG_NOTIFICATION_EMAIL")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PRIORITY_NOTIF_SMS_FK ON IGACORE.CFG_PRIORITY
("CFG_NOTIFICATION_SMS")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCESS_EXPIRATION_WF_FK ON IGACORE.CFG_PROCESS ("EXPIRATION_WF")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCESS_NOTIFICATION_FK ON IGACORE.CFG_PROCESS ("CFG_NOTIFICATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCESS_NOTIF_SMS_FK ON IGACORE.CFG_PROCESS ("CFG_NOTIFICATION_SMS")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCESS_PRIORITY_FK ON IGACORE.CFG_PROCESS ("CFG_PRIORITY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCACT_ESC_PROCESS_FK ON IGACORE.CFG_PROCESSACTIVITY
("ESCALATION_PROCESS")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROCACT_LINK_PROCESS_FK ON IGACORE.CFG_PROCESSACTIVITY
("LINKED_PROCESS")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_ACTION_FK ON IGACORE.CFG_PROCESSACTIVITY_ACTION
("CFG_PROCESSACTIVITY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CHANGELOG_PWDMANAGEMENT_FK ON IGACORE.CHANGELOG ("PWDMANAGEMENT")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CLASS_NAME_TYPE_FK ON IGACORE.CLASS_NAME ("TYPE")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_COMPARE_LOG_PARENT_ID_FK ON IGACORE.COMPARE_LOG ("PARENT_ID")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CONNECTOR_HISTORY_CONNECTOR_FK ON IGACORE.CONNECTOR_HISTORY
("CONNECTOR")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_CONN_PROPERTY_CONNECTOR_FK ON IGACORE.CONNECTOR_PROPERTY ("CONNECTOR")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_COUNTER_LOG_EXECUTION_LOG_FK ON IGACORE.COUNTER_LOG ("EXECUTION_LOG")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_DOMAIN_PROFILE_DOMAIN_FK ON IGACORE.DOMAIN_PROFILE ("ENVIRONMENT",
"DOMAIN")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_DRIVER_CLASS_NAME_FK ON IGACORE.DRIVER ("CLASS_NAME")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_DRIVER_HIER_CHILD_FK ON IGACORE.DRIVER_HIER ("CHILD")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_DRIVER_HIER_PARENT_FK ON IGACORE.DRIVER_HIER ("PARENT")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_DRIVER_PROPERTY_DRIVER_FK ON IGACORE.DRIVER_PROPERTY ("DRIVER")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMPLOYMENT_HIERARCHY_FK ON IGACORE.EMPLOYMENT ("HIERARCHY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_ENTITLEMENT_FK ON IGACORE.EMPLOYMENT_REVIEW ("ENTITLEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_GROUP_FK ON IGACORE.EMPLOYMENT_REVIEW ("ORGANIZATIONAL_UNIT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_REVIEWED_BY_FK ON IGACORE.EMPLOYMENT_REVIEW ("REVIEWED_BY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REVIEWER_FIRSTOWNER_FK ON IGACORE.EMPLOYMENT_REVIEWER
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REVIEWER_SECONDOWNER_FK ON IGACORE.EMPLOYMENT_REVIEWER
("CERT_SECOND_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_HIS_ENTITLEMENT_FK ON IGACORE.EMPLOYMENT_REVIEW_H
("ENTITLEMENT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_HIS_PERSON_FK ON IGACORE.EMPLOYMENT_REVIEW_H ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_HIS_REVIEWED_BY_FK ON IGACORE.EMPLOYMENT_REVIEW_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ERNOTE_EMPLOYMENT_REVIEWER_FK ON IGACORE.EMPLOYMENT_REVIEW_NOTE
("EMPLOYMENT_REVIEWER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ERNOTE_PERSON_FROM_FK ON IGACORE.EMPLOYMENT_REVIEW_NOTE ("PERSON_FROM")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ERNOTE_TO_FK ON IGACORE.EMPLOYMENT_REVIEW_NOTE ("PERSON_TO")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMPREVR_EMPREV_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT
("EMPLOYMENT_REVIEW")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMPREVR_PERMISSION_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT ("PERMISSION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMPREVR_PERSON_SERVICE_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT
("PERSON_SERVICE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMPREVR_REVIEWED_BY_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT
("REVIEWED_BY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_EMP_REV_R_H_ENTITLEMENT_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT_H
("ENTITLEMENT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_R_H_PERSON_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT_H ("PERSON")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_R_H_PER_SERV_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT_H
("PERSON_SERVICE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMP_REV_R_H_REVIEWED_BY_FK ON IGACORE.EMPLOYMENT_REVIEW_RIGHT_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMS_I18N_LANGUAGE_FK ON IGACORE.EMS_I18N_MESSAGE ("LANGUAGE")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMS_TEMPLATE_ALTERNATIVE_FK ON IGACORE.EMS_TEMPLATE ("ALTERNATIVE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMS_TEMPLATE_BODY_FK ON IGACORE.EMS_TEMPLATE ("BODY")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMS_TEMPLATE_SUBJECT_FK ON IGACORE.EMS_TEMPLATE ("SUBJECT")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EMS_TEMPLATE_TEMPLATE_TYPE_FK ON IGACORE.EMS_TEMPLATE ("TEMPLATE_TYPE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_APPLICATION_FK ON IGACORE.ENTITLEMENT ("APPLICATION")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_ENT_FAMILY_FK ON IGACORE.ENTITLEMENT ("FAMILY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_FLOW_AUTH_FK ON IGACORE.ENTITLEMENT ("FLOW_AUTH")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_FLOW_CHECK_FK ON IGACORE.ENTITLEMENT ("FLOW_CHECK")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_OWNER_FK ON IGACORE.ENTITLEMENT ("OWNER")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_PROFILE_TYPE_FK ON IGACORE.ENTITLEMENT ("PROFILE_TYPE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_F_HIER_CHILD_APP_FK ON IGACORE.ENTITLEMENT_FLAT_HIER
("CHILD_APPLICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_F_HIER_CHILD_PROF_TYPE_FK ON IGACORE.ENTITLEMENT_FLAT_HIER
("CHILD_PROFILE_TYPE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_F_HIER_PARENT_APP_FK ON IGACORE.ENTITLEMENT_FLAT_HIER
("PARENT_APPLICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_F_HIER_PARENT_PROF_TYPE_FK ON IGACORE.ENTITLEMENT_FLAT_HIER
("PARENT_PROFILE_TYPE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_INCOMP_ENT_FK ON IGACORE.ENTITLEMENT_INCOMP ("ENTITLEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_REV_REVIEWED_BY_FK ON IGACORE.ENTITLEMENT_REVIEW ("REVIEWED_BY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_REVIEWER_FIRSTOWNER_FK ON IGACORE.ENTITLEMENT_REVIEWER
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_REVIEWER_SECONDOWNER_FK ON IGACORE.ENTITLEMENT_REVIEWER
("CERT_SECOND_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_ENT_REV_HIS_ENTITLEMENT_FK ON IGACORE.ENTITLEMENT_REVIEW_H
("ENTITLEMENT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENT_REV_HIS_REVIEWED_BY_FK ON IGACORE.ENTITLEMENT_REVIEW_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTNOTE_ENT_REVIEWER_FK ON IGACORE.ENTITLEMENT_REVIEW_NOTE
("ENTITLEMENT_REVIEWER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTNOTE_PERSON_FROM_FK ON IGACORE.ENTITLEMENT_REVIEW_NOTE
("PERSON_FROM")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTNOTE_TO_FK ON IGACORE.ENTITLEMENT_REVIEW_NOTE ("PERSON_TO")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ENTITLEMENT_SERV_SERV_ATTR_FK ON IGACORE.ENTITLEMENT_SERVICE
("SERVICE_ATTRIBUTE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_EXTMAPPING_LOCALIZATION_FK ON IGACORE.EXTERNALMAPPING ("LOCALIZATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FLOW_IDEAS_MODULE_FK ON IGACORE.FLOW ("IDEAS_MODULE")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FLOW_ENTITY_IDEAS_MODULE_FK ON IGACORE.FLOW_ENTITY ("IDEAS_MODULE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FLOW_EV_TYPE_IDEAS_MODULE_FK ON IGACORE.FLOW_EVENT_TYPE ("IDEAS_MODULE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FLOW_RULE_IDEAS_MODULE_FK ON IGACORE.FLOW_RULE ("IDEAS_MODULE")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_FLOW_STD_CONF_IDEAS_MODULE_FK ON IGACORE.FLOW_STD_CONF ("IDEAS_MODULE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_GROUP_PERSON_HIERARCHY_FK ON IGACORE.GROUP_PERSON ("HIERARCHY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_HIERARCHY_FLOW_FK ON IGACORE.HIERARCHY ("FLOW")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_HAPH_PWDCFG_ATTR_KEY_FK ON IGACORE.HR_ACCOUNT_PHOLDER
("PWDCFG_ATTR_KEY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_HAPH_PWDCFG_FK ON IGACORE.HR_ACCOUNT_PHOLDER ("PWDCFG")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_HPC_PERSON_FK ON IGACORE.HR_PHOLDER_CHANGES ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_HPV_PERSON_FK ON IGACORE.HR_PHOLDER_VALUES ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_INT_RES_REQUEST_FK ON IGACORE.INT_RESOURCES ("REQUEST")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_INT_RES_AUTH_RISK_FK ON IGACORE.INT_RES_AUTH ("ENVIRONMENT", "RISK")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_INT_RES_AUTH_TASK_FK ON IGACORE.INT_RES_AUTH ("ENVIRONMENT", "TASK")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JOB_IDEAS_MODULE_FK ON IGACORE.JOB ("IDEAS_MODULE")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JOB_PROPERTY_IDEAS_MODULE_FK ON IGACORE.JOB_PROPERTY ("IDEAS_MODULE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_JOB_PROPERTY_EXT_JOB_PROP_FK ON IGACORE.JOB_PROPERTY_EXT
("JOB_PROPERTY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JOB_PROP_EXT_IDEAS_MODULE_FK ON IGACORE.JOB_PROPERTY_EXT
("IDEAS_MODULE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JOB_UNIT_HIERARCHY_FK ON IGACORE.JOB_UNIT ("HIERARCHY")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REV_ATTESTATION_FK ON IGACORE.JOB_UNIT_REVIEW ("ATTESTATION")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REV_REVIEWED_BY_FK ON IGACORE.JOB_UNIT_REVIEW ("REVIEWED_BY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REVIEWER_FIRSTOWNER_FK ON IGACORE.JOB_UNIT_REVIEWER
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REVIEWER_SECONDOWNER_FK ON IGACORE.JOB_UNIT_REVIEWER
("CERT_SECOND_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REV_HIS_ENTITLEMENT_FK ON IGACORE.JOB_UNIT_REVIEW_H ("ENTITLEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REV_HIS_ORG_UNIT_FK ON IGACORE.JOB_UNIT_REVIEW_H
("ORGANIZATIONAL_UNIT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JU_REV_HIS_REVIEWED_BY_FK ON IGACORE.JOB_UNIT_REVIEW_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JURNOTE_JOB_UNIT_REVIEWER_FK ON IGACORE.JOB_UNIT_REVIEW_NOTE
("JOB_UNIT_REVIEWER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JURNOTE_PERSON_FROM_FK ON IGACORE.JOB_UNIT_REVIEW_NOTE ("PERSON_FROM")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_JURNOTE_TO_FK ON IGACORE.JOB_UNIT_REVIEW_NOTE ("PERSON_TO")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MAPPING_FUNCTION_FK ON IGACORE.MAPPING ("FUNCTION")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MAPPING_MAPPING_OBJ_CLASS_FK ON IGACORE.MAPPING ("MAPPING_OBJ_CLASS")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MPS_FIELD_FK ON IGACORE.MAPPING_SOURCE_FIELD ("SOURCE_FIELD")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MPS_MAPPING_FK ON IGACORE.MAPPING_SOURCE_FIELD ("MAPPING")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_CFG_PROACTIVITY_FK ON IGACORE.MN_LINK ("CFG_PROCESSACTIVITY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MN_LINK_CFG_ACTIVITY_FK ON IGACORE.MN_LINK ("CFG_ACTIVITY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MN_LINK_MN_ROLEMAPPING_FK ON IGACORE.MN_LINK ("MN_ROLE_MAPPING")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MN_ROLE_MAP_LOCALIZ_CODE_FK ON IGACORE.MN_ROLE_MAPPING
("LOCALIZATION_CODE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MN_TREE_LOCALIZ_CODE_FK ON IGACORE.MN_TREE ("LOCALIZATION_CODE")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_MN_TREE_MN_LINK_FK ON IGACORE.MN_TREE ("MN_LINK")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_OBJECT_CLASS_PARENT_FK ON IGACORE.OBJ_CLASS ("PARENT")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_OCF_PARENT_FK ON IGACORE.OBJ_CLASS_FIELD ("PARENT")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ORG_UNIT_OWNER_FK ON IGACORE.ORGANIZATIONAL_UNIT ("OWNER")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ORG_UNIT_PARENT_FK ON IGACORE.ORGANIZATIONAL_UNIT ("PARENT")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_OU_INCOMP_HIERARCHY_FK ON IGACORE.ORGANIZATIONAL_UNIT_INCOMP
("HIERARCHY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_OU_INCOMP_ORG_UNIT_FK ON IGACORE.ORGANIZATIONAL_UNIT_INCOMP
("ORGANIZATIONAL_UNIT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_ORG_UNIT_PROPERTY_OU_FK ON IGACORE.ORG_UNIT_PROPERTY
("ORGANIZATIONAL_UNIT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_INCOMP_ENT_FK ON IGACORE.PERSON_INCOMP ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PER_PRO_SERV_SA_FK ON IGACORE.PERSON_PROFILE_SERVICE
("SERVICE_ATTRIBUTE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_PWDSYNC_PERSON_FK ON IGACORE.PERSON_PWDSYNC ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_PWDSYNC_PWDPOLICY_FK ON IGACORE.PERSON_PWDSYNC ("PWDPOLICY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_PERSONREM_FK ON IGACORE.PERSON_REM_REVIEW ("PERSON_REMEDIATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_PERSON_FK ON IGACORE.PERSON_REM_REVIEW ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_REMEDIATION_FK ON IGACORE.PERSON_REM_REVIEW ("REMEDIATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_REVIEWED_BY_FK ON IGACORE.PERSON_REM_REVIEW ("REVIEWED_BY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_RISK_FK ON IGACORE.PERSON_REM_REVIEW ("ENVIRONMENT", "RISK")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PR_REVIEWER_FIRSTOWNER_FK ON IGACORE.PERSON_REM_REVIEWER
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PR_REVIEWER_SECONDOWNER_FK ON IGACORE.PERSON_REM_REVIEWER
("CERT_SECOND_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_HIS_PERSON_FK ON IGACORE.PERSON_REM_REVIEW_H ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_HIS_REMEDIATION_FK ON IGACORE.PERSON_REM_REVIEW_H ("REMEDIATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_UR_REV_HIS_REVIEWED_BY_FK ON IGACORE.PERSON_REM_REVIEW_H
("CERT_FIRST_OWNER")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PRRNOTE_PERSON_FROM_FK ON IGACORE.PERSON_REM_REVIEW_NOTE ("PERSON_FROM")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PRRNOTE_PERSON_TO_FK ON IGACORE.PERSON_REM_REVIEW_NOTE ("PERSON_TO")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_PERSON_RISK_PERSON_FK ON IGACORE.PERSON_RISK ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_RISK_RISK_FK ON IGACORE.PERSON_RISK ("ENVIRONMENT", "RISK")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_SERVICE_PWDCFG_FK ON IGACORE.PERSON_SERVICE ("PWDCFG")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_SERVICE_PWDMGM_FK ON IGACORE.PERSON_SERVICE ("PWDMANAGEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PERSON_SERVICE_SERV_ATTR_FK ON IGACORE.PERSON_SERVICE
("SERVICE_ATTRIBUTE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PSERVICEPENDING_PROFILE_FK ON IGACORE.PERSON_SERVICE_PENDING ("PROFILE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PWDCFG_PWDPOLICY_FK ON IGACORE.PWDCFG ("PWDPOLICY")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PWDCFG_AK_LOCALIZATION_FK ON IGACORE.PWDCFG_ATTR_KEY ("LOCALIZATION")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PWDMANAGEMENT_PWDMAN_TYPE_FK ON IGACORE.PWDMANAGEMENT
("PWDMANAGEMENT_TYPE")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PWDPOLICY_PROPS_PWDPOLICY_FK ON IGACORE.PWDPOLICY_PROPS ("PWDPOLICY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_PWDPOLICY_WORDS_PWDPOLICY_FK ON IGACORE.PWDPOLICY_WORDS ("PWDPOLICY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_RECORD_LOG_EXECUTION_LOG_FK ON IGACORE.RECORD_LOG ("EXECUTION_LOG")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REMEDIATION_PROPERTY_REM_FK ON IGACORE.REMEDIATION_PROPERTY
("REMEDIATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_CFG_PRIORITY_FK ON IGACORE.REQUEST ("CFG_PRIORITY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_ESC_PROCESSACTIVITY_FK ON IGACORE.REQUEST
("ESCALATION_PROCESSACTIVITY")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_PROCESSACTIVITY_FK ON IGACORE.REQUEST ("PROCESSACTIVITY")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_SUBPROCESSACTIVITY_FK ON IGACORE.REQUEST ("SUB_PROCESSACTIVITY")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_NOTIFICATION_PA_N_FK ON IGACORE.REQUEST_NOTIFICATION
("PA_NOTIFICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_REQUEST_NOTIFICATION_PA_W_FK ON IGACORE.REQUEST_NOTIFICATION
("PA_WORKING")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_RESOURCES_REQUEST_FK ON IGACORE.RESOURCES ("REQUEST")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_RISK_OWNER_FK ON IGACORE.RISK ("OWNER")  ALLOW REVERSE SCANS COLLECT
DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_RISK_RISK_TYPE_FK ON IGACORE.RISK ("ENVIRONMENT", "RISK_TYPE")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SERVICE_ROLES_FK ON IGACORE.SERVICE ("ROLES")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_SATTR_PROFILE_FK ON IGACORE.SERVICE_ATTRIBUTE ("PROFILE")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SATTR_PWDCFG_FK ON IGACORE.SERVICE_ATTRIBUTE ("PWDCFG")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SUPERVISOR_ATTESTATION_FK ON IGACORE.SUPERVISOR ("ATTESTATION")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SUPERVISOR_PERSON_FK ON IGACORE.SUPERVISOR ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SURVEYCFGEXT_SURVEY_CONFIG_FK ON IGACORE.SURVEY_CONFIG_EXT ("SURVEY_ID")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SURVEY_PAGES_SURVEY_CONFIG_FK ON IGACORE.SURVEY_PAGES ("SURVEY_ID")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_SURVEY_USER_SURVEY_CONFIG_FK ON IGACORE.SURVEY_USER ("SURVEY_ID")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TARGET_PWDCFG_FK ON IGACORE.TARGET ("PWDCFG")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TARGET_AUTH_CONNECTOR_FK ON IGACORE.TARGET_AUTH ("CONNECTOR")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TARGET_RES_SCHEMA_DEF_TRSID_FK ON IGACORE.TARGET_RESOURCE_SCHEMA_DEF
("TR_SCHEMA_ID")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TAR_UID_LOCK_USER_EV_ERC_FK ON IGACORE.TARGET_UID_LOCK ("LAST_EVENT_ID")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TASK_OWNER_FK ON IGACORE.TASK ("OWNER")  ALLOW REVERSE SCANS COLLECT
DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TASK_PARENT_FK ON IGACORE.TASK ("ENVIRONMENT", "PARENT")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TASK_ENTITLEMENT_ENTITL_FK ON IGACORE.TASK_ENTITLEMENT ("ENTITLEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TASK_GROUP_ORG_UNIT_FK ON IGACORE.TASK_GROUP ("ORGANIZATIONAL_UNIT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TASK_PERSON_PERSON_FK ON IGACORE.TASK_PERSON ("PERSON")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_APPLICATION_FK ON IGACORE.TEMPLATE_ENTITY
("APPLICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_ENTITLEMENT_FK ON IGACORE.TEMPLATE_ENTITY
("ENTITLEMENT")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_OU_FK ON IGACORE.TEMPLATE_ENTITY ("ORGANIZATIONAL_UNIT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_PERSON_FK ON IGACORE.TEMPLATE_ENTITY ("PERSON")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_PWDCFG_FK ON IGACORE.TEMPLATE_ENTITY ("PWDCFG")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_PWDM_FK ON IGACORE.TEMPLATE_ENTITY ("PWDMANAGEMENT")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_RISK_FK ON IGACORE.TEMPLATE_ENTITY ("ENVIRONMENT",
"RISK")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

```
CREATE INDEX IGACORE.IDX_TEMPLATE_ENTITY_TEMPLATE_FK ON IGACORE.TEMPLATE_ENTITY ("TEMPLATE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_PROPERTY_FLOW_FK ON IGACORE.TEMPLATE_PROPERTY ("FLOW")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGACORE.IDX_TEMPLATE_PROPERTY_TEMPLATE_FK ON IGACORE.TEMPLATE_PROPERTY ("TEMPLATE")
ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;
```

The DB connection for the following script must be initiated with the IGASERV user.

CREATE_INDEXES_IGASERV.sql

The script contains the following indexes:

```
CREATE INDEX IGASERV.IDX_CFG_CONFIG_PROP_CFG_APP_FK ON IGASERV.CFG_CONFIG_PROP
("CFG_APPLICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_FAMILY_FK ON IGASERV.TIMER ("REALM", "FAMILY")  ALLOW REVERSE
SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_HISTORY_TIMER_WORK_FK ON IGASERV.TIMER_HISTORY ("REALM",
"TIMER_WORK")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_WORK_PARENT_FK ON IGASERV.TIMER_WORK ("REALM", "PARENT")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_WORK_TIMER_FK ON IGASERV.TIMER_WORK ("REALM", "TIMER")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_WORK_WORK_FK ON IGASERV.TIMER_WORK ("REALM", "WORK")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_WORK_CLASS_NAME_FK ON IGASERV.WORK ("REALM", "CLASS_NAME")  ALLOW
REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_CFG_APPLICATION_PROP_APP_FK ON IGASERV.CFG_APPLICATION_PROP
("CFG_APPLICATION")  ALLOW REVERSE SCANS COLLECT DETAILED STATISTICS ;

CREATE INDEX IGASERV.IDX_TIMER_SHEDULER_FK ON IGASERV.TIMER ("SCHEDULER")  ALLOW REVERSE SCANS
COLLECT DETAILED STATISTICS ;
```

27. Separation of Duty Scans

Up to versions 5.2.5, to enhance the performance of the Separation of Duty (SoD) scan, the following index (example below is for DB2) were suggested. In laboratory tests, this index improved the SoD scan by 50%, mostly reduced CPU consumption on the DB.

```
CREATE INDEX igacore.personRisk_Remediation_IDX ON IGACORE.PERSON_RISK(PERSON
ASC, REMEDIATION ASC, ENVIRONMENT ASC, LAST_MOD_TIME ASC, RISK ASC, ID ASC)
ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
```

Starting by 5.2.6 is no longer needed to create the above index because the IGI has the index coded in the installation process

28. Launching Multiple Campaigns

The IGI administrator could launch more multiple campaigns. The maximum parallelism is depending by System Scheduler threads. Working with default setting value 2, the maximum number of campaigns to be launched in parallel is 2, if user increase the value to n (in accordance to the maximum recommendation proposed by IGI), then the maximum number of campaigns to be launched in parallel will be n

29. Index on Oracle environment

In the laboratory environment in IGI Oracle Environment, significant performance improvement was observed after applying the following index for Reconciliation – Permission Change scenario.

```
CREATE INDEX IGA_CORE.EMP_REV_PWDM_PWDCFG_FK_IDX ON
IGA_CORE.EMPLOYMENT_REVIEW_PWDMAN(PWDCFG) COMPUTE STATISTICS;
```

30. Indexes on Foreign Key constraints on Oracle Environment

In the laboratory environment in IGI Oracle Environment, significant performance improvement was observed after applying the following indexes in the Foreign Key Constraints for Reconciliation – Delete Account scenario.

```
CREATE INDEX "IGA_CORE"."EMP_REV_PWDM_PWDCFG_FK_IDX" ON
"IGA_CORE"."EMPLOYMENT_REVIEW_PWDMAN" ("PWDCFG") REVERSE PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS NOLOGGING TABLESPACE "IGA_CORE_INDX";


CREATE INDEX "IGA_CORE"."EMP_REV_PWDM_PWDMANAGEMENT_FK_IDX" ON
"IGA_CORE"."EMPLOYMENT_REVIEW_PWDMAN" ("PWDMANAGEMENT") REVERSE PCTFREE 10 INITRANS 2
MAXTRANS 255 COMPUTE STATISTICS NOLOGGING TABLESPACE "IGA_CORE_INDX";
```

31. Tuning IGI in preparation for ISIM -> ISIQ -> ISIG scenario

These indexes are useful for tuning IGI in preparation for the ISIM → ISIQ → ISIG scenario

```
CREATE INDEX PERSON_PROFILE_LF1_IDX       ON IGACORE.PERSON_PROFILE
(LOWER(ACCOUNT_CODE) ASC,LOWER(PERMISSION_CODE) ASC,LOWER(PERMISSION_TYPE)
ASC,LOWER(TARGET) ASC)COMPRESS NO INCLUDE NULL KEYS ALLOW REVERSE SCANS;

CREATE INDEX target_attr3 ON IGACORE.EVENT_TARGET(lower(target),
lower(attr3)) COLLECT DETAILED STATISTICS;

CREATE INDEX APPIDX1 ON "IGACORE "."APPLICATION"
("VALUE" ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

CREATE INDEX PIDX1 ON "IGACORE "."PERSON"
("DN" ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

CREATE INDEX PWDCFIDX ON "IGACORE "."PWDCFG"
("VALUE" ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;

CREATE INDEX attr11 ON IGACORE.USER_ERC(attr11) collect detailed statistics;

CREATE INDEX IGACORE.PERSON_PROFILE_TARGET_ACCOUNTCODE_PERMISSIONCODE ON
IGACORE.PERSON_PROFILE (lcase(TARGET), lcase(ACCOUNT_CODE),
lcase(PERMISSION_CODE)) COLLECT DETAILED STATISTICS;

COMMIT WORK;
```

32. General Tips

Listed here are a few items which do not require much explanation, and thus do not warrant an entire section. These are general guidelines that would apply in most environments.

1. When possible, run Reports in the off-hours. With global workforces, global deployments, and round-the-clock VA usage, it might be difficult to find a time when the VA is at rest. However, there are very likely times when the VA will be less busy. Use this time to run reports, NightShift activities, campaigns, etc.
2. Configure smaller certification campaigns, hierarchies, or role mining jobs. Schedule these smaller jobs to run in periods of lower utilization.
3. Use the procedure listed in Collecting Java Core Dumps to periodically clear the system logs.
4. It is very important for the VA and the database tier to have synchronized clocks. A difference between the clocks can cause delays in operations and affect the accuracy of log comparisons during problem determination.

5. The Identity Broker REST APIs are not enabled by default. To use REST APIs for broker management, group membership, permissions, use the CLI and navigate to igi → utilities → ib_settings → ib_api → enable. Type ‘YES’ to confirm.

Appendix A: Event Record Archival

There is no method provided by IGI for maintenance of the event tables. In normal operation, these tables could grow indefinitely and impact event execution performance with increased time to retrieve the next work event.

A field-tested approach to reducing the size of the EVENT tables is to remove the processed events or archive them based on a specific time window.

The tables involved in this process are:

- EVENT_TARGET
- EVENT_IN
- EVENT_OUT
- EVENT_INTERNAL

To archive the events, the following procedure can be used:

1. Create a copy of the selected event table, duplicating the table structure.
2. Copy the event records from the source table to the duplicate table, using filtering based upon the specific time frame.
3. Remove the original event records from the source table.

Below is an example of this procedure using the DB2 schema and the EVENT_TARGET table.

1. Create the table with no index, etc.

```
CREATE TABLE IGACORE.EVENT_TARGET_2018  (
  ID                            NUMERIC(16)         NOT NULL,
  CODE                          VARCHAR(128)        NOT NULL,
  OPERATION                     NUMERIC(3)          NOT NULL,
  STATE                         NUMERIC(1)          DEFAULT 0 NOT NULL,
  TRACE                         VARCHAR(2000),
```

```
TARGET                      VARCHAR(256)        NOT NULL,
FUNCTIONALITY               VARCHAR(256),
FUNCTIONALITY_TYPE          VARCHAR(256),
ATTR1                       VARCHAR(256),
ATTR2                       VARCHAR(256),
ATTR3                       VARCHAR(256),
ATTR4                       VARCHAR(256),
ATTR5                       VARCHAR(256),
DATE_EVENT                  DATE                DEFAULT CURRENT_DATE NOT NULL,
DATE_PROCESS                DATE                DEFAULT CURRENT_DATE NOT NULL,
OWNERSHIP                   VARCHAR(128)        DEFAULT user,
LAST_MOD_USER               VARCHAR(128),
LAST_MOD_TIME               DATE                DEFAULT CURRENT_DATE,
APPLICATION                 VARCHAR(256),
PRIORITY                    NUMERIC(1)          DEFAULT 0,
PROCESS_ID                  VARCHAR(256),
EMAIL                       VARCHAR(512),
DISPLAY_NAME                VARCHAR(512),
NAME                        VARCHAR(64),
SURNAME                     VARCHAR(64),
DN                          VARCHAR(256),
IDENTITY_UID                VARCHAR(128),
DETAIL                      VARCHAR(512),
ATTR6                       VARCHAR(256),
ATTR7                       VARCHAR(256),
ATTR8                       VARCHAR(256),
ATTR9                       VARCHAR(256),
ATTR10                      VARCHAR(256),
ATTR11                      VARCHAR(256),
ATTR12                      VARCHAR(256),
ATTR13                      VARCHAR(256),
ATTR14                      VARCHAR(256),
ATTR15                      VARCHAR(256),
ATTR16                      VARCHAR(256),
ATTR17                      VARCHAR(256),
ATTR18                      VARCHAR(256),
```

```
  ATTR19                          VARCHAR(256),
  ATTR20                          VARCHAR(256),
  DISPATCHER_ID                   NUMERIC(16),
  SYNC_STATUS                     NUMERIC(16)         DEFAULT 0
)
 IN "IGA_CORE_DATA" INDEX IN "IGA_CORE_INDX" LONG IN "IGA_CORE_BLOB"
 ORGANIZE BY ROW;
```

2. Copy the original event records into the new table.

```
INSERT  INTO    IGACORE.EVENT_TARGET_2018
SELECT
                *
        FROM
                IGACORE.EVENT_TARGET
        WHERE
                DATE_EVENT > '2018-01-01-00.00.00.000000'
                AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

3. Remove the original event records.

```
DELETE FROM IGACORE.EVENT_TARGET
WHERE   DATE_EVENT > '2018-01-01-00.00.00.000000' AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

Below is an example of this procedure using the DB2 schema and the EVENT_IN table.

1. Create the table with no index.

```
CREATE TABLE IGACORE.EVENT_IN_2018  (
  ID                              NUMERIC(16)         NOT NULL,
  EXT_TABLE                       NUMERIC(16)         NOT NULL,
  OPERATION                       NUMERIC(3)          NOT NULL,
```

```
  TRACE                         VARCHAR(2000),
  STATE                         NUMERIC(1)          DEFAULT 0 NOT NULL,
  DATE_PROCESS                  DATE                DEFAULT CURRENT_DATE,
  LAST_MOD_USER                 VARCHAR(128),
  LAST_MOD_TIME                 DATE,
  EXT_ATTR1                     VARCHAR(128),
  EXT_ATTR2                     VARCHAR(128),
  EXT_ATTR3                     VARCHAR(128),
  EXT_ATTR4                     VARCHAR(128),
  EXT_ATTR5                     VARCHAR(128),
  EXT_ATTR6                     VARCHAR(128),
  EXT_ATTR7                     VARCHAR(128),
  EXT_ATTR8                     VARCHAR(128),
  EXT_ATTR9                     VARCHAR(128),
  EXT_ATTR10                    VARCHAR(128),
  OWNERSHIP                     VARCHAR(128)        DEFAULT user,
  DATE_EVENT                    DATE                DEFAULT CURRENT_DATE,
  ERC                           VARCHAR(64),
  PROCESS_ID                    VARCHAR(256),
  DETAIL                        VARCHAR(512),
  DISPATCHER_ID                 NUMERIC(16)
)
 IN "IGA_CORE_DATA" INDEX IN "IGA_CORE_INDX" LONG IN "IGA_CORE_BLOB"
 ORGANIZE BY ROW;
```

2. Copy the original event records into the new table.

```
INSERT  INTO IGACORE.EVENT_IN_2018
SELECT
                *
        FROM
                IGACORE.EVENT_IN
        WHERE
                DATE_EVENT > '2018-01-01-00.00.00.000000'
                AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

3. Remove the original event records.

```
DELETE FROM IGACORE.EVENT_IN
WHERE   DATE_EVENT > '2018-01-01-00.00.00.000000' AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

Below is an example of this procedure using the DB2 schema and the EVENT_OUT table.

1. Create the table with no index.

```
CREATE TABLE IGACORE.EVENT_OUT_2018  (
  ID                            NUMERIC(16)         NOT NULL,
  PERSON                        NUMERIC(16)         NOT NULL,
  USER_ERC                      NUMERIC(16),
  TARGET                        VARCHAR(256)        NOT NULL,
  COD_OPERATION                 VARCHAR(128)        NOT NULL,
  OPERATION                     NUMERIC(3)          NOT NULL,
  STATE                         NUMERIC(1)          DEFAULT 0 NOT NULL,
  DATE_EVENT                    DATE                DEFAULT CURRENT_DATE,
  DATE_PROCESS                  DATE                DEFAULT CURRENT_DATE,
  TRACE                         VARCHAR(1024),
  ATTR1                         VARCHAR(256),
  ATTR2                         VARCHAR(256),
  ATTR3                         VARCHAR(256),
  ATTR4                         VARCHAR(256),
  ATTR5                         VARCHAR(256),
  LAST_MOD_TIME                 DATE,
  LAST_MOD_USER                 VARCHAR(128),
  APPLICATION                   VARCHAR(256),
  PRIORITY                      NUMERIC(1)          DEFAULT 0,
  CONNECTOR                     VARCHAR(256),
  CONNECTOR_STATE               NUMERIC(1),
  PROCESS_ID                    VARCHAR(256),
  ERC_STATUS                    NUMERIC(1)          DEFAULT 0 NOT NULL,
```

```
  USER_ID                       VARCHAR(256),
  DETAIL                        VARCHAR(512),
  DISPATCHER_ID                 NUMERIC(16),
  CHANGELOG                     NUMERIC(16),
  SYNC_STATUS                   NUMERIC(16)         DEFAULT 0
)
 IN "IGA_CORE_DATA" INDEX IN "IGA_CORE_INDX" LONG IN "IGA_CORE_BLOB"
 ORGANIZE BY ROW;
```

2. Copy the original event records into the new table.

```
INSERT  INTO IGACORE.EVENT_OUT_2018
SELECT
                *
        FROM
                IGACORE.EVENT_OUT
        WHERE
                DATE_EVENT > '2018-01-01-00.00.00.000000'
                AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

3. Remove the original event records.

```
DELETE FROM IGACORE.EVENT_OUT
WHERE   DATE_EVENT > '2018-01-01-00.00.00.000000' AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

Below is an example of this procedure using the DB2 schema and the EVENT_INTERNAL table.

1. Create the table with no index.

```
CREATE TABLE IGACORE.EVENT_INTERNAL_2018  (
ID                              NUMERIC(16)         NOT NULL,
STATE                           NUMERIC(16)         DEFAULT 0,
```

```
OPERATION                    NUMERIC(16),
ENTITY1                      NUMERIC(16),
ENTITY2                      NUMERIC(16),
ENTITY1_TYPE                 NUMERIC(16),
ENTITY2_TYPE                 NUMERIC(16),
COD_OPERATION                VARCHAR(512),
TRACE                        VARCHAR(2000),
DATE_EVENT                   DATE              DEFAULT CURRENT_DATE,
DATE_PROCESS                 DATE              DEFAULT CURRENT_DATE,
LAST_MOD_USER                VARCHAR(128),
LAST_MOD_TIME                DATE              DEFAULT CURRENT_DATE,
CREATION_USER                VARCHAR(128),
CREATION_DATE                DATE              DEFAULT CURRENT_DATE,
ENTITY1_NAME                 VARCHAR(512),
ENTITY1_DESC                 VARCHAR(512),
ENTITY1_CODE                 VARCHAR(512),
ENTITY1_OWNER                NUMERIC(16),
ENTITY1_ATTR1                VARCHAR(512),
ENTITY1_ATTR2                VARCHAR(512),
ENTITY1_ATTR3                VARCHAR(512),
ENTITY1_ATTR4                VARCHAR(512),
ENTITY1_ATTR5                VARCHAR(512),
ENTITY2_NAME                 VARCHAR(512),
ENTITY2_DESC                 VARCHAR(512),
ENTITY2_CODE                 VARCHAR(512),
ENTITY2_OWNER                NUMERIC(16),
ENTITY2_ATTR1                VARCHAR(512),
ENTITY2_ATTR2                VARCHAR(512),
ENTITY2_ATTR3                VARCHAR(512),
ENTITY2_ATTR4                VARCHAR(512),
ENTITY2_ATTR5                VARCHAR(512),
DISPATCHER_ID                NUMERIC(16)
)
 IN "IGA_CORE_DATA" INDEX IN "IGA_CORE_INDX" LONG IN "IGA_CORE_BLOB"
 ORGANIZE BY ROW;
```

2. Copy the original event records into the new table.

```
INSERT  INTO IGACORE.EVENT_INTERNAL_2018
SELECT
                        *
        FROM
                IGACORE.EVENT_INTERNAL
        WHERE
                DATE_EVENT > '2018-01-01-00.00.00.000000'
                AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

3. Remove the original event records.

```
DELETE FROM IGACORE.EVENT_INTERNAL
WHERE   DATE_EVENT > '2018-01-01-00.00.00.000000' AND DATE_EVENT < '2018-12-31-00.00.00.000000'
```

Maintenance of the EVENT_ACCOUNT table is less complicated. This is a transitional table where data remains before being automatically copied/moved into the IGASERV.AUDIT_LOG table. All entries in the EVENT_ACCOUNT table having STATE=1 can be safely removed because those records have already been copied into the IGASERV.AUDIT_LOG table.

Different methods can be used to perform maintenance on the EVENT_ACCOUNT table based on how many records are present with STATE=1.

If **ALL** records have STATE=1, then

```
TRUNCATE IGACORE.EVENT_ACCOUNT;
```

If there are records with STATE=1 and records with STATE=0, then

```
DELETE FROM IGACORE.EVENT_OUT WHERE STATE=1;
```

If there is a very large number of records with STATE=1, the previous query must be performed partitioning the events into subsets.

Lastly, the JOB table has the potential to grow indefinitely if the processed operations are not removed from the system with a certain frequency. That frequency depends on the individual operating environment. With respect to the JOB table, there is no reason to maintain the processed jobs in a history table. A simple delete can clear the table.

```
DELETE FROM IGACORE.JOB WHERE STATE in (2,3,5,6,7);
```

As this table is likely to contain many records, the operation is best performed in multiple steps (removing 1000 elements at a time).

```
DELETE FROM IGACORE.JOB WHERE STATE in (2,3,5,6,7)AND
ID<=(SELECT min(ID) FROM IGACORE.JOB)+1000;
```

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features contained in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM might have patents or pending patent applications that cover subject matter described in this document. The furnishing of this document does not grant you any license to these patents. Send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web

sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it to enable: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information might be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments might vary significantly. Some measurements might have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements might have been estimated through extrapolation. Actual results might vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources.

IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products.

Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding the future direction or intent of IBM are subject to change or withdrawal without notice and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing,

using, marketing, or distributing application programs that conform to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp.

2004, 2013. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations might not appear.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both: http://www.ibm.com/legal/copytrade.shtml

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names might be trademarks or service marks of others.