SUCCESS WORKBOOK

Fine tune your ServiceNow platform with regular performance administration

Developed by the ServiceNow Best Practice Center of Excellence

March 2022

What is ServiceNow maintenance and performance tuning?

While we monitor your instance hardware in our data centers, every organization makes configurations to their environments that can affect them individually. If left completely unattended, small issues can build up over time and can cause performance problems, difficulty upgrading to new releases, or issues when deploying additional applications. This workbook will provide guidance on how to keep your instances fine tuned. Steps included are:

- Check for configuration that is not best practice
- Review your log data for errors and warnings
- Maintain your tables for peak performance

Why should I invest in topic?

With effective maintenance, you can improve both the upgradeability and scalability of your ServiceNow instance as well as ensure that you don't have performance issues related to your configurations.

What will this workbook help me do?

ServiceNow instances can be like cars—two people can own the same model, year, and make of a vehicle, but depending on how they use, maintain, and customize it, they could have two entirely different experiences with the car.

Perform the tasks in all five of the following steps as they are explained in the workbook in order to keep your instance running optimally.

Step 1 – Daily instance maintenance

Step 2 – Weekly instance performance

Step 3 – Monthly instance performance

Step 4 – Quarterly instance performance

Step 5 – Keep your instance continually improving

How to use this Success Workbook

This workbook will guide you through the steps to get started with ServiceNow governance.

Start by reading through the Workbook Checklist and review the steps and tasks to get started with ServiceNow governance. From the checklist page, you can either proceed through the workbook page by page or navigate only to the sections that you need using the hyperlinks.

Each section includes "check your progress" questions that you can answer to test whether you need to complete the tasks listed on that page or if you've already done so and should move forward. Hyperlinks are included throughout the workbook so you can navigate back and forth between the checklist page and different sections.

Workbook Checklist: Fine tune your ServiceNow platform with regular performance administration

Step 1: Daily instance maintenance

- ☐ Review the System Diagnostics homepage
- ☐ Review the previous day's slow transactions
- ☐ Consider you use cases

Step 2: Weekly instance maintenance

- ☐ Review your scheduled jobs
- ☐ Check for repeated errors in the error log
- ☐ Look for excessive logging
- ☐ Find log files over 1 GB
- ☐ Find slow-running running jobs
- ☐ Find long-running jobs
- ☐ Trend your top 20 transactions

Step 3: Monthly instance maintenance

- ☐ Monitor your table growth rates
- ☐ Clean your tables
- ☐ Clean your tables

Step 4: Quarterly Instance performance

- ☐ Check for coding that could impact your ability to upgrade

Step 5: Keep your instance continually improving

- ☐ Find and fix issues with your end to end response time

Step 1: Daily Instance Maintenance

What's slowing you down might be simple to find. Check the three following areas to see if your transactions or data pulls are issues.

Check your progress:

- ❑ Do you have a daily routine for monitoring your instances' health?
- ❑ Do you check for errors in your systems diagnostics?
- ❑ Do you receive total response times for transactions?
- ❑ Do you monitor users who are pulling large data sets and review their business need?

If you answered "yes" to the questions above, proceed to the next step. If not, complete these action steps to complete this step:

1. **Review the System Diagnostics homepage**

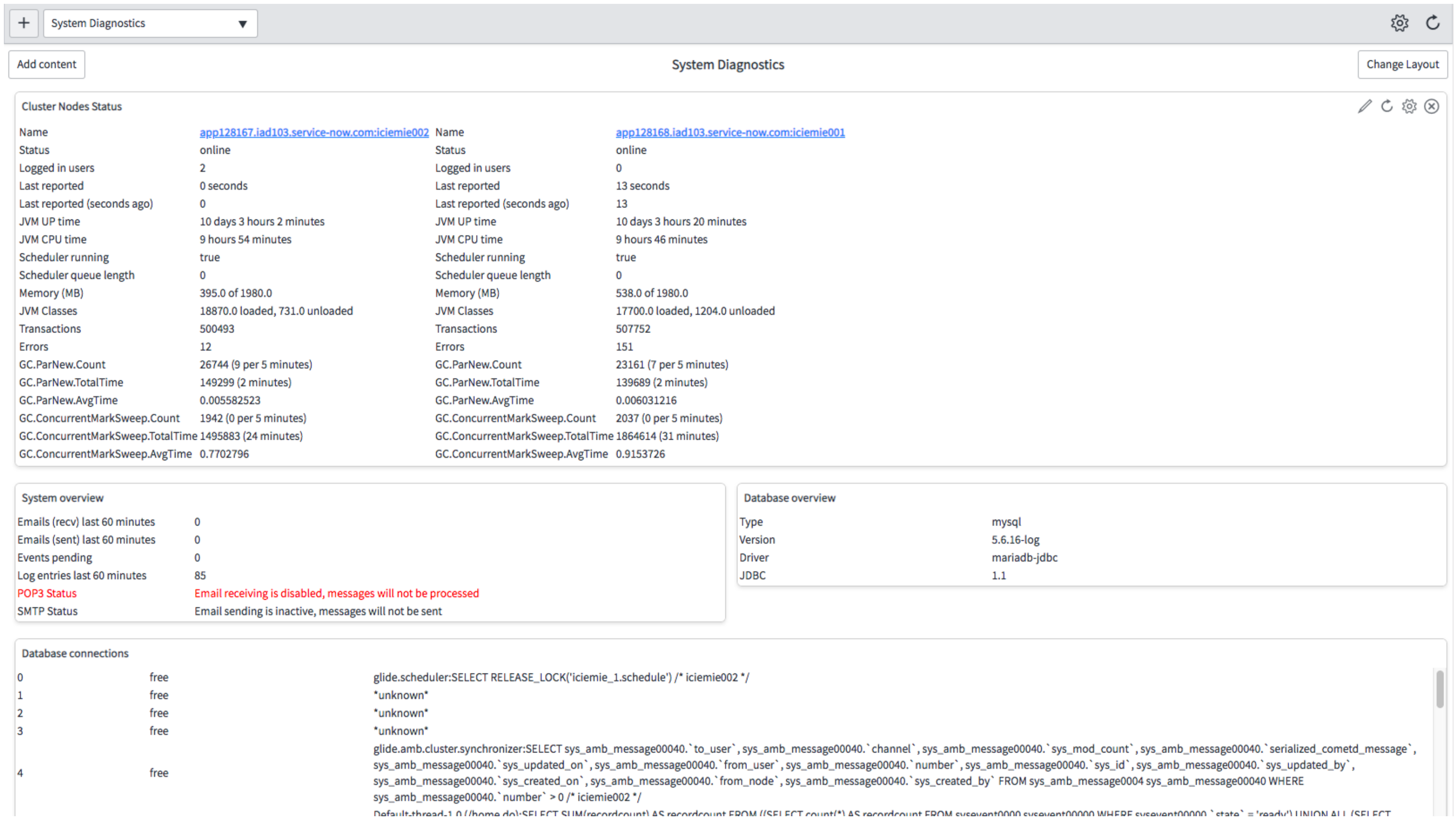
   The System Diagnostics page tracks some high-level statistics for each of the nodes (JVMs) in your instance. (See Figure 1.)

   When you review this information, don't worry if the total number of JVM Classes differs between nodes. This metric is showing the number of classes that have been loaded and subsequently unloaded on each JVM. Depending on the activities users performed on each node, you might notice a legitimate disparity in what has been called since that JVM was last started.

   a) Go to the System Diagnostics homepage.

   b) Review the values on this page. You'll see values either in real time at the point the page is rendered or as cumulative counts (such as the transactions and error values) since the node was last started (see the JVM UP time).

   c) Track this information in a spreadsheet or a table in your instance. Include the uptime, number of errors since the last restart, the number of transactions performed, and the number of logged-in users for each node. While the Now Platform® does have built-in performance graphs that show this information, they're rendered on a per-node basis. If you spot an uncharacteristic jump in these numbers, it can be a good indicator there is an underlying performance issue you need to identify and address.



System Diagnostics

Add content | Change Layout

Cluster Nodes Status

Name	app128167.iad103.service-now.com:iciemie002	Name	app128168.iad103.service-now.com:iciemie001
Status	online	Status	online
Logged in users	2	Logged in users	0
Last reported	0 seconds	Last reported	13 seconds
Last reported (seconds ago)	0	Last reported (seconds ago)	13
JVM UP time	10 days 3 hours 2 minutes	JVM UP time	10 days 3 hours 20 minutes
JVM CPU time	9 hours 54 minutes	JVM CPU time	9 hours 46 minutes
Scheduler running	true	Scheduler running	true
Scheduler queue length	0	Scheduler queue length	0
Memory (MB)	395.0 of 1980.0	Memory (MB)	538.0 of 1980.0
JVM Classes	18870.0 loaded, 731.0 unloaded	JVM Classes	17700.0 loaded, 1204.0 unloaded
Transactions	500493	Transactions	507752
Errors	12	Errors	151
GC.ParNew.Count	26744 (9 per 5 minutes)	GC.ParNew.Count	23161 (7 per 5 minutes)
GC.ParNew.TotalTime	149299 (2 minutes)	GC.ParNew.TotalTime	139689 (2 minutes)
GC.ParNew.AvgTime	0.005582523	GC.ParNew.AvgTime	0.006031216
GC.ConcurrentMarkSweep.Count	1942 (0 per 5 minutes)	GC.ConcurrentMarkSweep.Count	2037 (0 per 5 minutes)
GC.ConcurrentMarkSweep.TotalTime	1495883 (24 minutes)	GC.ConcurrentMarkSweep.TotalTime	1864614 (31 minutes)
GC.ConcurrentMarkSweep.AvgTime	0.7702796	GC.ConcurrentMarkSweep.AvgTime	0.9153726

System overview

Emails (recv) last 60 minutes	0
Emails (sent) last 60 minutes	0
Events pending	0
Log entries last 60 minutes	85
POP3 Status	Email receiving is disabled, messages will not be processed
SMTP Status	Email sending is inactive, messages will not be sent

Database overview

Type	mysql
Version	5.6.16-log
Driver	mariadb-jdbc
JDBC	1.1

Database connections

0	free	glide.scheduler:SELECT RELEASE_LOCK('iciemie_1.schedule') /* iciemie002 */
1	free	*unknown*
2	free	*unknown*
3	free	*unknown*
4	free	glide.amb.cluster.synchronizer:SELECT sys_amb_message00040.\`to_user\`, sys_amb_message00040.\`channel\`, sys_amb_message00040.\`sys_mod_count\`, sys_amb_message00040.\`serialized_cometd_message\`, sys_amb_message00040.\`sys_updated_on\`, sys_amb_message00040.\`from_user\`, sys_amb_message00040.\`number\`, sys_amb_message00040.\`sys_id\`, sys_amb_message00040.\`sys_updated_by\`, sys_amb_message00040.\`sys_created_on\`, sys_amb_message00040.\`from_node\`, sys_amb_message00040.\`sys_created_by\` FROM sys_amb_message0004 sys_amb_message00040 WHERE sys_amb_message00040.\`number\` > 0 /* iciemie002 */
		Default-thread-1 0 (/home.do):SELECT SUM(recordcount) AS recordcount FROM ((SELECT count(*) AS recordcount FROM sysevent0000 sysevent00000 WHERE sysevent00000.\`state\` = 'ready') UNION ALL (SELECT

Figure 1: System Diagnostics homepage

2. Review the previous day's slow transactions

The ServiceNow system logs module provides a variety of logs that you can use to troubleshoot and debug transactions and events that take place within the instance. The transaction logs in the system logs table records all browser activity for an instance. See image below:

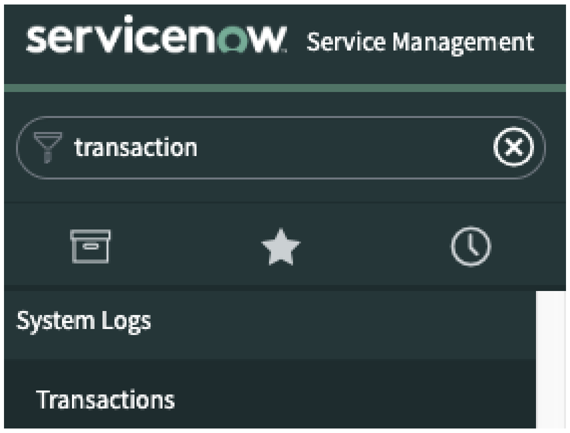


Figure 2: Access transaction logs through system logs table

By reviewing all users' transaction information, you can see which transactions are taking more than a reasonable amount of time. Before you start, ensure you have the Client Transaction Timings plugin enabled to capture all the data. Also note, depending on the size of your instance, this table can be huge and may time out before the results load, so ensure you load it with 'sysparm_filter_only=true' then specify a filter to start looking at results.

In the list of transactions, you can view the total response times along with:

- A breakdown of the composite parts – This includes the time spent rendering in the browser, time spent on the server processing the transaction, and calculated time spent in the network.
- The details of which node processed the request
- The IP address of the host making the request
- The user making the request
- When the transaction occurred
- The session ID – Since this is also captured, it's possible to review the application logs to dissect every action a user has performed in their session.

Look for:

- If there is a particular time of day when transactions execute slowly
- If these transactions are all being processed by the same node – This suggests one or more transactions or background jobs are consuming large quantities of memory.
- If the transaction response times are poor across all nodes – This typically signifies the database was working harder than usual, impacting all transactions.

You might notice that the top 10 slowest transactions were all issued by a single user and are incident lists. If that's the case, you can review the user's settings or impersonate that user and try to recreate the issue.

You may also want to filter transactions by URL to analyze the slow transactions. Additionally, reporting on aggregate response times can be a powerful way to track how overall instance performance is changing with time.

For more detailed instructions on how to work with the transaction logs, see our product documentation site.

3. Consider your use cases

How much data do your users truly need to review in a single screen? If you identify that your list transactions are slow, find out how much data your users are requesting. When a user selects "Show 100 rows per page" on a list, this sets a user preference. From that point forward, every time that user runs a list view, it will include 100 rows. This includes related and embedded lists on forms as well as the list views where the user set the preference.

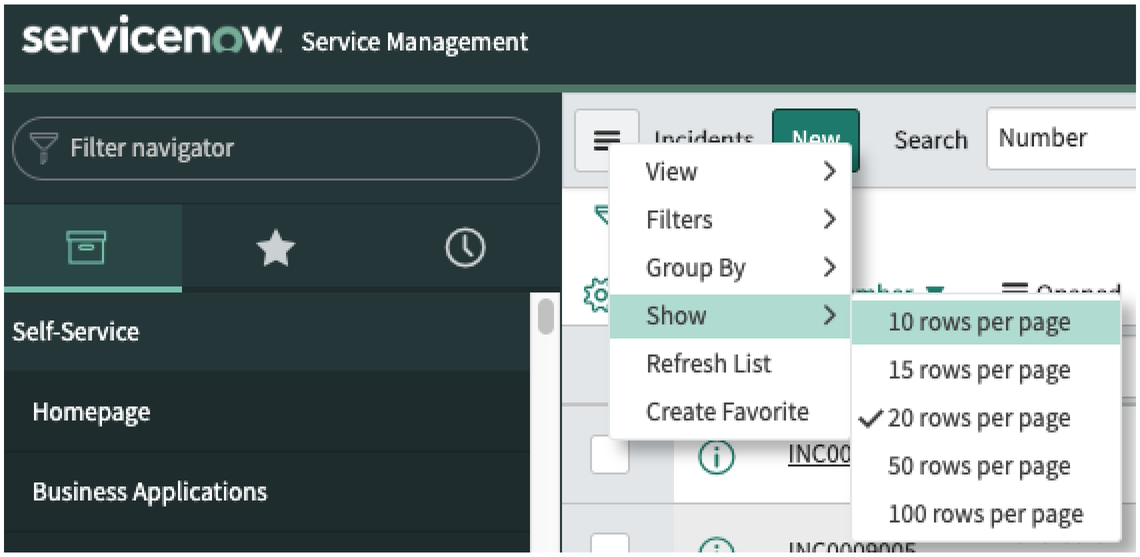
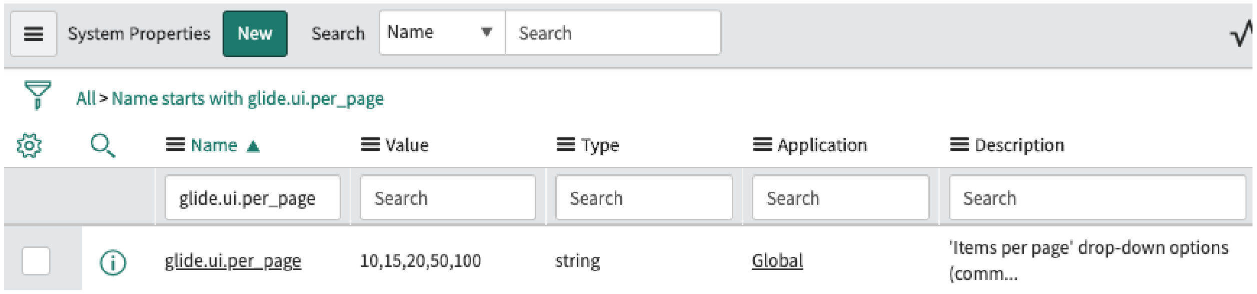


Figure 3: User chooses to show 100 rows per page

If you have a sufficient number of users who are requesting high numbers of rows, you could experience a platform wide performance degradation because those row counts are making high demands on the JVM memory required to render the lists.

This becomes problematic when a table with many reference fields must render a list. The platform has to build the relationships for all the reference fields for all the rows displayed on screen.

For most service environments, agents can't practically use more than 20–30 rows at a time. If the page load is fast, you can make a good case for "paging" to the next chunk of results rather than scrolling down. Consider to use the 'Glide.ui.per_page' property to define the items per page drop down options that are visible for users.



Name ▲	Value	Type	Application	Description
glide.ui.per_page	Search	Search	Search	Search
glide.ui.per_page	10,15,20,50,100	string	Global	'Items per page' drop-down options (comm...

Figure 4 Glide.ui.per_page property

Consider removing any options for more than 50 rows at a time from the Now Platform. For further details, visit our knowledge base for an article called "Good practices to improve instance performance through Rowcount, Related Lists and Dashboards."

Return to workbook checklist

Step 2: Weekly Instance performance

If routine tasks have become a problem, finding errors, warnings, large log files, and slow jobs will help you get those tasks running smoothly.

Check your progress:

- ❑ Do your scheduled jobs run smoothly?
- ❑ Do you regularly investigate repeated errors and warning in your logs?
- ❑ Do you have a process to monitor users logging excessively and large log files?
- ❑ Do you regularly check for slow running jobs?

If you answered "yes" to each question above, proceed to the next step. If not, complete these action tasks:

1. **Review your scheduled jobs**

   By reviewing your scheduled job activity, you can help ensure that background activities, such as scheduled reports, discovery sensors, and other routine tasks, run smoothly. Check for anything that's running for more than an hour (3,60,000 ms)

   a) Navigate to **System Logs** > **Transactions** (Background)

   b) Apply a filter with the following conditions (see Figure 3):

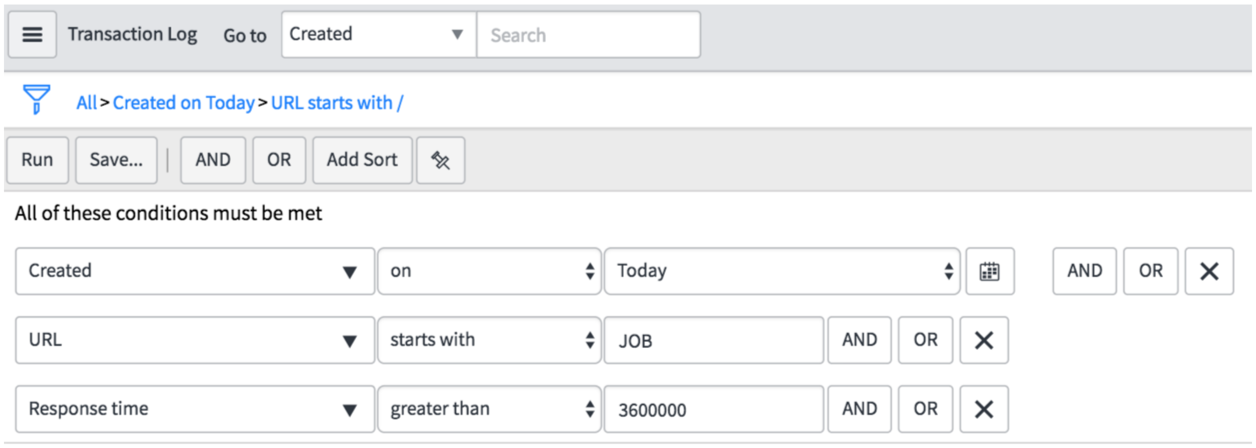
      i. Created > on > This week

      ii. URL > starts with > JOB

      iii. Response time > Greater than > 360000

      NOTE: The response may take several minutes to return. If you don't return any results for an hour, try the same steps again with a more stringent value such as a half hour (18000 ms) Of course, some scheduled jobs are going to take a long time because they have a lot of work to process. Due to how the transaction log tables are stored and rotated in the database, it is not possible to use the "group by" function in the list view. Because of this, you may find it easier to do you trend analysis by exporting the result set to Excel.

   c) If you see a job that has executed multiple times for a long duration, drill down into what the problem is. The most common culprits are glide record queries, which request information from large tables with un-indexed "where" clauses or sorts/groups. These are often found inside of scripted transform maps and sometimes inside of script includes or business rules.



All of these conditions must be met

Created	on	Today	AND	OR
URL	starts with	JOB	AND	OR
Response time	greater than	3600000	AND	OR

Figure 5: Filter showing all job transactions created in the current week that took more than 360,000 ms to complete

2. **Configure scheduled jobs to use "Burst" scheduler workers**

   To insulate against backed up scheduler worker queues, set the Priority field on the sys_trigger entry for the scheduled job to 25. This ensures that the core jobs—event processors, SMTP sender, POP reader, and SMS sender—get triggered in a timely fashion. Should all the scheduler workers be busy with other jobs, an "important" job, which is more than 60 seconds past due, will spawn a "Burst" scheduler worker and execute in parallel to the core eight schedulers on the node.

Practitioner insight: Using "Burst" scheduler worker is good insulation, but don't use it as an excuse to avoid addressing the root causes of the other long-running or high-volume scheduled jobs.

3. **Check for repeated errors in the error log**
   a) Navigate to the System Log.
   b) Select Errors.
   c) Look for actionable errors as well as frequency within the warning messages.
   d) Look for an increased volume in the number of errors by checking the total number in the top right corner of the screen.
   e) If you see a message like **org.mozilla.javascript.gen.sys_script_include_5daa9bf593233100fa71b33e867ffb9b_script_2555.call(sys_script_include_5daa9bf593233100fa71b33e867ffb9b.script**, you can discover more about the error by examining the script_include record with that sys_id.
   f) Look for repeated errors in the warnings log
      I. Navigate to the **System Log**.
      II. Select **Warnings**.
      III. Look for actionable warnings as well as frequency.
      IV. Based on the warnings you see, you may be able to search through a **sys_script** for the text output.

4. **Look for excessive logging**

   Next, look for unusually large log files. This is a relatively crude—but surprisingly accurate—way to spot potential problems that warrant closer attention.

   a) Navigate to **Utilities > Node Log File Download**.
   b) Apply a **Name starts with local** filter. This will show you all the application logs for the node your session is active on.
   c) Note that the most recent five days of log files are unzipped, and the remaining files are zipped. The size value is measured in KBs. If you notice that one day is significantly larger than the others, or that there is a progressive increase in file size, you may need to investigate further.

Practitioner insight: The application logs all transactions and associated parameters, so if the number of users has ramped up or a new piece of functionality has gone live, the log files will naturally increase.

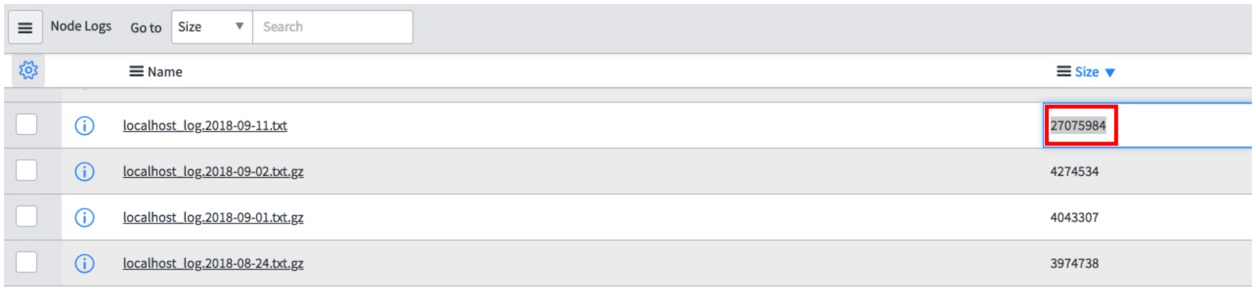
5. **Find log files over 1 GB**

   Log files over 1 GTB may suggest possible frequent errors or logging issues that you need to fix.

   a) First, look for a significant spike in log file size.

   b) Note: This may indicate that the gs.log or gs.pirnt statements ,which were used in sub production testing, have not been removed. Unnecessary logging makes the tables bulky, which slows maintenance activates, like backups, and makes searching the syslog table slow and cumbersome. If that's the case, try to remove the gs.log and/or gs.print statements (unless you need them) and complete steps 1-4 again.

   c) Find the log files that are over 1 GB.



Name	Size
localhost_log.2018-09-11.txt	27075984
localhost_log.2018-09-02.txt.gz	4274534
localhost_log.2018-09-01.txt.gz	4043307
localhost_log.2018-08-24.txt.gz	3974738

Figure 6: A log file over 1 GB

6. **Find slow running jobs**

   a) Apply a Navigate to the **System Scheduler**.

   b) Select **Slow Job Log**.

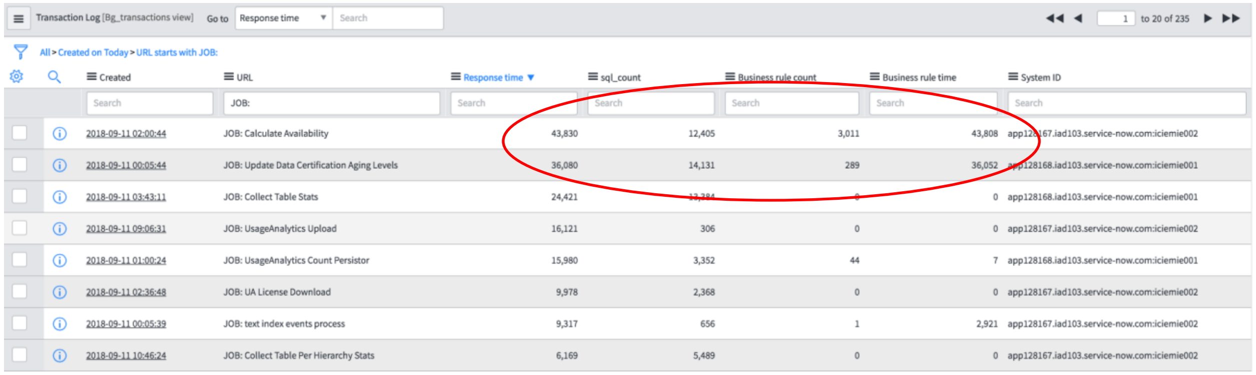
   c) View the job details in the **URL and Response time** columns.

   d) Check the **SQL time column** for the time the job has been in the database.

   e) Check the **Business rule time** column for the amount of time the job has been in logic (execution).

   f) Right-click the **Response time** column heading and select Sort (z to a).

   g) Review the **Response time**, **SQL time**, and **Business rule time** to look for suspiciously long run times.



Transaction Log [Bg_transactions view] Go to Response time Search — 1 to 20 of 235

All > Created on Today > URL starts with JOB:

Created	URL	Response time	sql_count	Business rule count	Business rule time	System ID
2018-09-11 02:00:44	JOB: Calculate Availability	43,830	12,405	3,011	43,808	app128167.iad103.service-now.com:iciemie002
2018-09-11 00:05:44	JOB: Update Data Certification Aging Levels	36,080	14,131	289	36,052	app128168.iad103.service-now.com:iciemie001
2018-09-11 03:43:11	JOB: Collect Table Stats	24,421	13,384	0	0	app128168.iad103.service-now.com:iciemie001
2018-09-11 09:06:31	JOB: UsageAnalytics Upload	16,121	306	0	0	app128167.iad103.service-now.com:iciemie002
2018-09-11 01:00:24	JOB: UsageAnalytics Count Persistor	15,980	3,352	44	7	app128168.iad103.service-now.com:iciemie001
2018-09-11 02:36:48	JOB: UA License Download	9,978	2,368	0	0	app128167.iad103.service-now.com:iciemie002
2018-09-11 00:05:39	JOB: text index events process	9,317	656	1	2,921	app128167.iad103.service-now.com:iciemie002
2018-09-11 10:46:24	JOB: Collect Table Per Hierarchy Stats	6,169	5,489	0	0	app128167.iad103.service-now.com:iciemie002

Figure 7: Example of a Slow Job log

7. **Find long running jogs**

   a) Navigate to **User Administration**.

   b) Select **Active Transactions**.

   c) If there is a background job running, it will show in the User column. Check the Age column to see how long it's been running.

   d) To kill a job that's been running for too long or seems to be completely stuck, right-click the **User name** and select **Kill. Only kill a job if you are <u>ABSOLUTELY</u> sure it is a stuck or illegitimate long running job.**

   e) A confirmation message will appear at the top of the list.

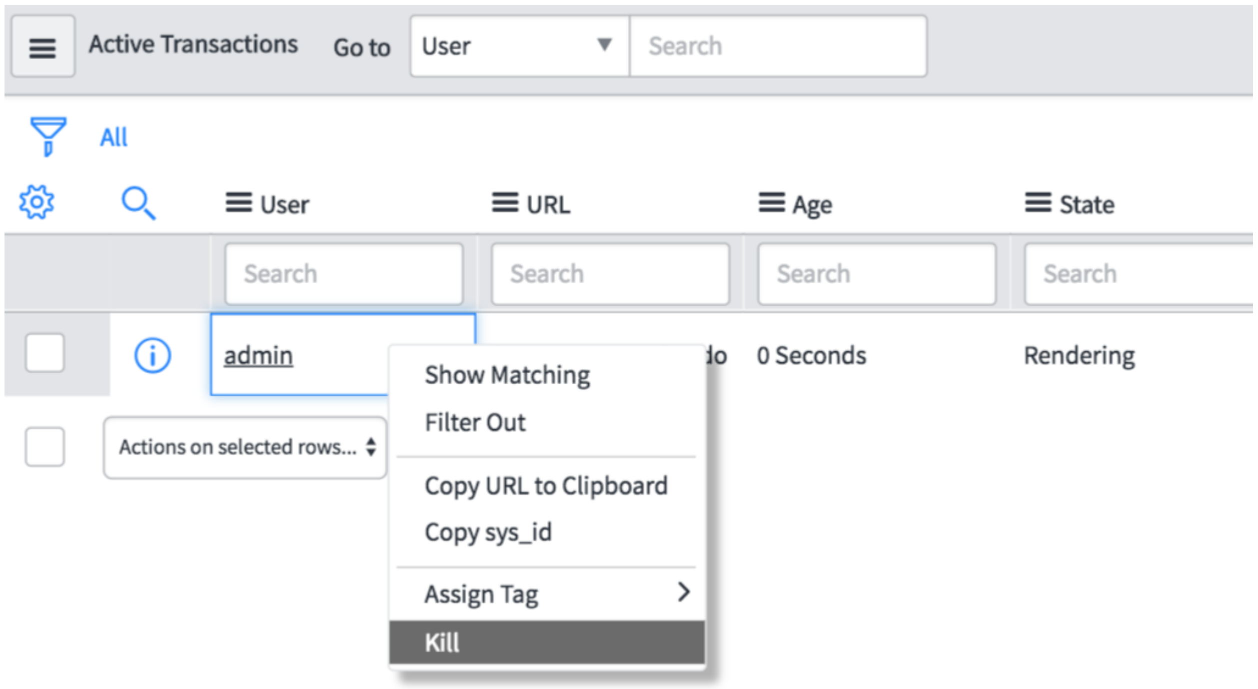


Figure 8: Right-click menu for killing a stuck job

8. Trend your top 20 transactions

Create a spreadsheet to trend your top 20 transactions. These may constitute the 20 most executed transactions in a given week. Or you may choose to track the most business-critical transactions (like incident or catalog transactions). Or it may be helpful to trend a mixture of these. Keep tracking data week after week .

Refer to this knowledge base article on the Now Support Portal for advice on how to investigate the performance of individual transactions.

Return to workbook checklist

Step 3: Monthly Instance performance

Don't let slow queries or transient data bog down your performance. Practice these monthly reviews to avoid those issues.

There are broadly two types of data stored in your ServiceNow instance:

- Persistent data that you want to retain, such as a task or user info
- Transient data that needs to be cleared after a given time frame, such as log information or staging data for imports or integrations

It's normal to see persistent data growth over time. But when you see increased table sizes along with a decreased response time, you may have list definitions or glide record queries that need to be refactored or indexed so you can accommodate the data growth.

If there's an increase in response times for end users and an increase in execution time for maintenance tasks such as cloning, backup, and restore, it probably means your data sets are growing. If that's the case, it's important to monitor your table growth on a monthly basis. If you created a spreadsheet to track your top 20 request response times, you can extend it to track the number of rows rendering in your tables, as well as your slow queries, so you can track their improvement over time.

Check your progress:

- ❑ Do you regularly track table growth?
- ❑ Do you purge or archive data that is no longer needed in the active system?
- ❑ Do you review slow queries that might indicate a table that needs to be indexed?

If you answered "yes" to each question above, proceed to the next step. If not, complete these action tasks:

1. **Monitor your table growth rates**

   When you check your table growth, you'll look for two things:

   a) Dramatic changes in size from month to month

   b) The total number of records in your tables

Follow these steps:

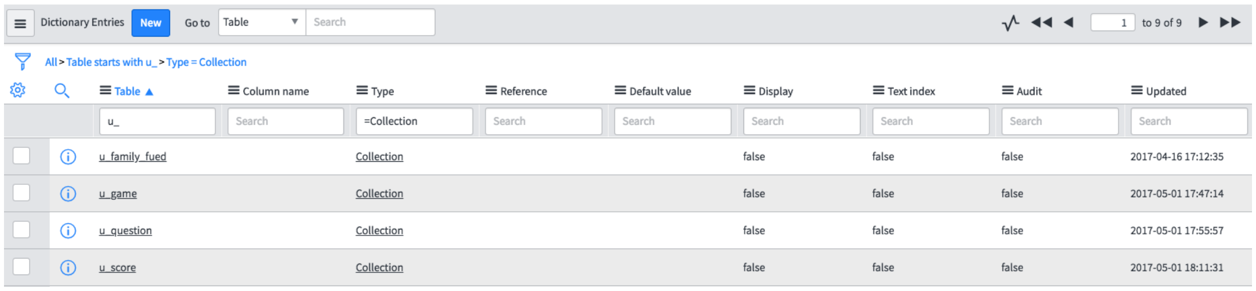
a) Navigate to **System Definition**.

b) Select **Tables**.

c) Your list of tables appears on the screen.

d) Filter the table information typing u_ in the box. This will show you all the user-created tables .

e) To see the total number of records in a table, type <name of table>.list in the navigation field at the top left of the screen.



Dictionary Entries New Go to Table Search 1 to 9 of 9

All > Table starts with u_ > Type = Collection

Table ▲	Column name	Type	Reference	Default value	Display	Text index	Audit	Updated
u_	Search	=Collection	Search	Search	Search	Search	Search	Search
u_family_fued		Collection			false	false	false	2017-04-16 17:12:35
u_game		Collection			false	false	false	2017-05-01 17:47:14
u_question		Collection			false	false	false	2017-05-01 17:55:57
u_score		Collection			false	false	false	2017-05-01 18:11:31

Figure 9: A table list showing a total of four tables

f) Look for a dramatic change between the previous month and this month. If you see an increase, you may need to investigate why the spike occurred.

g) Check the total records.

If the total number of records in your tables is over 50K, you may need to complete one of these tasks or a combination of them:

- **Index the fields used in filters or other queries –** To determine if you need to index fields, review the Slow Queries log. If you find slow queries, contact ServiceNow for support about indexing.
- **Set up table rotation with from ServiceNow technical support –** For more information, read the next section and our product page on table rotation.
- **Extend large tables to store more data or as a part of a table rotation process –** For more information, read our product page explaining table extension for data retention.
- **Clean tables by purging rows –** For more information, read the next section.

Practitioner insight: To see the total size of your database and the 10 biggest tables, use the Database Footprint service catalog item on the Now Support portal.

2. Clean your tables

To list the data you may want to purge (such as incidents older than one year):

a) Type incident.list in the left navigation field, or type https://<instancename>.service-now.com/incident_list.do into your browser's address bar.

b) A count of the records will display.

c) You may want to amend your show x records preference to 10 or 20 to speed up the list rendering time.

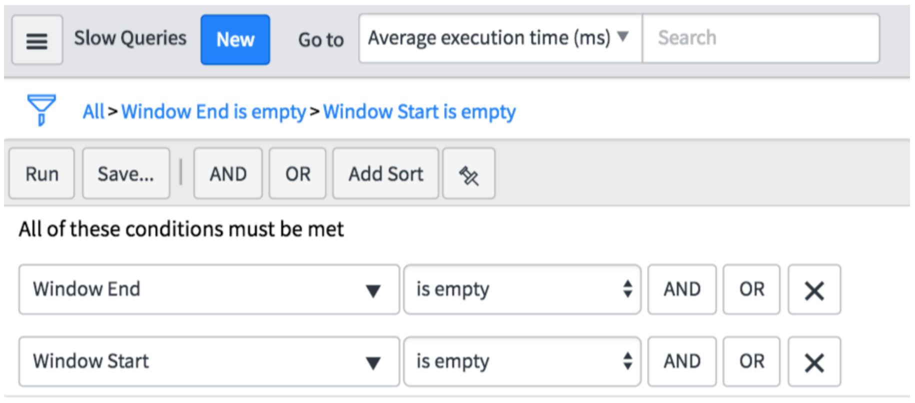
Practitioner insight: If you discover that you need to purge more than one million records, submit a ticket on the Now Support portal to handle the purge. Purging them yourself could cause an outage or other issues.

3. Review the slow queries log

The Slow Queries log aggregates the data for similar queries. The platform records any SQL statement where the total execution time exceeds five seconds.

Not every slow query is a concern—you can expect to have a few. The slow queries on user created tables, those that begin with "u_," are the ones that are cause for concern. Dramatic changes in size from month to month could indicate an issue.

a) Navigate to System Diagnostics **> Stats > Slow Queries** to see your Slow Queries log. The platform records any SQL statement that takes more than 100ms to complete. The Slow Queries log groups these transactions into similar patterns, providing you with an example set of parameters


Slow Queries
New
Go to
Average execution time (ms)
Search
All > Window End is empty > Window Start is empty
Run
Save...
AND
OR
Add Sort

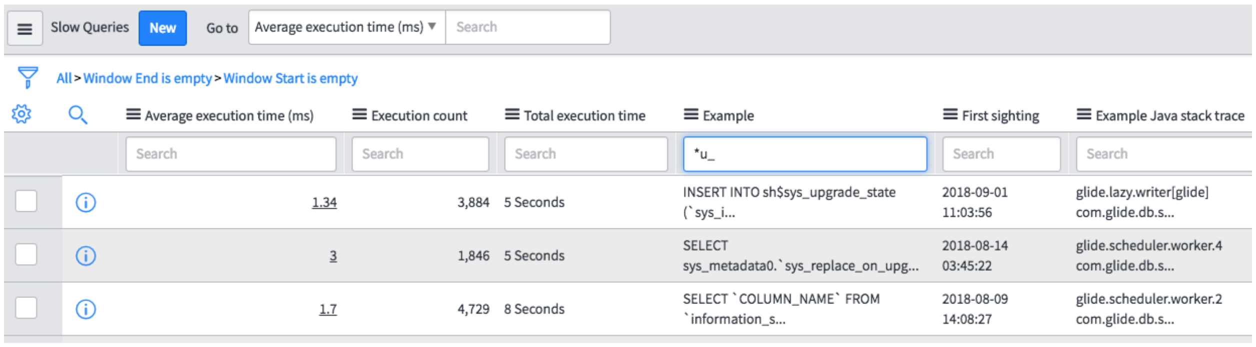
All of these conditions must be met

Window End — is empty

Window Start — is empty

Figure 10: A Slow Queries log

b) Type **u_** in the **Example** text box


Slow Queries
New
Go to
Average execution time (ms)
Search
All > Window End is empty > Window Start is empty

Average execution time (ms)	Execution count	Total execution time	Example	First sighting	Example Java stack trace
Search	Search	Search	*u_	Search	Search
1.34	3,884	5 Seconds	INSERT INTO sh$sys_upgrade_state (`sys_i...	2018-09-01 11:03:56	glide.lazy.writer[glide] com.glide.db.s...
3	1,846	5 Seconds	SELECT sys_metadata0.`sys_replace_on_upg...	2018-08-14 03:45:22	glide.scheduler.worker.4 com.glide.db.s...
1.7	4,729	8 Seconds	SELECT `COLUMN_NAME` FROM `information_s...	2018-08-09 14:08:27	glide.scheduler.worker.2 com.glide.db.s...

Figure 11: A Slow Query log showing the Example text box.

The slow query log records the queries' patterns since the beginning of time (or since the last time **sys_query_pattern** was truncated). You may find the results more meaningful by applying a filter to show only patterns that were first sighted in the last month and that occurred more than 100 times.

If you click through to an individual query pattern record, you'll see an example URL where the query was generated, the first and last sighting, the number of executions, and the average execution time.

The stack trace of the thread executing the query also displays. From here:

a) Cross-reference which element on the screen requested the information.

b) Once you know this, you can review the gauge or list that made the call and verify whether it would benefit from refactoring or supporting with an index. Many times, you can significantly reduce the execution time by simply adding **active=1** to a query. This will only query and return active records, thus reducing the number of records included in the query.

Return to workbook checklist

Step 4: Quarterly Instance performance

Running a quarterly review of your upgrade history can tell you a lot about your instance performance as you can see issues/performance trending over time.

Check your progress:

- ❑ Have you checked your skipped, inserted, updated, and deleted records for unusual behavior such as odd growth patterns, mass deletes, etc.?
- ❑ Do you monitor changes or customizations to ServiceNow out of the box object?

If you answered "yes" to each question above, proceed to the next step. If not, complete these action steps:

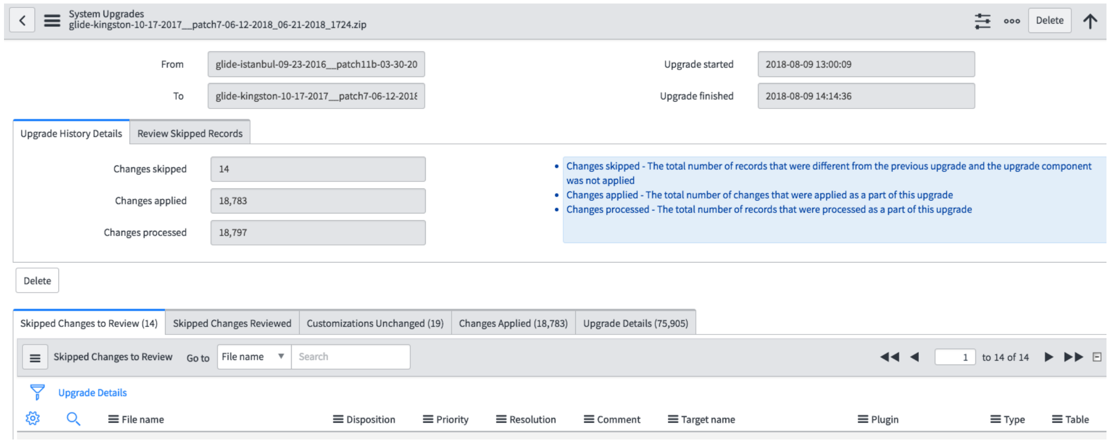
1. **Check your instance for any configurations that could impact your upgrades**

   a) Navigate to **System Diagnostics**.

   b) Select **Upgrade History**.

   c) A table showing your upgrade history appears. Under the **Upgrade started** column, look for the last upgrade and select it.

   d) Within the upgrade record, check the **Skipped Changes to Review** tab to see what was skipped, updated, inserted, or deleted.



System Upgrades
glide-kingston-10-17-2017__patch7-06-12-2018_06-21-2018_1724.zip

From	glide-istanbul-09-23-2016__patch11b-03-30-20	Upgrade started	2018-08-09 13:00:09
To	glide-kingston-10-17-2017__patch7-06-12-2018	Upgrade finished	2018-08-09 14:14:36

Upgrade History Details | Review Skipped Records

Changes skipped	14
Changes applied	18,783
Changes processed	18,797

- Changes skipped - The total number of records that were different from the previous upgrade and the upgrade component was not applied
- Changes applied - The total number of changes that were applied as a part of this upgrade
- Changes processed - The total number of records that were processed as a part of this upgrade

Figure 12: An upgrade record showing the ***Skipped Changes*** *to* ***Review*** *tab*

   e) Review the name of the table and the sys_id of the record to see exactly what was skipped and where it was skipped from.

   f) Check the type of object that was skipped. If the upgrade skipped an out-of-the-box object, consider reverting that object so it is updated with every upgrade.

Return to workbook checklist

Step 5: Continually improve your instance

So far, all the activities mentioned in this workbook contribute to continuous improvement. Check out the tasks in this stage if you're experiencing:

- Poor list response times
- Poor form load and submit response times
- Poor module response times

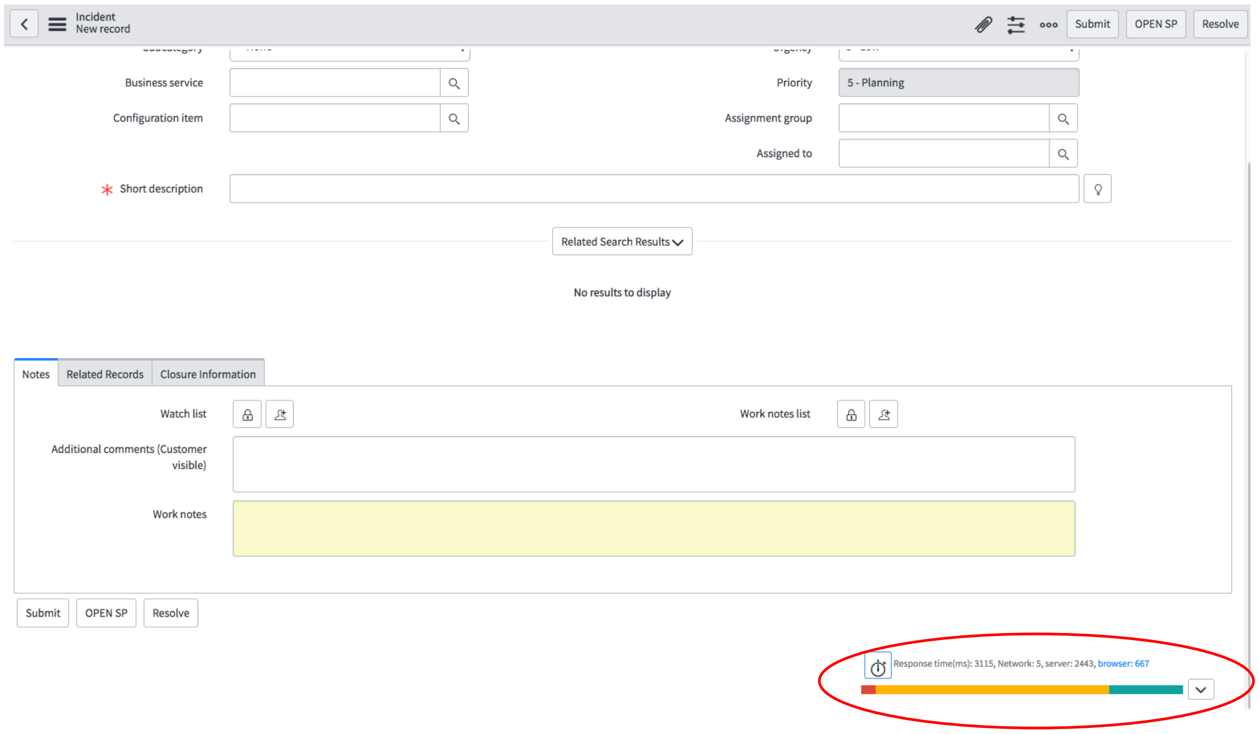
Check your progress:

- ❑ Have you checked form load response time?
- ❑ Have you checked form submit time?
- ❑ Have you monitored modules response times?

If you answered "yes" to the questions above, you have completed the steps outlined in the workbook. If not, complete these actions:

1. Find out if low-response, form load, form submit, and module response times are keeping your instance down.

   a) Navigate to your used forms and select **Incident**.
   b) Select **Create New**.



Priority: 5 - Planning

Response time(ms): 3115, Network: 5, server: 2443, browser: 667

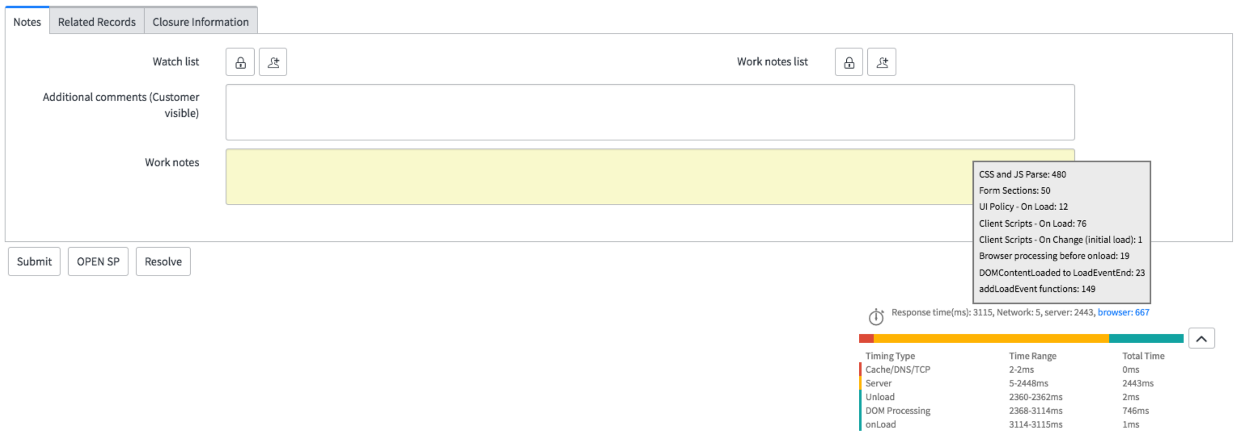
Figure 13: A new incident record with the end-to-end response time bar (bottom right corner)

Return to workbook checklist

c) When the form opens, at the bottom right you will see the details of the end-to-end response time displayed by a colorful bar (see Figure 11), including:

- **Response time (ms) –** The total time between clicking **Create New** and seeing the form load
- **Network –** The total time spent over the wire
- **Server –** The total time spent processing the request on the server
- **Browser –** The total time the browser spent rendering the form, including running the client-side script

d) Click any of the elements of the end-to-end response time to see its details. (See Figure 14.)



CSS and JS Parse: 480
Form Sections: 50
UI Policy - On Load: 12
Client Scripts - On Load: 76
Client Scripts - On Change (initial load): 1
Browser processing before onload: 19
DOMContentLoaded to LoadEventEnd: 23
addLoadEvent functions: 149

Response time(ms): 3115, Network: 5, server: 2443, browser: 667

Timing Type	Time Range	Total Time
Cache/DNS/TCP	2-2ms	0ms
Server	5-2448ms	2443ms
Unload	2360-2362ms	2ms
DOM Processing	2368-3114ms	746ms
onLoad	3114-3115ms	1ms

Figure 14: An incident record showing end-to-end response time details

In the pop-up box, review how much time each section, script, field, etc., took to load. This information helps you identify any potential bad scripts or bottlenecks in your load times so you can address them.

When you have slow form load times, the most common causes are:

- **Related/embedded lists –** To fix this, either look for a bad query or filter or the number of rows you're requesting.
- **A high number of AJAX calls –** You may want to consolidate these into fewer round trips.
- **An inefficient client-side script –** Avoid synchronous AJAX calls or DOM manipulation.

For additional assistance, check out:

Troubleshooting performance

Client transaction timings

Performance and performance debugging

Return to workbook checklist

© 2021 ServiceNow, Inc. All rights reserved. ServiceNow, the ServiceNow logo, Now, Now Platform, and other ServiceNow marks are trademarks and/or registered trademarks of ServiceNow, Inc. in the United States and/or other countries. Other company and product names may be trademarks of the respective companies with which they are associated.