servicenow

# Go full throttle with ServiceNow through instance performance administration

**Increase the upgradability, scalability, and performance of your ServiceNow instance**

## What's in this Success Playbook

This Success Playbook will help you keep your ServiceNow® instance running at optimal performance by explaining how to:

- Perform the activities your instance needs to give you the most value
- Review your log data for errors and warnings
- Maintain your tables for peak performance

## Key takeaways

### The most important things to know

To keep your instance running optimally, you should consider performance management tasks that you can put in place on a daily, weekly, monthly, and quarterly basis.

### The payoff of getting this right

With effective performance management, you can improve both the upgradeability and scalability of your ServiceNow instance.

### What you need to get started

#### Prerequisites

You need admin access to your ServiceNow instance.

## Playbook overview

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 it or customize it, they could have entirely different experiences with the vehicle in the long run.

To keep your instance running optimally, you may need to perform the tasks in all five stages regularly—or you may only need to complete the tasks in one or two of the stages.

Consider your instance and review these five stages to determine if all or only some of them are required to keep your ServiceNow instance running like a well-tuned racecar:

**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

## Terms and definitions

**Instance performance administration –** The instance hygiene activities that ServiceNow administrators can perform to monitor and maintain the overall health of their ServiceNow system

servicenow

## Step 1 – Daily instance performance

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

### KEY INSIGHTS

- Look for errors in your system diagnostics to troubleshoot.
- Review your transactions' total response times.
- Check for users who are selecting more data than they can use at once.

### 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.

1. Go to the System Diagnostics homepage.
2. 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).
3. 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.

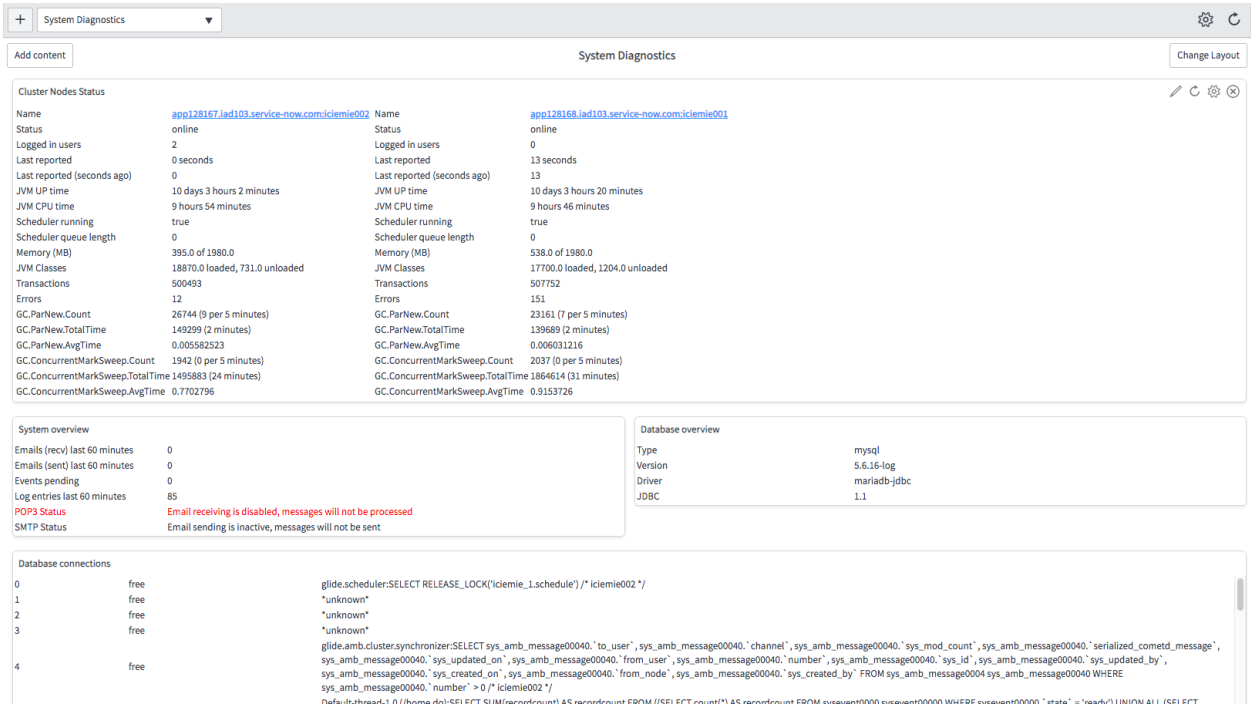



*Figure 1: System Diagnostics homepage*

## 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:

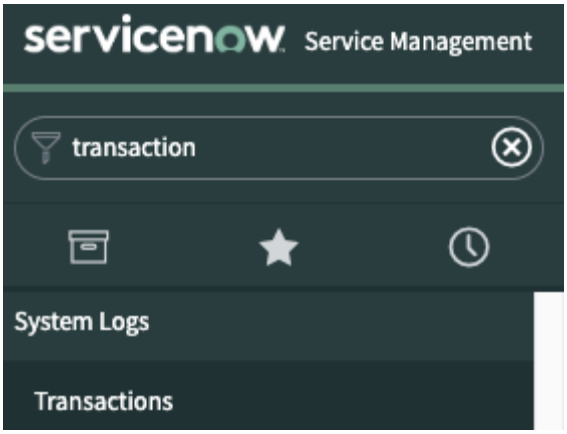



*Figure 2: Access transaction logs through system logs table*

By reviewing all users' transaction information, you can see which transactions are taking more than a specified 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 time 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.

Things that should stand out for you include:

- Whether there is a particular time of day when transactions execute slowly
- Whether 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.
- Whether 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.

## 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 list in

the platform will show that user 100 rows. This includes related and embedded lists on forms as well as the list views where the user set the preference.

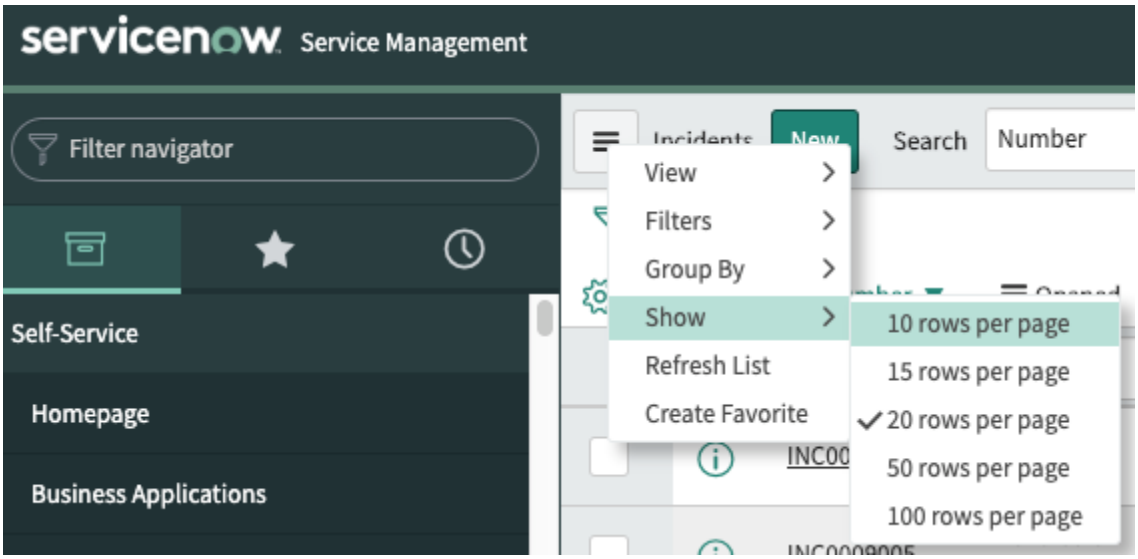



*Figure 3: User chooses to show 100 rows per page*

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 such high demands on the memory required from the JVM to render the lists.

This becomes problematic when a table with many reference fields has to render a list. The platform has to build the relationships for all of 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 option visible for users.

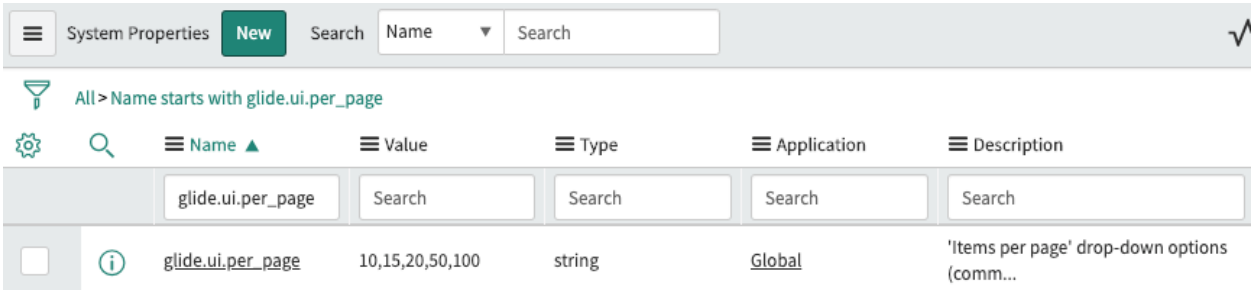



*Figure 4 Glide.ui.per_page property*

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."

## Step 2 – Weekly instance performance

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

### KEY INSIGHTS

- Make sure your scheduled jobs run as they should.
- Find the repeated errors and warnings in your logs.
- Find data users logged excessively and large log files.
- Determine if slow- or long-running jobs are causing issues.

### Review your scheduled jobs

By reviewing your scheduled job activity, you can help ensure your 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 (3,600,000 ms).

1. Navigate to **System Logs > Transactions (Background)**.
2. Apply a filter with the following conditions (see Figure 3):

   **Created > on > This week**

   **URL > starts with > JOB**

   **Response time > greater than > 3600000**

   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 (1800000 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.
3. 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.

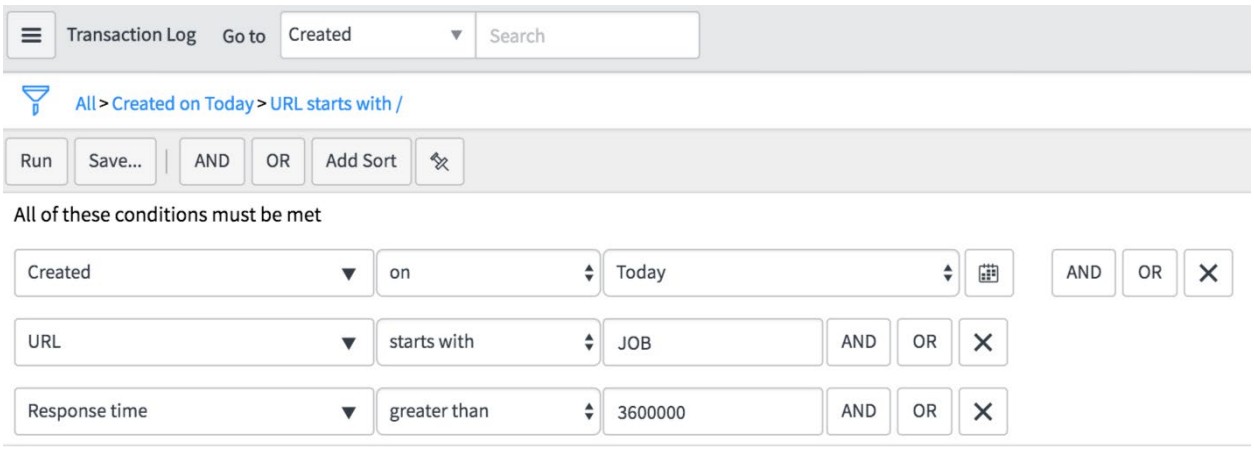



*Figure 5: Filter showing all job transactions created in the current week that took more than 360,000 ms to complete*

## 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.

**Heads up!**

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.

## Check for repeated errors in the error log

1. Navigate to the **System Log**.
2. Select **Errors**.
3. Look for actionable errors as well as frequency within the warning messages.
4. Look for an increase in volume in the number of errors by checking the total number in the top right corner of the screen.
5. 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**.

Look for repeated errors in the warnings log

1. Navigate to the **System Logs**.
2. Select **Warnings**.
3. Look for actionable warnings as well as frequency.
4. Based on the warnings you see, you may be able to search through a **sys_script** for the text output.

## 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.

1. Navigate to **System Logs > Utilities > Node Log File Download**.
2. Apply a **Name starts with local** filter. This will show you all the application logs for the node your session is active on.
3. 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.

---

**EXPERT TIP**

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.

---

## Find log files over 1 GB

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

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

   Note: This may indicate that the **gs.log or gs.print statements**, which were used in sub-production testing, have not been removed. Unnecessary logging makes the tables bulky, which slows maintenance activities, like backups, and also 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.
2. Find the log files that are over 1 GB.

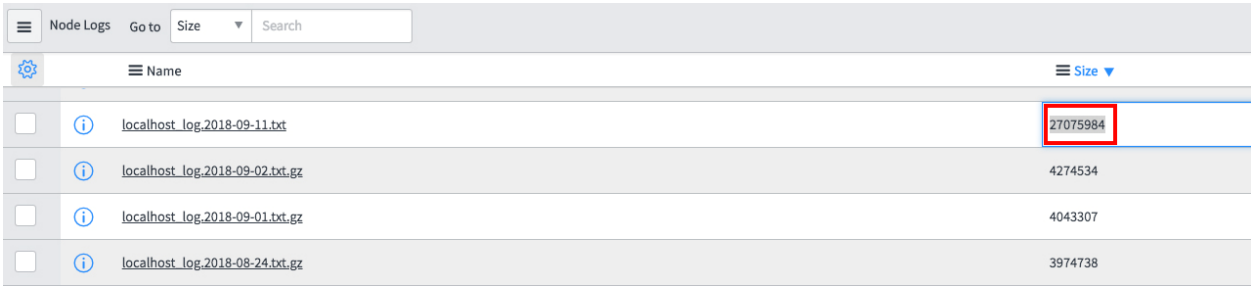



*Figure 6: A log file over 1 GB*

## Find slow-running jobs

1. Navigate to the **System Scheduler**.
2. Select **Slow Job Log**.
3. View the job details in the **URL and Response time** columns.
4. Check the **SQL time column** for the time the job has been in the database.
5. Check the **Business rule time** column for the amount of time the job has been in logic (execution).
6. Right-click the **Response time** column heading and select Sort (z to a).
7. Review the **Response time**, **SQL time**, and **Business rule time** to look for suspiciously long run times.

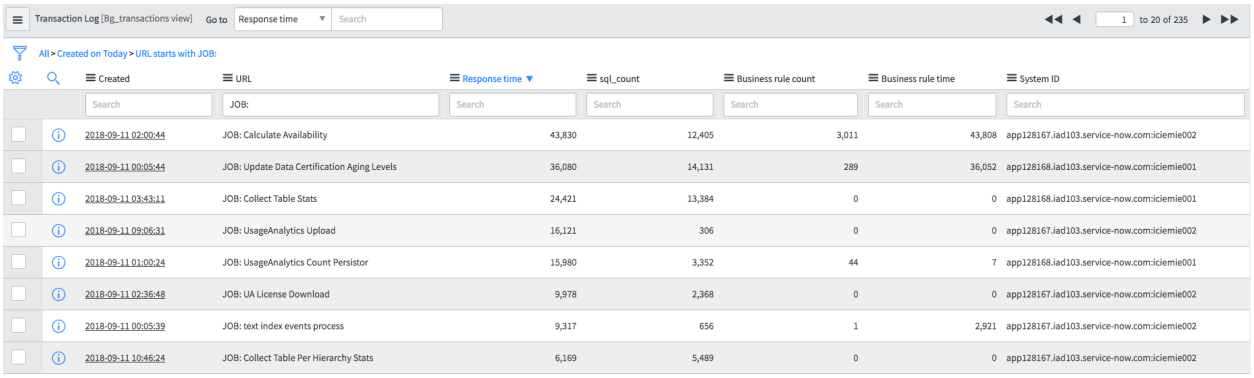



*Figure 7: Example of a Slow Job log*

## Find long-running jobs

1. Navigate to **User Administration**.
2. Select **Active Transactions**.
3. 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.

4. 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**.

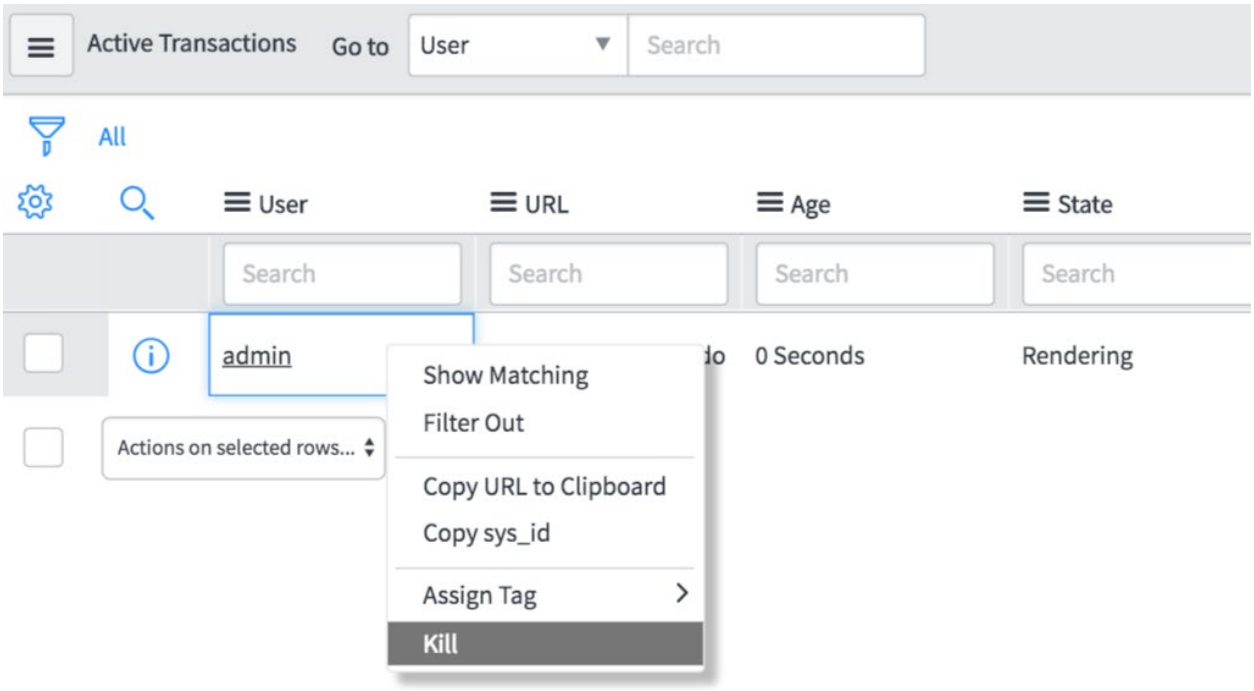



*Figure 8: Right-click menu for killing a stuck job*

5. A confirmation message will appear at the top of the list.

## 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 HI Service Portal for advice on how to investigate the performance of individual transactions.

servicenow

## Step 3 – Monthly instance performance

Don't let slow queries or transient data bog down your instance.

**KEY INSIGHTS**

- Track your table growth to ensure they don't get too large.
- Remove unnecessary records when the data is no longer needed.
- Review your slow queries to see if you need to index or change frequent queries.

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 supported by appropriate indexing so you can accommodate the data growth.

Or your data sets may be growing because 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. 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.

### Monitor your table growth rates

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

- Dramatic changes in size from month to month
- The total number of records in your tables

Follow these steps:

1. Navigate to **System Definition**.
2. Select **Tables**.
3. Your list of tables appears on the screen.

4. Filter the table information typing **u_** in the box. This will show you all of the user-created tables.
5. Continue filtering the table by typing **=NULL** under **Column** name. This shows you all your customized tables (those that aren't out of the box)—the tables you need to check for growth.
6. 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.

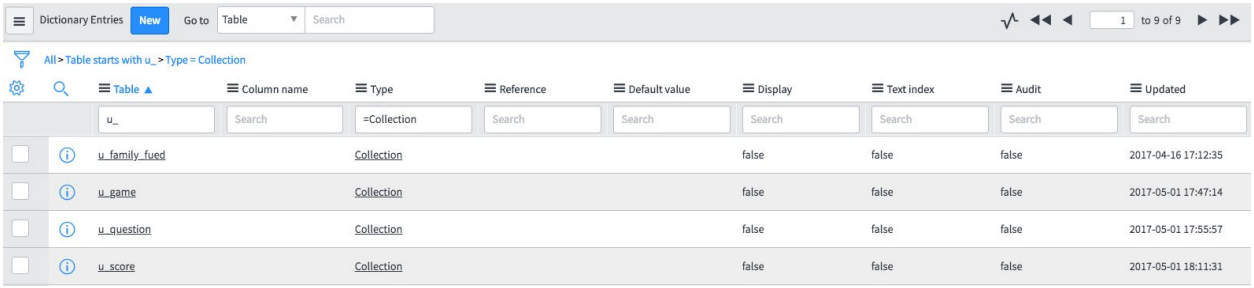



*Figure 9: A table list showing a total of four tables*

7. Look for a dramatic change between the previous month and this month. If you see this, you may need to investigate why the spike occurred.
8. 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 discover that you need to index, contact ServiceNow for support.
- **Request a table rotation from ServiceNow technical support –** For more information, read the next section and our product page on table rotation.
- **Extend your tables –** For more information, read our product page explaining table extension.
- **Clean them by purging rows –** See the next section.

## Clean your tables

When you look at your transient data tables, make sure you see evidence that a sensible data retention policy is being enforced. If a record is a throwaway, there's no need to retain it once it has been processed. Set up a table cleaner to remove the row in a timely fashion.

With the advent of solid-state drives, the table cleaner can comfortably delete approximately one million rows from a table on a daily basis and keep up.

When you need to purge more than one million rows at a time, you may need to request a table rotation. Table rotation is part of the Now Platform functionality and, while it's open to users with the admin role, we recommend sending a ticket to the ServiceNow technical support team to investigate your individual requirements.

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

1. Type incident.list in the navigation field, or type https://<instancename>.service-now.com/incident_list.do into your browser's address bar.
2. A count of the records will display.
3. You may want to amend your show x records preference to 10 or 20 to speed up the list rendering time.

**Heads up!**

If you discover that you need to purge more than one million records, submit a ticket to the ServiceNow tech support team to handle the purge. Purging them yourself could negatively impact your instance.

## 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 that you need to be concerned with are the user created tables, those that begin with "u_."

1. 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.
2. Type **u_** in the **Example** text box.

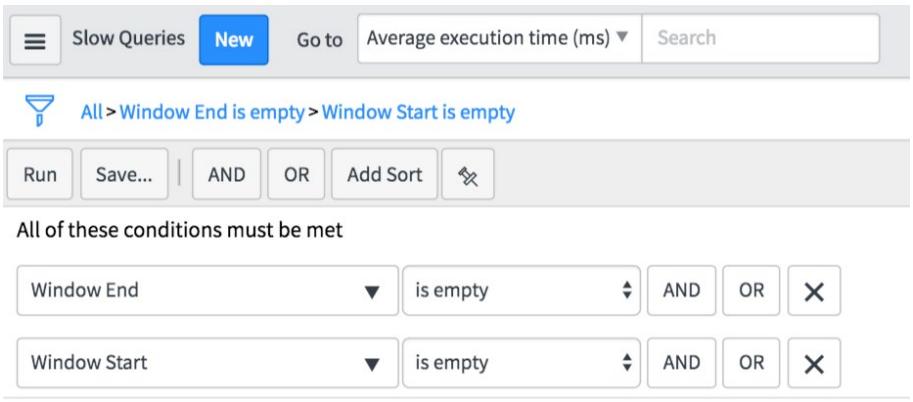



*Figure 10: A Slow Queries log*

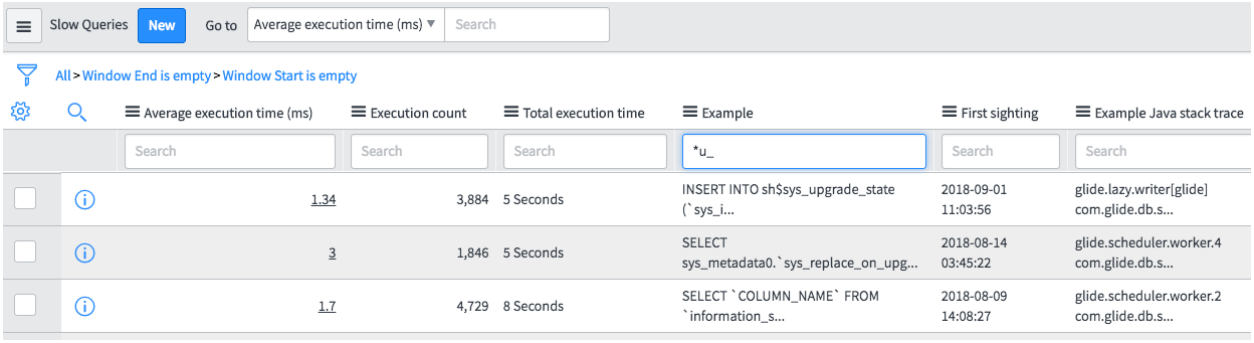



*Figure 11: A Slow Query log showing the Example text box*

The slow query log records the slow 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:

1. Cross-reference which element on the screen requested the information.
2. 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.

## Step 4 – Quarterly instance performance

Running a quarterly review of your upgrade history can tell you a lot about your instance performance.

### KEY INSIGHT

- Find your skipped, inserted, updated, or deleted records to find potential fixes.

Once per quarter, check your instance for any configurations that could impact your upgrades—in other words, the changes or customizations made to out-of-the-box ServiceNow objects.

1. Navigate to **System Diagnostics**.
2. Select **Upgrade History**.
3. A table showing your upgrade history appears. Under the **Upgrade started** column, look for the last upgrade and select it.
4. Within the upgrade record, check the **Skipped Changes to Review** tab to see what was skipped, updated, inserted, or deleted.

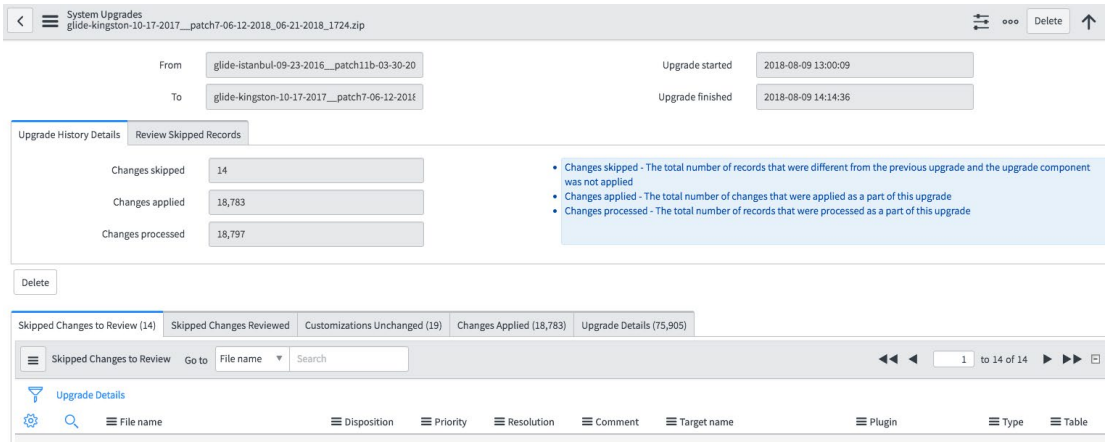



*Figure 12: An upgrade record showing the **Skipped Changes** to **Review** tab*

5. 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.
6. 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.

## Step 5 – Continually improve your instance

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

### KEY INSIGHT

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

All of the activities mentioned in this playbook so far definitely contribute to continual improvement. But you should complete the tasks in this stage if you're experiencing:

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

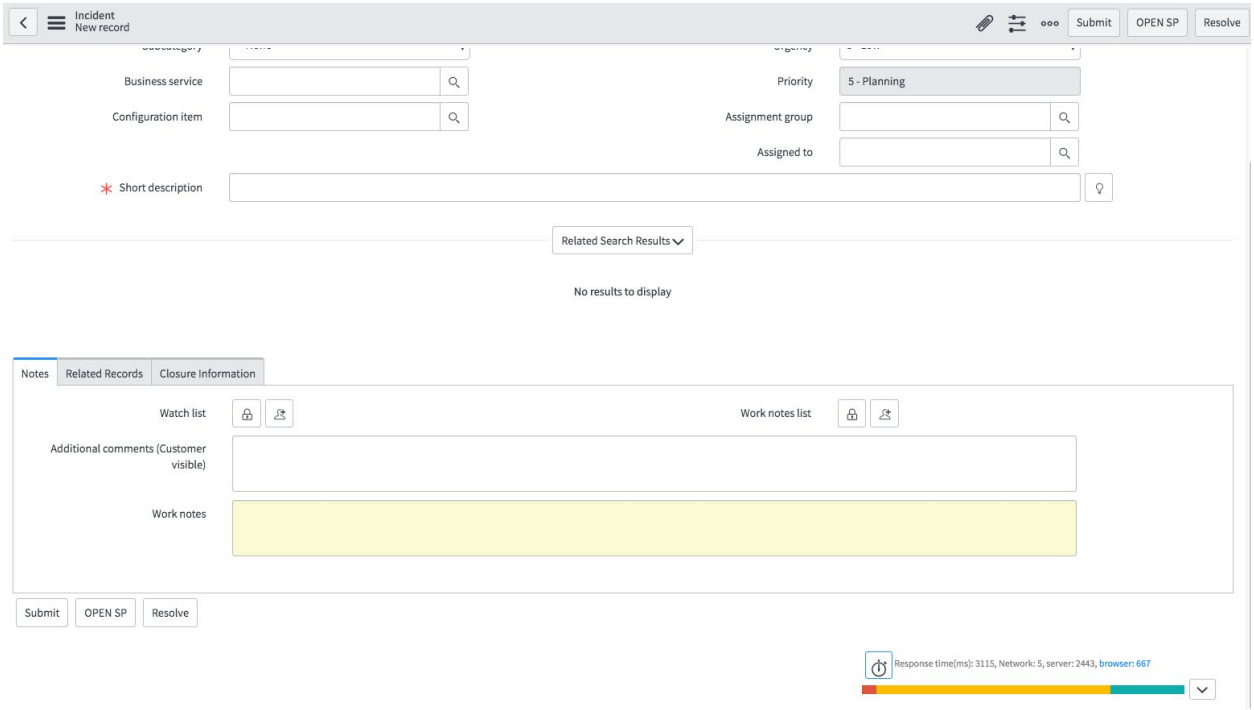



*Figure 13: A new incident record with the end-to-end response time bar (bottom right corner)*

1. Navigate to your used forms and select **Incident**.
2. Select **Create New**.

3. 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 spent rendering the form, including running the client-side script
4. Click any of the elements of the end-to-end response time to see its details. (See Figure 12.)

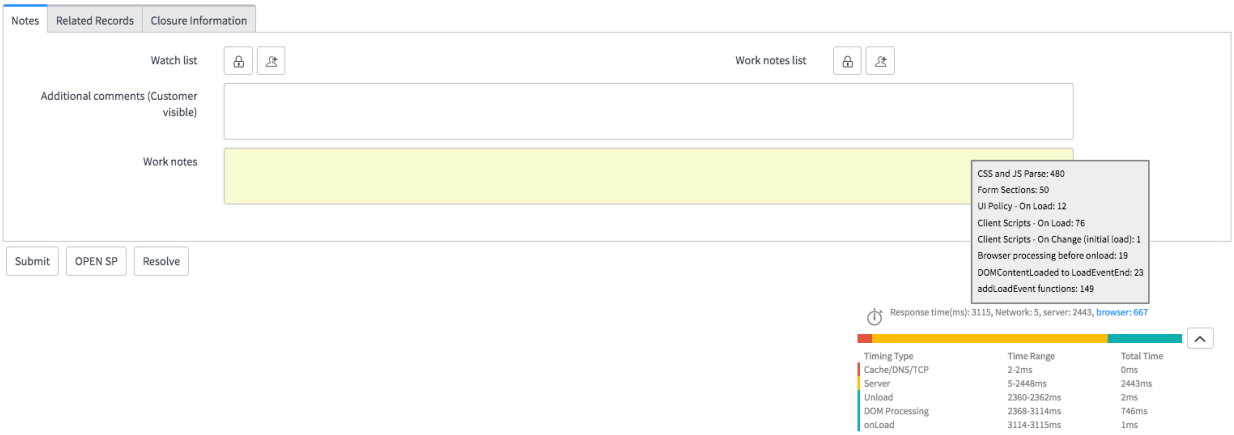



*Figure 14: An incident record showing end-to-end response time details*

5. 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.

## The takeaway

Find out which of these tasks—daily, weekly, monthly, quarterly, or continual—your instance requires. These simple steps will help you monitor and manage your performance, so you truly get the most you can from your ServiceNow instance. You'll increase its upgradability, scalability, and performance—and you'll reduce errors that can lead to unexpected results and data integrity issues.

## Appendix

### Related resources

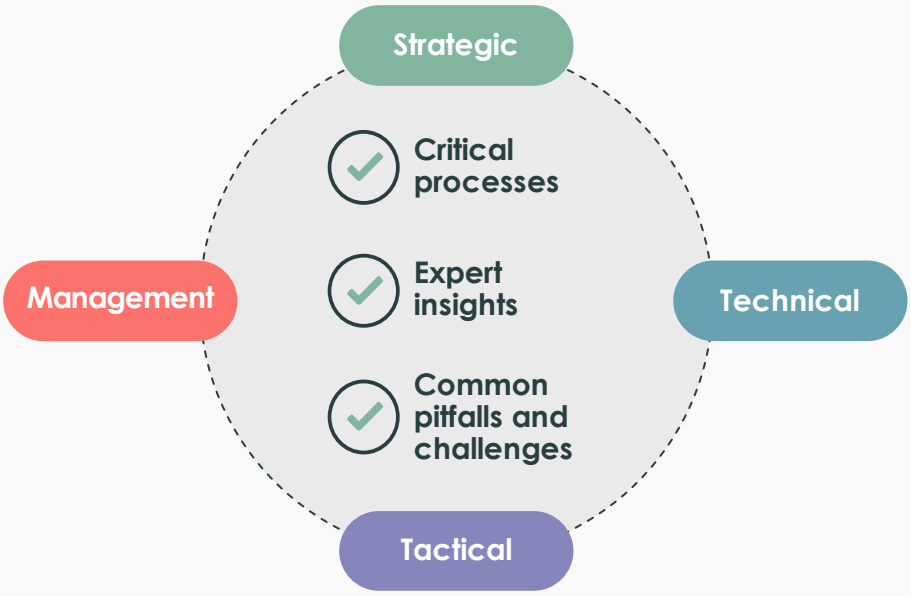
- Troubleshooting performance
- Client transaction timings
- Performance fine-tuning
- Performance and performance debugging

servicenow

# Customer Success Best Practices

**ServiceNow's Best Practice Center of Excellence** provides prescriptive, actionable advice to help you maximize the value of your ServiceNow investment.

## Definitive guidance on a breadth of topics




### Designed for:

- Executive sponsors
- Platform owners and teams
- Service and process owners

## Created and vetted by experts

**Best practice insights** from customers, partners, and ServiceNow teams

Based on **thousands of successful implementations** across the globe

Distilled through a **rigorous process** to enhance your success

## Proven to help you transform with confidence

- Practical
- Actionable
- Value-added
- Expert-validated

**Get started today.**

Visit Customer Success Center.

**Contact your ServiceNow team for personalized assistance.**