CMS QPP Incident Management Process

Overview

This document outlines the process QPP Front End Program must follow during an incident.

Summary

- Anyone may open an incident.
- PagerDuty is the sole mechanism for opening an incident.
- Response teams must designate a single point of contact for all external communication, which happens via an incident response document.
- Response teams work on resolving the incident until they and the affected business owners are satisfied and declare the incident closed.
- After every incident, teams must complete a root cause analysis and take steps to prevent a recurrence.

What is an “incident”?

Software breaks all the time. Sometimes it breaks and is still usable; other times it breaks and systems grind to a halt. For the purpose of this document, an incident is defined as the deviation of a system from its normal operation such that immediate corrective action is required.

Examples of incidents include: a production web service cannot connect to its database and returns an error on every request; a software license expires causing a system to stop responding; a new release suffers from a remotely-exploitable security vulnerability.

Examples of non-incidents include: errors in development or test environments; a UI bug that does not affect a large number of users (escalate to Product Owner for prioritization) ; intermittent performance degradation.

In general terms, an incident:

- requires an immediate response
- may require cross-team collaboration
- disrupts the activities or privacy of end users
- has a significant impact on normal business operations

When in doubt - treat every issue as an active incident until proven otherwise.

How to report an incident

Anyone may report an incident. Early warning is crucial for mitigating the harm an incident may cause, and for quickly returning to normal operations.

The person reporting an incident *must* use PagerDuty to report the incident. PagerDuty is used by all QPP teams to manage on-call response teams, and using it to alert the responsible team will ensure the fastest, most reliable response. PagerDuty allows for tracking the response and escalation, and integrates well with other tools QPP teams use (i.e. Slack, HipChat, and New Relic).

Positive example of reporting an incident

```
To: example@cms-qpp.pagerduty.com
From: reporter@cms.hhs.gov
Subject: many errors on Auth service

Starting at 23:50pm, we've seen a massive increase in errors
logged in the Auth service logs in Splunk.

Here's a link to a search that shows the errors:
https://splunk.example.com/...

New Relic is also showing errors in the same timeframe:
https://rpm.newrelic.com/example-url/...
```

```
Please acknowledge and advise on next steps
```

Notes:

- The email is to PagerDuty, so it will immediately alert the current on-call engineer.
- The body of the email has enough substance, including evidence, for the responder to understand the scope and severity of the issue.

Negative example of reporting an incident

```
To: developer@qppcontractor.com,
other-developer@qppcontractor.com,
unrelated-developer@qppcontractor.com
From: reporter@cms.hhs.gov
Subject: Auth is not forwarding Splunk logs

John Sysadmin just called me and said that Auth Production
systems are not forwarding Splunk logs.  Please contact him as
soon as possible to resolved (sic) this issue.  His phone
number is 419-555-1234.

Thanks,

Jeff Sysadmin
```

Notes:

- This is an actual incident report from WDS Team during Open Enrollment, December 2016. The email was shunted to “Low Priority” by Google Inbox, delaying a response.
- The email was sent to 3 potential responders, who may assume someone else will respond and therefore ignore the email.
- The email doesn’t offer critical details.

Teams are advised to not report incidents by:

- sending an email to a distribution list
- calling, texting, or emailing an individual
- demanding a person or team join a call or webconference

Using these channels will delay the initial response, and in the worst case, may cause the initial incident report to be ignored.

Each QPP team is responsible for maintaining an up-to-date on-call schedule in PagerDuty.

Master List: https://confluence.cms.gov/display/~CPR9/QPP+Monitoring

	Escalation Policy	**Incident Email**
QPP-FE (Ad Hoc)	https://cms-qpp.pagerduty.com/escalation_policies	prod-service-email@cms-qpp.pagerduty.com
QPP-A (Nava)		
QPP-MA (SemanticBits)		
QPP-QS (SemanticBits)		
QPP-BSR (SemanticBits)		
QPP-CT (Flexion)		
GDIT	https://confluence.cms.gov/pages/viewpage.action?pageId=56365010 The GDIT NOC/SOC is available 24x7x365, and can be utilized as a central point of contact to obtain GDIT/AQ Infrastructure resources at any time. • Notification and Escalation of Incidents • 24x7x365 Central point of contact for Infrastructure Resources as needed • Account Resets (Require Documented CMS approval in JIRA Ticket to proceed) • AWS Console / OpenVPN / AWS Active Directory • Escalate Issues to Infrastructure and/or Application Teams (Splunk)	• 540-316-6700 • OpsSupport-Warrenton@gdit.com

Akamai	https://confluence.cms.gov/display/AKAM/Akamai+Support+and+Escalation	https://confluence.cms.gov/display/AKAM/Akamai+Support+and+Escalation

First response

When an incident is reported, the primary goal is to minimize the disruption caused by the incident, then the restoration of normal business operations.

The on-call engineer is responsible for acknowledging the incident report as soon as possible.

The response time goal for initial notification is 15 minutes. PagerDuty schedules should allow for automatic escalation to another responder after the initial 15 minute period.

The on-call engineer responding to the incident should acknowledge the report and immediately review the available information. They should open a new Incident Response Document (see “Sample Incident Response Document” in the appendix for a suggested document structure) and add all available information to it. They should alert at least one additional team member to the incident, and then begin the process of debugging the issue.

The responding team must designate a single Incident Commander, responsible for managing the incident response and communicating the current state of the response. This person is also the single point of contact for all inbound inquiries from other entities, such as business owners.

A good candidate for an Incident Commander is someone who:

- Can communicate in a clear, concise manner
- Understands the basic technical principles involved in the incident and response
- Has access to communication channels, including CMS HipChat and email
- Can devote their entire time and attention to managing the response and managing external entities
- Is capable of prioritizing the work of the response team to focus their efforts on the best path to resolution of the issue

During the response

Effective incident response is predicated on:

- Allowing engineers time and space to investigate the issue and implement a fix
- Real-time communication of status

The engineers must be focused on diagnosing the issue and developing a solution. The initial solution may be a code change, adjusting operational processes (i.e. toggling a feature flag, throttling a service), or reverting to a known good release. Response teams are encouraged to provide incremental resolution, rather than wait for a “big bang” fix to be deployed.

We should follow these rules for any incident response:

1. Do no harm. When responding to an incident, don't do anything that is going make the problem worse or create new problems.
2. Fix as much as possible as quickly as possible. If you can get 60% back in 20 minutes, then you have bought yourself time to get the other 40% back.
3. Communicate as much as possible as often as possible to those who are impacted, without breaking rules 1 and 2.
4. Be skeptical of assumptions. Information should be provably correct before being acted on, and a second assumption should be sought when there is any doubt.

It is imperative that the engineers not be subject to external interruptions. The Incident Commander must act as a proxy between the response team and all external entities.

If an incident spans many hours, it is necessary to rotate teams in order to avoid errors caused by fatigue or stress. If a response team rotation occurs, the Incident Commander is responsible for briefing the new team on status, and for broadcasting the changeover to external entities.

During the incident, the response team should prioritize their efforts in the following order. Work shouldn't begin on a lower priority task until we are moving as fast as possible on all higher priority tasks:

1. Triage and mitigate the issue. In short, make the system work as well as it can under the circumstances, as fast as possible. Finding the root cause of the incident should always take a back seat to stopping the damage caused by it as quickly as possible.
2. Determine the extent of the damage. This is needed for root cause analysis but also for later auditing/postmortems and potentially other damage control activity (e.g. notifying affected parties after a PII leak). It is also a double check that there are no other parts of the system that were affected that were missed in the triage and mitigation step
3. Assess and stabilize the state of the system, and determine a long term plan. Basically, evaluate any short-term fixes in place and any system degradation that is still happening, and ensure that it is sufficient for the period of time needed to put the long-term fix into place. If not, find a more stable short-term solution that will enable us to get to the long term. It is possible but uncommon (for a system of this size) that you will need to do this several times on a longer timescale.

We need to initiate the formal CMS incident response process in parallel with our remediation efforts - this should be started as soon as the incident response process is kicked off.

Communication methods

The Incident Response Document is the primary channel for communicating incident status. By maintaining a live document with a running log of actions the team has taken, external parties can quickly learn the current status without interrupting the response team. The Incident Response Document must be accessible to all parties that are affected by the incident; Google Docs and Confluence is the preferred method for publishing the document.

During the incident response, the Incident Commander must keep the Incident Response Document up to date with all activities. The Incident Commander should update the document on a regular interval, at least once every 2 hours, even if only to note that no new information is available. The Incident Commander may also make regular updates via email or HipChat to external entities on a mutually agreed upon schedule.

The Incident Commander will use the “QPP Incidents” HipChat room to host real-time chat between the various teams involved in the response. In general, HipChat should be preferred to conference calls, as it’s faster and simpler to see the chat history, and to see who is involved in the response.

Closing an incident

An incident is deemed to be resolved when the system has returned to normal operations. In general, both the response team and the affected business owners should reach consensus on the correctness of the resolution before declaring the incident to be over. An effective method for determining if an incident is closed is to announce: “Is anyone opposed to me closing this incident?” As long as anyone has a valid reason to keep the incident open, it remains open.

The response team must remain available and dedicated to resolving the incident as long as the incident has not been declared closed. In general terms, this means that the Incident Commander is available and responsive to all external entities, typically responding to inquiries within 15 minutes. The response team should be dedicated entirely to resolving the incident, and should not be performing any work other than incident response.

Likewise, business owners must also be available for the duration of the incident to approve actions necessary for the response, such as emergency code deployments and configuration changes, and for assisting with managing the external impact of the incident.

After an incident

After the incident is declared closed, the response team must write a post mortem document, using the template here, then spend time filing bugs to track all necessary follow up work. Teams may consider adding additional alerting, tests, or larger changes as necessary.

The response team must also complete a root cause analysis (RCA) that may be shared with other teams within CMS. The RCA should clearly state what the problem was, how the team responded, and what steps they are taking to prevent the issue from recurring. The RCA should be completed within one business day of the incident.

How to write a good postmortem

- **Be precise**. Use precise, clear language. Refer to specific items and actions taken, not vague categories. Don't assume the reader knows to what you are referring. Don't add superfluous adjectives, which often carry judgment and can obscure analysis.
- **Use simple language**. Write at a 6th-grade reading level or lower. (Use the Hemingway app as a writing guide.) Substitute simpler, more direct words for complicated ones.
- **Don't assign blame or use judgmental language**. Incidents occur, outages happen to everyone—the important thing is how we respond to them and learn from them. The point of the document is not to fall on your sword, or drive it into others. Don't use blaming language, on yourself or others.
- **Write in the 3rd-person**. The document should appear to be authored by the team, not by a particular person. Don't use "I", "my", "me", "we", etc.
- **Don't use the passive-voice, except to avoid assigning blame**. It's important to know who was performing the action. Don't write "the logs were copied to a remote machine", instead, "Sally copied the logs to a remote machine". The exception is in lessons-learned-type sections, were the active voice could be read as assigning blame.
- **Be comprehensive**. Error on the side of including more information about the incident rather than less. It may not be apparent at the time of writing what information is relevant to future analyses. Add start and end times to the outage and incident (an outage typically starts before the incident starts--an incident is when the team acknowledges it as such--and ends before the incident ends--after any cleanup or resolving notifications are sent) in the timeline, and mark them clearly for ease of scanning.
- **Don't use casual language**. Avoid informal words and joking.
- **Connect the dots**. Draw connections between actions and outcomes, observations and analysis. Explain how and why the team came to conclusions. Don't leave it to the reader to piece it all together, they won't have enough context.

Credit

This document is deeply influenced by the Google Site Reliability Engineering book, especially the chapter “Managing Incidents”: https://landing.google.com/sre/book/chapters/managing-incidents.html.

Appendix

Sample Incident Response Document

TITLE (incident #ID)

Date: $DATE

Authors: $AUTHORS

Status: $STATUS

Summary: A short (two sentence) summary of what went wrong.

Impact: Quantified statement about the impact of the incident.

Timeline *(all times EDT)*

$DATE

- $TIME: $ACTION

Root Causes: What contributed

Trigger: What caused the incident

Resolution: Steps taken to resolve the incident (add internal ticket tracking)

Detection: How did we learn about the incident (alerts, user reports, etc)

Action Items:

Action item	Type	Owner	Issue

Lessons Learned

What went well

-

What went wrong

-

Where we got lucky

-

Sample incident status update

This can help direct interested parties to the Live doc

$TITLE
Current Incident Commander: $NAME - $EMAIL
Status: $STATUS
Live Document: $DOC_LINK