ServiceNow portfolio · Test evidence report

Outage Shield

Version 1 test evidence

Scoped ServiceNow application · GitHub and Cloudflare status integrations · Personal Developer Instance

Evidence captured 3 November 2026 · Report revised 7 November 2026

Ezaz Kabir

ServiceNow CSA · CIS-ITSM · CIS-Data Foundations

Case study	ezazkabir.com/projects/outage-shield
Source code	github.com/ezazkabir/outage-shield
LinkedIn	linkedin.com/in/ezazkabir
Email	ezaz.kabir.uk@gmail.com

Summary

Outage Shield polls the GitHub and Cloudflare status APIs, maps the components they report to internal Service Offerings and assesses a preliminary business risk. It then takes a proportionate Incident Management action: an investigative Incident, an in-place escalation, or a Major Incident proposal for a person to decide on. It also tells subscribers when an offering may be affected, and when that clears.

This report records the Version 1 tests run on 3 November 2026. Each test states what was expected, what the screenshots show and where the evidence stops.

Live tests ran with Incident automation switched off, so live vendor data never wrote Incidents. Controlled tests exercised the Incident logic with synthetic vendor payloads.

Results at a glance

ID	Test	Type	Expected outcome	Result	Figure
TC-01	Vendor components mapped to offerings	Configuration	Active mappings link components to internal offerings	Pass	01
TC-02	Medium risk opens an investigative Incident	Controlled	New Incident at P3 (Impact 2, Urgency 2)	Pass	02a, 02b
TC-03	Higher risk escalates the same Incident	Controlled	Same record raised to P2; no second Incident	Pass	02b
TC-04	Very High risk proposes a Major Incident	Controlled	P2 Incident and a native proposal; no declaration	Pass	03a, 03b
TC-05	Ambiguous data reviewed; invalid data rejected	Controlled	Review for ambiguous data; invalid JSON rejected; no escalation	Pass · 4/4	04
TC-06	Scheduled polling observed (GitHub 7/7, Cloudflare 6/7 cycles)	Live	Both APIs polled every five minutes	Qualified pass	05
TC-07	Notifications when impact changes	Live	One queued notification per change; none when unchanged	Pass	06

Qualified pass: the expected behaviour was observed, with an exception recorded on the test page.

About this report

How the tests were run

Two kinds of test

Controlled tests (TC-02 to TC-05) passed synthetic vendor incidents, marked `OS-TEST`, to the application's own processing methods. This exercised the real decision and Incident logic on demand, without waiting for a genuine GitHub or Cloudflare outage.

Live tests (TC-06 and TC-07) used the real GitHub and Cloudflare status APIs through the scheduled job. Incident automation was switched off for these runs, so live data could reach mappings, events and email, but not Incident records.

Reading the evidence

Each test gives its purpose, the expected result, what the evidence shows and where it stops. Figures are cropped from the original screenshots; boxes and shading mark the fields the text refers to. Nothing else in the screenshots was changed; the full-resolution originals are linked in the evidence index.

Figure numbers match the evidence numbers on the case study page. Times are as displayed in the instance.

Decision policy under test

Preliminary risk	Response	Target
Low	Keep monitoring; no Incident	—
Medium	Open an investigative Incident	P3
High	Open, or escalate the existing primary Incident	P2
Very High	Investigative Incident and a Major Incident proposal	P2

Risk combines vendor severity with the affected offering's criticality. Outage Shield sets impact and urgency; ServiceNow's priority matrix turns them into a priority.

Test environment

Item	Detail
Instance	ServiceNow Personal Developer Instance (PDI)
Application	Outage Shield, scoped application `x_1951923_outage_0`, version 1.0.0
ITSM features	Incident Management with Major Incident Management; standard Impact/Urgency priority matrix
Vendor sources	GitHub and Cloudflare public status APIs (REST, JSON)
Polling	Scheduled job “Outage Shield - Vendor Status Polling”, every 5 minutes from a configured start of 19:04:21, run as the Outage Shield Automation user
Safety switch	System property `x_1951923_outage_0.enable_itsm_automation`, set to `false` for the live tests
Test data	Synthetic vendor incidents (`OS-TEST-…`); TEST offering “Outage Shield – Critical Website Availability (TEST)”; subscribers abel.tuter@example.com and admin@example.com (PDI demo users)
Test user	Controlled tests ran as System Administrator, so that name appears on the test records
Source code	github.com/ezazkabir/outage-shield

Evidence index

Figure	Shows	Full-resolution screenshot
01	Component Mappings list	component-mappings.png
02a	INC0010007 form	incident-p2.png
02b	INC0010007 audit history	incident-audit-p3-to-p2.png
03a	INC0010008 form	major-incident-details.png
03b	Major Incident tab	major-incident-proposed.png
04	Safeguard test output	failure-safeguards.png
05	System Log poll results	scheduled-polling.png
06	Email outbox	outbound-notifications.png

TC-01 · Configuration

Vendor components are mapped to internal Service Offerings

Purpose	Expected result	Result
Give the poller business context: which internal offering a reported vendor component could affect.	Active mappings for the GitHub and Cloudflare components in scope. Test mappings kept, but switched off.	Pass

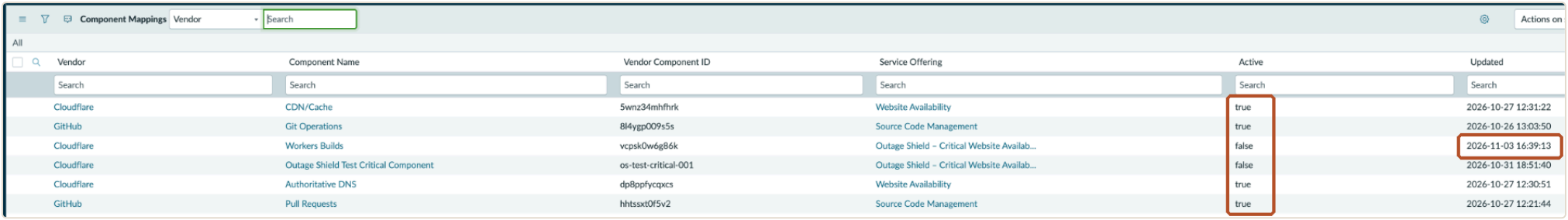


Figure 01 · Component Mappings list. Boxes: the Active column, and the last update of the Workers Builds test mapping.

Vendor	Component	Service Offering	Active	Last updated
GitHub	Git Operations	Source Code Management	true	26 Oct, 13:03:50
GitHub	Pull Requests	Source Code Management	true	27 Oct, 12:21:44
Cloudflare	CDN/Cache	Website Availability	true	27 Oct, 12:31:22
Cloudflare	Authoritative DNS	Website Availability	true	27 Oct, 12:30:51
Cloudflare	Workers Builds	Outage Shield – Critical Website Availability (TEST)	false	3 Nov, 16:39:13
Cloudflare	Outage Shield Test Critical Component	Outage Shield – Critical Website Availability (TEST)	false	31 Oct, 18:51:40

What the evidence shows

Four mappings are active and connect the GitHub and Cloudflare components in scope to Source Code Management and Website Availability. The two test mappings point to the TEST offering and are switched off.

The Workers Builds mapping was last updated at 16:39:13, nine seconds before the poll that cleared the TC-07 test impact.

> **Limits.** This list shows the configuration exists. TC-07 shows a mapping being used by the live poller.

TC-02 · Native ITSM · Controlled test

Medium risk opens an investigative P3 Incident

Purpose	Expected result	Result
Check that a Medium preliminary business risk opens a native Incident to investigate potential, not confirmed, impact.	New Incident linked to the affected offering, with Impact 2 – Medium and Urgency 2 – Medium giving P3 through the priority matrix. The description says internal impact is unconfirmed.	Pass

This screenshot shows P2 because it was captured after escalation. Figure 02a was taken after TC-03 raised the Incident to Priority 2 – High. The original P3 state (Urgency 2 – Medium, Priority 3 – Moderate, created at 15:58:56) is in the audit history, Figure 02b on page 5.

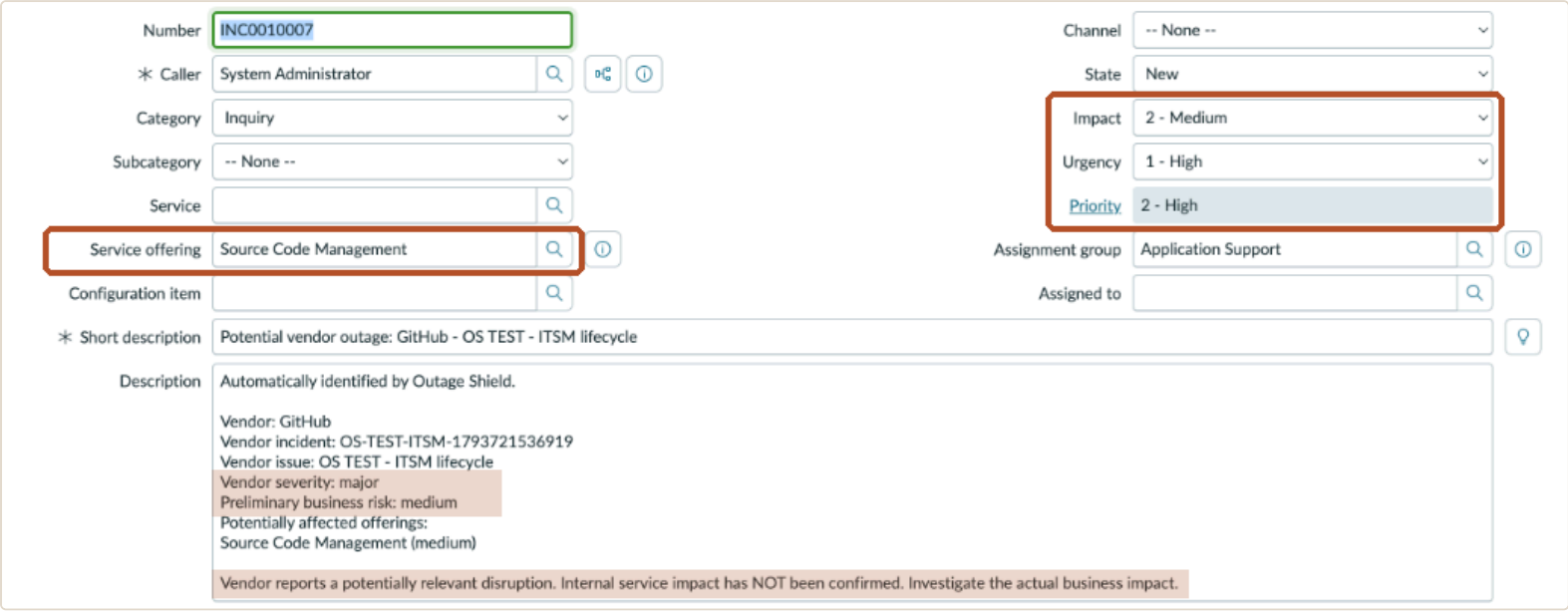


Figure 02a · INC0010007, captured after the TC-03 escalation. Boxes: service offering; impact, urgency and priority. Shading: vendor severity, preliminary risk and the unconfirmed-impact statement.

What the evidence shows

INC0010007 is linked to the Source Code Management offering and assigned to Application Support. It was created at Priority 3 – Moderate (Figure 02b).

The description records the vendor (GitHub), the vendor severity (major) and the preliminary business risk (medium), and states that internal service impact has not been confirmed.

Limits. A controlled payload (`OS-TEST-ITSM-1793721536919`), not a live GitHub incident. Live polling ran with Incident automation off.

TC-03 · Native ITSM · Controlled test

Higher risk escalates the same Incident from P3 to P2

Purpose	Expected result	Result
Check that a higher risk for the same vendor outage raises the existing primary Incident instead of opening another.	INC0010007 raised to P2 by changing Urgency from 2 to 1, with Impact unchanged. No second Incident for the same outage.	Pass

Label	Old	New	Type	Update number	Update time	User name
Urgency	2 - Medium	1 - High	Audit	1	2026-11-03 15:58:58	System Administrator
Priority	3 - Moderate	2 - High	Audit	1	2026-11-03 15:58:58	System Administrator
Urgency		2 - Medium		0	2026-11-03 15:58:56	System Administrator
Upon reject		Cancel all future Tasks		0	2026-11-03 15:58:56	System Administrator
Upon approval		Proceed to Next Task		0	2026-11-03 15:58:56	System Administrator
Effective number		INC0010007		0	2026-11-03 15:58:56	System Administrator
Domain		global		0	2026-11-03 15:58:56	System Administrator
Task type		Incident		0	2026-11-03 15:58:56	System Administrator
State		New		0	2026-11-03 15:58:56	System Administrator
Short description		Potential vendor outage: GitHub - OS TES...		0	2026-11-03 15:58:56	System Administrator
Severity		3 - Low		0	2026-11-03 15:58:56	System Administrator
Service offering		Source Code Management		0	2026-11-03 15:58:56	System Administrator
Reopen count		0		0	2026-11-03 15:58:56	System Administrator
Reassignment count		0		0	2026-11-03 15:58:56	System Administrator
Priority		3 - Moderate		0	2026-11-03 15:58:56	System Administrator
Opened by		System Administrator		0	2026-11-03 15:58:56	System Administrator
Opened		2026-11-03 15:58:56		0	2026-11-03 15:58:56	System Administrator

Figure 02b · INC0010007 audit history. Boxes: the escalation (update 1) and the initial urgency and priority (update 0).

What the evidence shows

Time	Update	Change on INC0010007
15:58:56	0	Created with Urgency 2 – Medium, Priority 3 – Moderate
15:58:58	1	Urgency 2 – Medium → 1 – High
15:58:58	1	Priority 3 – Moderate → 2 – High

Outage Shield finds the existing Incident through the vendor outage's primary Incident reference, with a correlation ID as a fallback. The outbox (Figure 06) is consistent with no duplicate: between 15:58 and 16:39 the only “Incident … was created” notifications are for INC0010007 and INC0010008.

> **Limits.** This figure evidences the in-place escalation. Controlled test output also confirmed that a repeated report for the same outage reused the primary Incident, but that output isn't reproduced in this report.

TC-04 · Governance · Controlled test

Very High risk proposes a Major Incident for a person to decide

Purpose	Expected result	Result
Check the Very High policy: open an investigative P2 Incident and submit a native Major Incident proposal, without declaring one.	Incident at P2 (Impact 2, Urgency 1). Major Incident state Proposed, not Accepted.	Pass

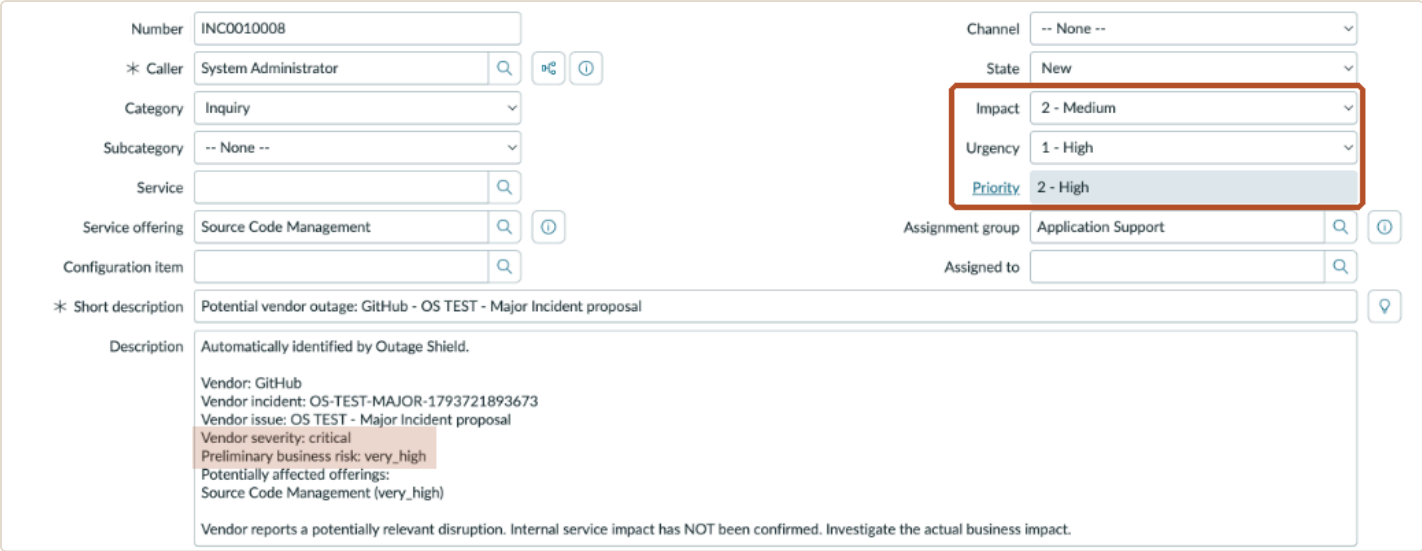


Figure 03a · INC0010008. Impact, urgency and priority; vendor severity and preliminary risk.

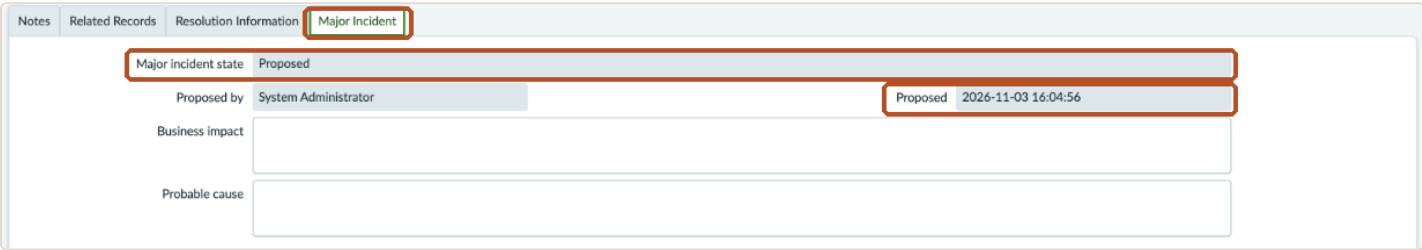


Figure 03b · Major Incident tab on INC0010008. State Proposed, with the proposal time.

What the evidence shows

INC0010008 opened at 16:04:55, linked to Source Code Management, with Impact 2 – Medium, Urgency 1 – High and Priority 2 – High.

Its description records a critical GitHub vendor incident (`OS-TEST-MAJOR-1793721893673`) and a preliminary business risk of very high, and states that internal impact has not been confirmed.

The Major Incident tab shows state **Proposed**, proposed at 16:04:56. Business impact and probable cause are left blank.

Nothing was accepted or declared automatically. Declaring a Major Incident stays a human decision by design.

> **Limits.** The proposal path only; accepting or rejecting the proposal was not part of this test.

TC-05 · Negative paths · Controlled, read-only test

Ambiguous data is routed to review; invalid payloads are rejected

Purpose	Expected result	Result
Check that insufficient, conflicting or malformed vendor data never produces a confident impact classification or an automatic escalation.	Unmapped, conflicting or missing component data is routed to review. An invalid JSON payload is rejected by the parser. No case leads to an Incident action.	Pass · 4 of 4

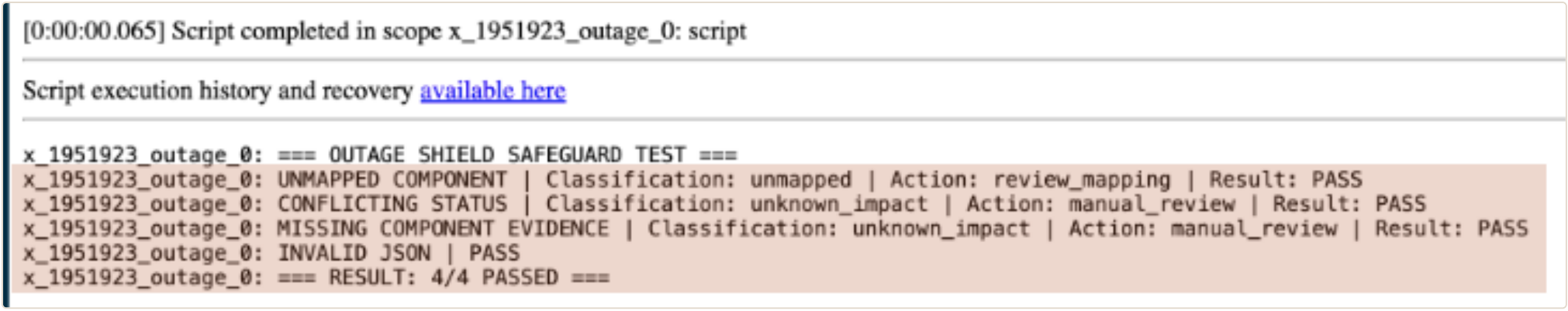


Figure 04 · Safeguard test output from Scripts – Background in the application scope. Shading: the four results and the summary line.

Scenario	Expected	Observed classification	Observed action	Result
Unmapped degraded component	Routed to mapping review	`unmapped`	`review_mapping`	Pass
Conflicting component status	Routed to manual review	`unknown_impact`	`manual_review`	Pass
Missing component evidence	Routed to manual review	`unknown_impact`	`manual_review`	Pass
Invalid JSON payload	Rejected at parsing	—	Rejected by the parser	Pass

What the evidence shows

The three ambiguous cases were routed to review: the unmapped component to mapping review, and the conflicting and missing component evidence to manual review. The invalid JSON payload was rejected by the parser before any assessment. None of the four produced a confident impact classification or an escalation.

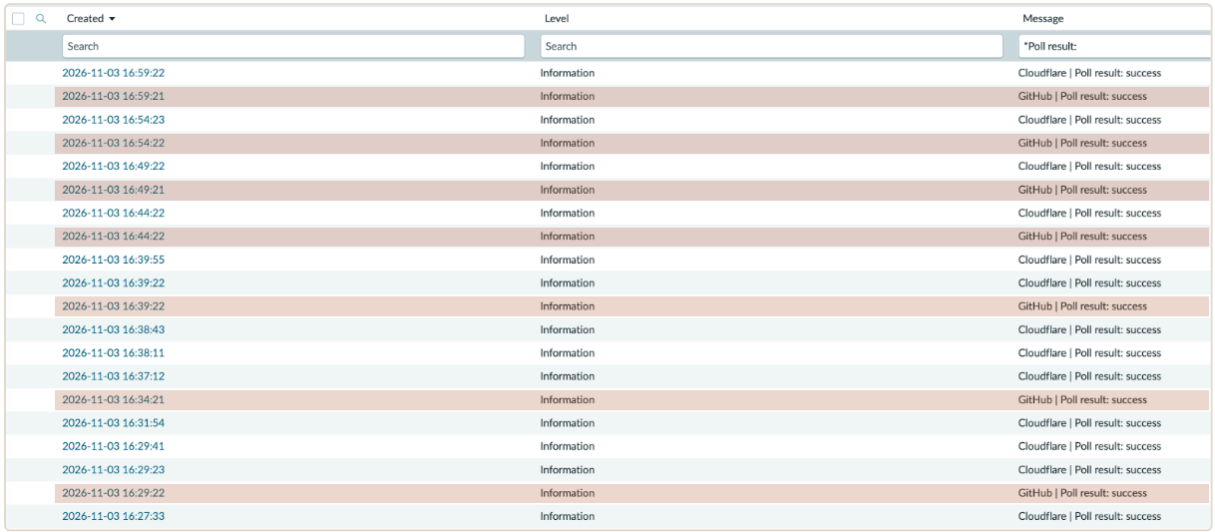
All four cases passed in a repeated, read-only run of the actual `VendorOutageProcessor` methods, completing in 0.065 seconds.

> **Limits.** HTTP timeouts, unavailable vendor endpoints and retry behaviour at the transport layer were not simulated.

TC-06 · Integration · Live test

Scheduled polling observed for GitHub and Cloudflare

Purpose	Expected result	Result
Check that polling runs on its own, on schedule, and records each result.	A GitHub and a Cloudflare “Poll result” entry every five minutes, at the times the schedule predicts (21 seconds past :04, :09, :14 and so on).	**Qualified pass** GitHub 7 of 7 cycles Cloudflare 6 of 7 cycles



Created	Level	Message
Search	Search	*Poll result:
2026-11-03 16:59:22	Information	Cloudflare | Poll result: success
2026-11-03 16:59:21	Information	GitHub | Poll result: success
2026-11-03 16:54:23	Information	Cloudflare | Poll result: success
2026-11-03 16:54:22	Information	GitHub | Poll result: success
2026-11-03 16:49:22	Information	Cloudflare | Poll result: success
2026-11-03 16:49:21	Information	GitHub | Poll result: success
2026-11-03 16:44:22	Information	Cloudflare | Poll result: success
2026-11-03 16:44:22	Information	GitHub | Poll result: success
2026-11-03 16:39:55	Information	Cloudflare | Poll result: success
2026-11-03 16:39:22	Information	Cloudflare | Poll result: success
2026-11-03 16:39:22	Information	GitHub | Poll result: success
2026-11-03 16:38:43	Information	Cloudflare | Poll result: success
2026-11-03 16:38:11	Information	Cloudflare | Poll result: success
2026-11-03 16:37:12	Information	Cloudflare | Poll result: success
2026-11-03 16:34:21	Information	GitHub | Poll result: success
2026-11-03 16:31:54	Information	Cloudflare | Poll result: success
2026-11-03 16:29:41	Information	Cloudflare | Poll result: success
2026-11-03 16:29:23	Information	Cloudflare | Poll result: success
2026-11-03 16:29:22	Information	GitHub | Poll result: success
2026-11-03 16:27:33	Information	Cloudflare | Poll result: success

Figure 05 · System Log filtered on “Poll result:”. GitHub entries shaded. Source column (all `x_1951923_outage_0`) cropped.

Scheduled cycle	GitHub	Cloudflare
16:29	16:29:22	16:29:23
16:34	16:34:21	no entry
16:39	16:39:22	16:39:22
16:44	16:44:22	16:44:22
16:49	16:49:21	16:49:22
16:54	16:54:22	16:54:23
16:59	16:59:21	16:59:22
Cycles polled	**7 of 7**	**6 of 7**

Each GitHub entry falls within one second of the time the schedule predicts. That regularity, matching the job's start time of 19:04:21, is consistent with the configured five-minute schedule. All 20 entries report success.

Seven more Cloudflare entries (16:27:33, 16:29:41, 16:31:54, 16:37:12, 16:38:11, 16:38:43 and 16:39:55) fall between cycles, around the TC-07 notification test.

> **Why qualified.** There is no Cloudflare entry for the 16:34 cycle, and the reason isn't recorded in this evidence. Log entries also don't record what started each run; the link to the scheduler rests on the timing.

TC-07 · Communications · Live test

Each impact change queues one notification; unchanged polls queue none

Purpose	Expected result	Result
Check that a real, poller-driven change in offering impact notifies subscribers once when it starts and once when it clears. A temporary Cloudflare Workers Builds mapping to the TEST offering was switched on and off while live polling ran.	One “Potential Service Disruption” email when impact starts and one “Potential Service Impact Cleared” when it clears, each queued for both test subscribers. No email from polls where nothing changed.	Pass · queued

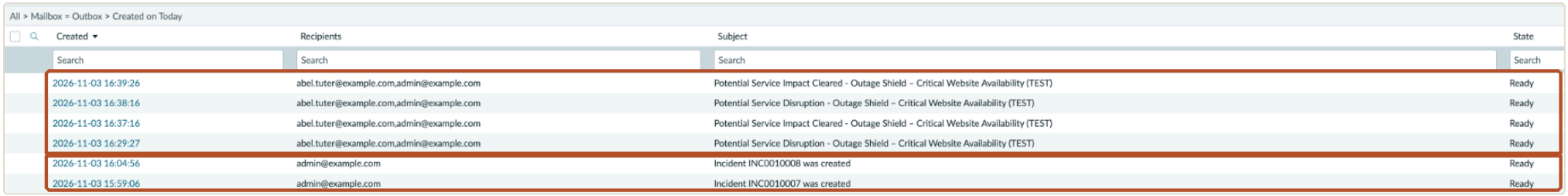


Figure 06 · Outbox, 15:59 to 16:39. Boxes: the four Outage Shield notification records, and ServiceNow's own Incident-created notifications. Earlier entries from that day are cropped.

Time	Figure	Event
16:29:23	05	Cloudflare poll, at a scheduled cycle time
16:29:27	06	Potential Service Disruption queued for both test subscribers
16:29:41	05	Cloudflare poll · no email
16:31:54	05	Cloudflare poll · no email
16:37:12	05	Cloudflare poll
16:37:16	06	Potential Service Impact Cleared queued
16:38:11	05	Cloudflare poll
16:38:16	06	Potential Service Disruption queued
16:38:43	05	Cloudflare poll · no email
16:39:13	01	Workers Builds mapping last updated (now inactive)
16:39:22	05	Cloudflare poll, at a scheduled cycle time
16:39:26	06	Potential Service Impact Cleared queued

What the evidence shows

Two disruption-and-clearance cycles generated four notification records, each queued for both test subscribers four to five seconds after a Cloudflare poll. The three polls taken between a disruption and its clearance (16:29:41, 16:31:54 and 16:38:43) queued nothing, so an ongoing impact doesn't repeat the notification.

The final clearance followed the mapping change at 16:39:13 and the next poll at 16:39:22, a scheduled cycle time.

The same outbox shows ServiceNow's own notifications for INC0010007 (15:59:06) and INC0010008 (16:04:56), matching TC-02 and TC-04.

> **Queued, not delivered.** State Ready means each notification was generated and queued in the outbox. Delivery to a real inbox was not tested; the recipients are example.com demo addresses.

Overall result and next steps

What Version 1 shows, and what it doesn't

Version 1 has a repeatable test trail for its central design: vendor alerts are read in the context of internal services, uncertain evidence goes to review, malformed data is rejected, Incident actions follow risk and governance rules, and subscribers hear when potential impact starts and clears.

Demonstrated

- Live polling and parsing of GitHub and Cloudflare status data, consistent with the five-minute schedule: GitHub in 7 of 7 cycles, Cloudflare in 6 of 7 (TC-06, qualified pass).
- Vendor components mapped to internal Service Offerings (TC-01).
- A native investigative Incident at P3, escalated in place to P2 on the same record (TC-02, TC-03).
- A native Major Incident proposal for Very High risk, with no declaration (TC-04).
- Unmapped, conflicting and missing component data routed to review, and an invalid JSON payload rejected, 4 of 4 (TC-05).
- Four notification records queued for both test subscribers across two disruption-and-clearance cycles, none repeated while impact continued (TC-07).

Not evidenced in this report

- Low risk, where Outage Shield keeps monitoring and opens no Incident.
- High risk opening a new P2 Incident directly, rather than escalating one.
- Recovery handling when the vendor reports its incident resolved.
- HTTP timeouts, retries and upstream transport failures.
- Email delivery beyond the outbox.
- Incident writes from live polling, which were switched off for the live tests.
- Production deployment and business-user acceptance.

By design

- Major Incidents are proposed, never declared automatically.
- Impact is described as potential until someone has investigated it.
- Uncertain data is routed for manual review and malformed payloads are rejected; neither escalates.
- Priority is set through the platform's Impact/Urgency matrix, so it follows the organisation's own configuration.

Next steps for Version 1.1

1. **Automated tests.** Move TC-02 to TC-05 into an Automated Test Framework (ATF) suite, and add Low risk, direct High risk and recovery handling.
2. **Transport failures.** Add timeouts, retries and error logging for vendor API calls, with tests for each.
3. **End-to-end run.** One live run with `enable_itsm_automation` switched on against the TEST offering.
4. **Incident data.** Set a category instead of the default Inquiry, populate Service from the offering's parent service, and pre-fill Major Incident business impact from the risk assessment.
5. **Configuration.** Move the assignment group from code into a system property.
6. **Delivery.** Confirm notifications arrive in a real test mailbox.