Internet-Draft
Intended status: Standards Track
Expires: May 25, 2026

Michael DALLA RIVA
Janeway and Wildman Limited (UK)

November 25, 2025

Service Status Resource Format for Web Services
draft-dallariva-web-service-status-json-00

Abstract

This specification defines a standard JSON format for service status resources. It provides a machine-readable format for overall service health indicators, component-level status information with criticality classification, geographic scope identification, and structured incident reporting. This standard enables interoperability between status monitoring tools and service providers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 23, 2026.

Table of Contents

1. Introduction

Many web services provide status pages to communicate service health and incident information to their users. While RFC 8631 [RFC8631] defined the "status" link relation to enable discovery of these status resources, it intentionally left the format of the status data unspecified, allowing flexibility for different implementations.

This has resulted in a fragmented ecosystem where each service provider implements their own status format, making it difficult for:

- Monitoring tools to aggregate status across multiple services
- Users to understand the scope and severity of incidents
- Automated systems to respond to service degradations
- Organizations to compare service reliability metrics
- Security teams to detect coordinated infrastructure attacks
- Dependent services to assess cascading failure impact

This specification defines a standard JSON format for status resources, building on lessons learned from existing implementations.

1.1. Critical Use Cases

This standard addresses several critical operational scenarios:

Cascading Failure Response: When a major infrastructure provider experiences an outage (e.g., AWS, Cloudflare, Azure), thousands of dependent services need to rapidly assess impact. Machine-readable criticality and component status enables automated assessment of which services are affected.

Multi-Provider Incident Correlation: During coordinated DDoS attacks or widespread network issues, security teams need to correlate incidents across multiple providers. Standardized format enables automated threat intelligence aggregation.

Automated Failover Decision Making: Services using multiple providers need to automatically fail over during outages. Machine-readable status with clear criticality and scope enables automated decision making without human intervention.

Incident Communication to End Users: Services dependent on infrastructure providers need to communicate clearly to their users. Standardized geographic scope and impact metrics enable accurate user communication.

Security Incident Detection: Unusual patterns across multiple providers may indicate coordinated attacks. Standardized format enables security monitoring tools to detect these patterns.

Disaster Recovery Planning: Organizations need to assess their exposure when providers have incidents. Clear component criticality enables rapid risk assessment.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the following terms:

Service: A web service or API that provides functionality to users.

Component: A discrete functional unit of a service (e.g., "API", "Database", "CDN").

Incident: An unplanned event that degrades or disrupts service functionality.

Status Resource: A resource linked via the "status" link relation that provides service health information.

1.3. Problem Statement

Analysis of 50+ production status page implementations reveals significant interoperability challenges:

- Format fragmentation: Services use incompatible JSON structures, HTML-only pages, or proprietary APIs
- Missing criticality semantics: No standard way to distinguish critical infrastructure (API, database) from non-critical features (billing UI, analytics dashboard)
- Geographic scope ambiguity: Regional incidents (e.g., "SMS delays in Kenya") indistinguishable from global outages
- Inconsistent impact reporting: No standard for communicating percentage of users affected or estimated resolution time
- Integration burden: Monitoring tools must implement custom parsers for each service provider

These problems result in:

- False alarms when non-critical components fail
- Delayed incident response due to unclear severity
- User confusion about whether issues affect them
- Wasted engineering time on integration and maintenance

1.4. Real-World Impact: Major Internet Outages

Recent major incidents demonstrate the critical need for standardized status communication:

Cloudflare Database Update (November 2025): A database schema update caused widespread service disruption affecting millions of websites. Human error in the deployment process led to cascading failures. Standardized criticality classification would have helped downstream services understand impact severity immediately.

Facebook/Meta BGP Incident (2021): Incorrect BGP route advertisements took down Facebook, Instagram, and WhatsApp for 6 hours, affecting 3.5 billion users globally. The lack of

standardized status reporting made it difficult for dependent services to communicate impact to their users.

AWS us-east-1 Outages (October 2025): Multiple major outages in AWS's largest region caused cascading failures across thousands of services. Services dependent on AWS struggled to communicate which of their features were affected versus operational.

Azure Active Directory Outage (2023): Authentication failures prevented users from accessing Microsoft 365, Teams, and thousands of dependent services. Geographic scope was initially unclear, causing global panic for what was primarily a regional issue.

These incidents share common challenges:

- Cascade effect: One service's outage impacts hundreds of dependent services
- Communication breakdown: No standard way to report which components are affected
- User confusion: Unclear whether issues are global or regional
- Delayed response: Time wasted parsing different status formats
- Amplified impact: Lack of criticality data causes overreaction

1.5. Security Considerations: Malicious Attacks

The increasing frequency of coordinated attacks on internet infrastructure highlights additional needs:

DDoS Attacks on DNS Providers: Distributed denial-of-service attacks against major DNS providers (Dyn 2016, Google Cloud 2017, AWS Route 53 2019) demonstrate the need for rapid, machine-readable status updates to enable automated failover.

Targeted Infrastructure Attacks: State-sponsored and criminal groups increasingly target internet infrastructure providers. Standardized status reporting enables:

  * Faster detection of coordinated multi-provider attacks
  * Automated correlation of incidents across services
  * Rapid triggering of disaster recovery procedures
  * Better communication during security incidents

Supply Chain Attacks: When infrastructure providers are compromised, dependent services need clear, machine-readable status to assess their exposure and communicate with their users.

Without standardized status reporting:

- Security teams cannot aggregate threat intelligence from status pages
- Automated defense systems cannot trigger based on provider status
- Incident response is delayed by manual status checking
- Users receive inconsistent security guidance

A standardized format enables:

- Automated monitoring for coordinated attacks
- Rapid failover when providers are under attack
- Machine-readable security incident disclosure
- Correlation of incidents across multiple providers
- Faster mean time to detection (MTTD) and response (MTTR)

2. Status Resource Format

2.1. Overall Structure

A status resource MUST be a JSON object containing the following top-level fields:

version (string, REQUIRED): The version of this specification. Value MUST be "1.0" for this specification.

service (object, REQUIRED): Information about the service. See Section 2.2.

status (object, REQUIRED): Overall service status. See Section 2.2.

components (array, OPTIONAL): Array of component status objects. See Section 2.3.

incidents (array, OPTIONAL): Array of active incident objects. See Section 2.4.

updated_at (string, REQUIRED): ISO 8601 timestamp of last status update.

Example:

```
{
  "version": "1.0",
  "service": { ... },
  "status": { ... },
  "components": [ ... ],
  "incidents": [ ... ],
  "updated_at": "2025-11-23T17:00:00Z"
}
```

2.2. Service Status Object

The "service" object MUST contain:

name (string, REQUIRED): Human-readable service name.

url (string, OPTIONAL): URL of the main service.

The "status" object MUST contain:

indicator (string, REQUIRED): Overall status indicator. MUST be one of:

* "operational" - All systems functioning normally
* "degraded" - Some systems experiencing issues
* "down" - Major service disruption

description (string, OPTIONAL): Human-readable status description.

The "status" object MAY contain:

affected_percentage (number, OPTIONAL): Percentage of users affected (0.0-100.0).

estimated_resolution (string, OPTIONAL): ISO 8601 timestamp of estimated resolution.

Example:

```
{
  "service": {
    "name": "Example CDN",
```

```
    "url": "https://example.com"
  },
  "status": {
    "indicator": "degraded",
    "description": "Elevated API error rates in US-East region",
    "affected_percentage": 5.2,
    "estimated_resolution": "2025-11-23T18:00:00Z"
  }
}
```

2.3. Component Object

Each component object MUST contain:

id (string, REQUIRED): Unique identifier for the component.

name (string, REQUIRED): Human-readable component name.

status (string, REQUIRED): Current component status. MUST be one of:

* "operational" - Functioning normally
* "degraded" - Performance issues
* "partial_outage" - Partially unavailable
* "major_outage" - Completely unavailable
* "under_maintenance" - Planned maintenance

Each component object SHOULD contain:

criticality (string, RECOMMENDED): Component importance level. MUST be one of:

* "critical" - Core service functionality (API, database, auth)
* "high" - Important but non-essential (dashboard, notifications)
* "medium" - Optional features (analytics, reporting)
* "low" - Cosmetic features (documentation, marketing pages)

scope (string, RECOMMENDED): Geographic scope of component. MUST be one of:

* "global" - Worldwide availability
* "regional" - Specific geographic region(s)
* "local" - Single location or facility

Each component object MAY contain:

description (string, OPTIONAL): Component description.

regions (array, OPTIONAL): Array of ISO 3166-1 alpha-2 country codes if scope is "regional".

group (string, OPTIONAL): Component group identifier for logical organization.

Example:

```
{
  "id": "api-prod",
  "name": "Production API",
  "status": "operational",
  "criticality": "critical",
  "scope": "global",
  "description": "REST API for customer applications"
}
```

```
{
  "id": "sms-kenya",
  "name": "SMS Delivery - Kenya",
  "status": "degraded",
  "criticality": "medium",
  "scope": "regional",
  "regions": ["KE"],
  "description": "SMS message delivery through Kenyan carriers"
}
```

2.4. Incident Object

Each incident object MUST contain:

id (string, REQUIRED): Unique incident identifier.

name (string, REQUIRED): Human-readable incident title.

status (string, REQUIRED): Current incident status. MUST be one of:

* "investigating" - Issue identified, investigation ongoing
* "identified" - Root cause found
* "monitoring" - Fix applied, monitoring stability
* "resolved" - Issue fully resolved

impact (string, REQUIRED): Incident severity. MUST be one of:

* "none" - No user impact (informational)
* "minor" - Minimal user impact
* "major" - Significant user impact
* "critical" - Severe widespread impact

started_at (string, REQUIRED): ISO 8601 timestamp when incident started.

Each incident object SHOULD contain:

updates (array, RECOMMENDED): Array of incident update objects with fields:

* timestamp (string, REQUIRED): ISO 8601 update time
* status (string, REQUIRED): Status at time of update
* message (string, REQUIRED): Human-readable update

affected_components (array, RECOMMENDED): Array of component IDs affected by this incident.

Each incident object MAY contain:

estimated_resolution (string, OPTIONAL): ISO 8601 timestamp of estimated resolution.

resolved_at (string, OPTIONAL): ISO 8601 timestamp when resolved.

affected_percentage (number, OPTIONAL): Percentage of users affected.

Example:

```
{
  "id": "inc-2025-11-23-001",
  "name": "API Response Time Degradation",
  "status": "monitoring",
  "impact": "minor",
```

```
  "started_at": "2025-11-23T16:45:00Z",
  "affected_components": ["api-prod"],
  "affected_percentage": 12.5,
  "updates": [
    {
      "timestamp": "2025-11-23T16:45:00Z",
      "status": "investigating",
      "message": "We are investigating elevated API response times."
    },
    {
      "timestamp": "2025-11-23T17:00:00Z",
      "status": "identified",
      "message": "Issue identified in database connection pool. Deploying fix."
    },
    {
      "timestamp": "2025-11-23T17:15:00Z",
      "status": "monitoring",
      "message": "Fix deployed. Monitoring for stability."
    }
  ]
}
```

3. Status Values and Semantics

3.1. Component Status Values

This specification defines five standard component status values:

operational: Component is functioning within normal parameters. Response times, error rates, and availability meet SLA targets.

degraded: Component is functional but experiencing performance issues. Users may experience slower response times or elevated error rates, but core functionality remains available.

partial_outage: Component is partially unavailable. Some features or geographic regions are affected while others remain operational.

major_outage: Component is completely unavailable or non-functional. All users are affected.

under_maintenance: Component is undergoing planned maintenance. This status SHOULD only be used for scheduled maintenance windows.

3.2. Criticality Levels

Components SHOULD be classified by criticality to enable monitoring tools to distinguish between critical infrastructure failures and minor feature issues:

critical: Core service functionality that, if degraded, prevents users from accessing primary service features. Examples: API endpoints, authentication systems, primary databases, CDN edge networks.

high: Important functionality that significantly impacts user experience but does not prevent core service usage. Examples: web dashboards, notification systems, secondary databases, caching layers.

medium: Optional features that enhance user experience but are not essential for basic service operation. Examples: analytics

dashboards, reporting tools, administrative interfaces.

low: Cosmetic or informational features that do not impact service functionality. Examples: marketing pages, documentation sites, status page itself, billing interfaces.

Rationale: This classification enables monitoring systems to:

- Generate alerts only for critical component failures
- Avoid alarm fatigue from non-critical issues
- Calculate more accurate service availability metrics
- Provide users with appropriate context about impact

3.3. Geographic Scope

Components and incidents SHOULD specify geographic scope to distinguish between global outages and regional issues:

global: Affects all users worldwide. This SHOULD be the default for components that serve all regions.

regional: Affects users in specific geographic regions. The "regions" array MUST be populated with ISO 3166-1 alpha-2 country codes.

local: Affects a single data center, facility, or point of presence.

Rationale: This classification enables:

- Users to determine if an incident affects their region
- Monitoring tools to display region-specific status
- Service providers to maintain accurate global vs regional SLA metrics
- Automated systems to route traffic away from affected regions

4. Examples

4.1. Minimal Status Resource

The simplest valid status resource:

```
{
  "version": "1.0",
  "service": {
    "name": "Example Service"
  },
  "status": {
    "indicator": "operational"
  },
  "updated_at": "2025-11-23T17:00:00Z"
}
```

4.2. Complete Status Resource

A comprehensive status resource with all optional fields:

```
{
  "version": "1.0",
  "service": {
    "name": "Example Cloud Platform",
    "url": "https://example.com"
  },
```

```
  "status": {
    "indicator": "degraded",
    "description": "Elevated error rates in API",
    "affected_percentage": 8.5
  },
  "components": [
    {
      "id": "api",
      "name": "REST API",
      "status": "degraded",
      "criticality": "critical",
      "scope": "global",
      "description": "Primary REST API endpoints"
    },
    {
      "id": "dashboard",
      "name": "Web Dashboard",
      "status": "operational",
      "criticality": "high",
      "scope": "global",
      "description": "Customer management dashboard"
    },
    {
      "id": "cdn-asia",
      "name": "CDN - Asia Pacific",
      "status": "operational",
      "criticality": "critical",
      "scope": "regional",
      "regions": ["JP", "SG", "AU", "IN"]
    }
  ],
  "incidents": [
    {
      "id": "inc-001",
      "name": "API Error Rate Increase",
      "status": "monitoring",
      "impact": "minor",
      "started_at": "2025-11-23T16:00:00Z",
      "affected_components": ["api"],
      "affected_percentage": 8.5,
      "estimated_resolution": "2025-11-23T18:00:00Z",
      "updates": [
        {
          "timestamp": "2025-11-23T16:00:00Z",
          "status": "investigating",
          "message": "Investigating elevated API error rates"
        },
        {
          "timestamp": "2025-11-23T16:30:00Z",
          "status": "identified",
          "message": "Database connection pool exhaustion identified"
        },
        {
          "timestamp": "2025-11-23T17:00:00Z",
          "status": "monitoring",
          "message": "Connection pool limits increased, monitoring"
        }
      ]
    }
  ],
  "updated_at": "2025-11-23T17:00:00Z"
}
```


4.3. Regional Incident

Status resource showing a regional carrier issue:


```
{
  "version": "1.0",
  "service": {
    "name": "SMS Gateway"
  },
  "status": {
    "indicator": "operational",
    "description": "Regional carrier issues (limited impact)"
  },
  "components": [
    {
      "id": "sms-global",
      "name": "SMS - Global",
      "status": "operational",
      "criticality": "critical",
      "scope": "global"
    },
    {
      "id": "sms-ke",
      "name": "SMS - Kenya (Telkom)",
      "status": "degraded",
      "criticality": "medium",
      "scope": "regional",
      "regions": ["KE"]
    }
  ],
  "incidents": [
    {
      "id": "inc-002",
      "name": "SMS Delays to Telkom Kenya",
      "status": "investigating",
      "impact": "minor",
      "started_at": "2025-11-23T15:00:00Z",
      "affected_components": ["sms-ke"],
      "updates": [
        {
          "timestamp": "2025-11-23T15:00:00Z",
          "status": "investigating",
          "message": "SMS delivery delays to Telkom Kenya. Working with carrier partne
r."
        }
      ]
    }
  ],
  "updated_at": "2025-11-23T17:00:00Z"
}
```


5. IANA Considerations

5.1. Media Type Registration

This document requests registration of the media type "application/vnd.service-status+json":

Type name: application

Subtype name: vnd.service-status+json

Required parameters: None

Optional parameters: version - Version of status format (default "1.0")

Encoding considerations: UTF-8

Security considerations: See Section 6

Interoperability considerations: None

Published specification: This document

Applications that use this media type: Status monitoring systems, service health dashboards, incident management tools

6. Security Considerations

6.1. Information Disclosure

Status resources may reveal information about service architecture, dependencies, and vulnerabilities. Service providers SHOULD:

- Avoid exposing internal system names or implementation details
- Use generic component names (e.g., "Database" not "PostgreSQL v14.2 on prod-db-01")
- Consider rate limiting or authentication for detailed status APIs
- Exclude security-sensitive incidents from public status pages

6.2. Denial of Service

Status resources SHOULD be:

- Cached aggressively to reduce load during incidents
- Served from separate infrastructure to maintain availability
- Rate limited to prevent abuse

6.3. Data Integrity

Status resources SHOULD be served over HTTPS to prevent tampering. Automated monitoring systems SHOULD validate:

- TLS certificates
- JSON schema conformance
- Timestamp freshness to detect stale data

6.4. Attack Detection and Response

Standardized status reporting provides significant security benefits:

6.4.1. Coordinated Attack Detection

Machine-readable status across multiple providers enables automated detection of coordinated attacks:

DDoS Pattern Recognition: When multiple DNS providers or CDNs show degraded performance simultaneously, automated systems can recognize coordinated DDoS campaigns.

Supply Chain Attack Indicators: Unusual incident patterns across infrastructure providers may indicate supply chain compromise.

Geographic Attack Correlation: Multiple providers reporting regional issues in the same geographic area may indicate targeted infrastructure attacks.

Example Detection Scenario:

1. Monitoring system observes Cloudflare, Akamai, and Fastly all report "degraded" status for "CDN" components

2. All incidents started within 15-minute window
3. All show scope: "global"
4. Automated alert: "Possible coordinated CDN attack detected"
5. Security team investigates and coordinates response

Without standardized format, this correlation requires manual effort and is often detected too late.

6.4.2. Automated Failover During Attacks

Clear criticality and scope enable automated security responses:

- Automatically fail over from attacked provider to backup
- Route traffic away from regions under attack
- Trigger incident response procedures based on criticality
- Disable non-critical features to preserve core functionality

Example Failover Scenario:

Primary DNS provider status:

```
{
  "components": [{
    "name": "DNS",
    "status": "major_outage",
    "criticality": "critical",
    "scope": "global"
  }]
}
```

Automated response:
1. Detect critical + major_outage + global
2. Automatically switch DNS to secondary provider
3. Notify operations team
4. Continue monitoring for recovery

6.4.3. Incident Response Coordination

During major incidents, standardized format enables:

Clear Communication: Security teams can rapidly assess which components are affected and communicate clearly with stakeholders.

Dependency Mapping: Organizations can automatically determine their exposure when infrastructure providers have incidents.

Prioritization: Critical component failures trigger immediate response while low criticality issues are queued.

Historical Analysis: Standardized incident reporting enables post-incident analysis of attack patterns and provider reliability.

6.5. Mitigating False Security Alerts

Standardized criticality reduces security alert fatigue:

- Security systems should only alert on critical component failures
- Non-critical issues (billing, UI) should not trigger security response
- Regional issues should not trigger global security alerts
- Maintenance windows should be distinguished from attacks

This reduces mean time to detection (MTTD) by eliminating noise and enabling security teams to focus on genuine threats.

6.6. Privacy Considerations

Status resources should not reveal:

- Individual customer data or affected user identities
- Specific attack vectors or vulnerabilities being exploited
- Internal security measures or monitoring capabilities
- Detailed incident response procedures

The "affected_percentage" field should provide aggregate metrics only and should not enable identification of specific users or customers.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[RFC8631] Wilde, E., "Link Relation Types for Web Services", RFC 8631, DOI 10.17487/RFC8631, July 2019, <https://www.rfc-editor.org/info/rfc8631>.

Appendix A. Acknowledgments

This specification builds on the work of Erik Wilde (RFC 8631) and practical experience from existing service status implementations.

Special thanks to the operators of monitoring systems who provided feedback on the challenges of status page integration.

Appendix B. Migration from Existing Formats

Many existing status page implementations use similar JSON structures. Common mappings include:

- Overall status indicators (operational, degraded, etc.)
- Component-level status tracking
- Incident impact levels

This specification adds standardization for:
- Component criticality classification
- Geographic scope identification
- Structured region information

Services with existing status APIs can:

1. Maintain existing endpoints for backward compatibility
2. Add new endpoint implementing this specification
3. Gradually migrate clients to the standardized format
4. Use content negotiation to serve multiple formats

Author’s Address

Michael DALLA RIVA
Janeway and Wildman Limited (UK)
ietf@janewayandwildman.com