10. Monitoring and Event Management

10.1 Ziele und Grundlagen des Monitoring and Event Management

> **Kurz erklärt**
>
> Monitoring and Event Management sorgt dafür, dass der Zustand von IT-Services und IT-Systemen kontinuierlich überwacht wird.
>
> Ziel ist es, Abweichungen möglichst früh zu erkennen, bevor daraus Incidents oder Serviceunterbrechungen entstehen.
>
> Ereignisse (Events) werden gesammelt, bewertet und – je nach Bedeutung – automatisch verarbeitet oder an die zuständigen Teams weitergeleitet.

Was ist Monitoring and Event Management?

Monitoring and Event Management ist eine ITIL Practice zur kontinuierlichen Überwachung von Services, Systemen und Infrastruktur.

Dabei werden Informationen über den aktuellen Zustand gesammelt und ausgewertet.

Beispiele:

- Server erreichbar?
- CPU-Auslastung zu hoch?
- Festplatte fast voll?
- Backup erfolgreich?
- Zertifikat läuft bald ab?
- Datenbank antwortet?
- Netzwerkverbindung aktiv?
- Webservice erreichbar?
- Temperatur im Serverraum normal?

Das Ziel besteht darin, Probleme möglichst frühzeitig zu erkennen.

Warum Monitoring wichtig ist

Ohne Monitoring werden viele Probleme erst bemerkt,

wenn Benutzer sie melden.

Dadurch entstehen:

- längere Ausfallzeiten,
- mehr Incidents,
- höhere Kosten,
- unzufriedene Benutzer,
- größere Geschäftsrisiken.

Ein gutes Monitoring erkennt viele Probleme bereits,

bevor Benutzer betroffen sind.

Ziele des Monitoring and Event Management

Die wichtigsten Ziele sind:

- frühzeitige Fehlererkennung,
- schnelle Reaktion,
- Verfügbarkeit erhöhen,
- Ausfälle vermeiden,
- automatische Benachrichtigung,
- Automatisierung unterstützen,
- Servicequalität verbessern,
- Trends erkennen,
- Ursachenanalyse erleichtern.

Monitoring ersetzt dabei keine Fehlerbehebung.

Es liefert die notwendigen Informationen.

Monitoring und Event Management unterscheiden

Diese beiden Begriffe werden häufig gemeinsam verwendet,

beschreiben jedoch unterschiedliche Aufgaben.

Monitoring	Event Management
sammelt Messwerte	bewertet Ereignisse
überwacht Systeme	entscheidet über Reaktionen
erkennt Veränderungen	verarbeitet Events
liefert Daten	löst Aktionen aus

Monitoring erzeugt Events.

Event Management verarbeitet diese.

Was ist ein Event?

Ein Event ist jede erkennbare Veränderung des Zustands eines Services oder Configuration Items.

Ein Event kann beispielsweise entstehen durch:

- Anmeldung,
- Fehler,
- Neustart,
- Backup,
- Alarm,
- Warnung,
- Hardwareausfall,
- erfolgreiche Aktualisierung,
- Zertifikatsablauf,
- Überschreiten eines Grenzwerts.

Nicht jedes Event bedeutet automatisch ein Problem.

Event ist nicht gleich Incident

Ein häufiger Irrtum:

> Jedes Event ist ein Incident.

Das stimmt nicht.

Beispiel:

Ein Server startet nach einer geplanten Wartung neu.

Das Monitoring erzeugt ein Event.

Da die Änderung geplant war,

liegt kein Incident vor.

Beispiel eines echten Incidents

Das Monitoring erkennt:

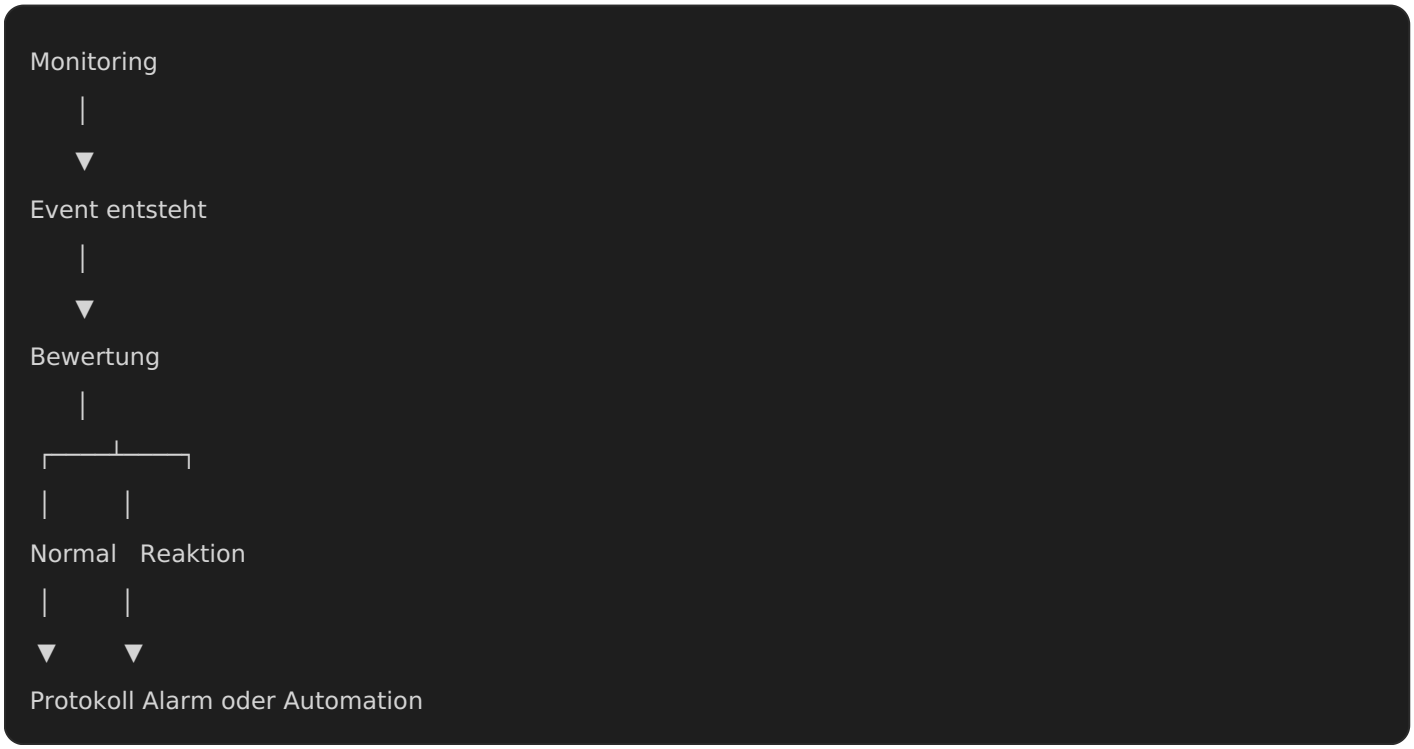
- Webserver nicht erreichbar.

Es existiert:

- keine geplante Wartung,
- kein geplanter Change.

Jetzt entsteht aus dem Event möglicherweise ein Incident.

Informationsfluss



Nicht jedes Event benötigt menschliches Eingreifen.

Arten von Monitoring

Typische Bereiche:

- Server
- Netzwerk
- Cloud
- Datenbanken
- Anwendungen
- Container
- virtuelle Maschinen
- Speicher
- Backup
- Sicherheit

- Zertifikate
- Hardware
- Umgebungsbedingungen

Je nach Service kommen unterschiedliche Überwachungen zum Einsatz.

Technisches Monitoring

Technisches Monitoring überwacht einzelne Komponenten.

Beispiele:

- CPU-Auslastung,
- RAM-Auslastung,
- Festplattenbelegung,
- Netzwerkverkehr,
- Temperatur,
- Lüfter,
- Spannungsversorgung,
- Hardwarefehler.

Es beantwortet die Frage:

> “ Funktioniert das System technisch?

Service Monitoring

Service Monitoring betrachtet den Service aus Benutzersicht.

Beispiele:

- Anmeldung möglich?
- Website erreichbar?
- E-Mail versendbar?
- Datenbankabfrage erfolgreich?
- VPN-Anmeldung funktioniert?
- API antwortet korrekt?

Ein Server kann technisch erreichbar sein,

während der eigentliche Service bereits gestört ist.

Aktives Monitoring

Beim aktiven Monitoring werden Systeme regelmäßig geprüft.

Beispiele:

- Ping,
- HTTP-Aufruf,
- Datenbankanmeldung,
- DNS-Abfrage,
- SMTP-Test,
- API-Aufruf.

Die Prüfungen erfolgen automatisch in festgelegten Intervallen.

Passives Monitoring

Beim passiven Monitoring senden Systeme selbst Informationen.

Beispiele:

- Syslog,
- Windows Event Log,
- SNMP-Traps,
- Sicherheitsmeldungen,
- Container-Logs,
- Anwendungsprotokolle.

Hier wartet das Monitoring auf eintreffende Ereignisse.

Synthetisches Monitoring

Synthetisches Monitoring simuliert typische Benutzeraktionen.

Beispiele:

- Benutzer anmelden,
- Bestellung durchführen,
- Suchfunktion testen,
- Datei hochladen,
- VPN verbinden.

Dadurch wird geprüft,

ob ein Service tatsächlich nutzbar ist.

Real User Monitoring (RUM)

Beim Real User Monitoring werden echte Benutzeraktionen ausgewertet.

Beispiele:

- Ladezeiten,
- Antwortzeiten,
- Fehlermeldungen,
- Browserdaten,
- Standortinformationen.

Dadurch erhält die IT Informationen über die tatsächliche Benutzererfahrung.

Überwachungsobjekte

Monitoring kann sich beziehen auf:

- Configuration Items,
- Services,
- Anwendungen,
- Netzwerke,
- Cloud-Ressourcen,
- Container,
- Datenbanken,
- APIs,
- Sicherheitskomponenten,
- Backup-Systeme.

Nicht jedes Objekt benötigt dieselbe Überwachung.

Grenzwerte (Thresholds)

Viele Überwachungen arbeiten mit Grenzwerten.

Beispiele:

Messwert	Grenzwert
CPU	über 90 %
RAM	über 95 %
Festplatte	über 85 % belegt
Zertifikat	läuft in 30 Tagen ab
Backup	fehlgeschlagen
Antwortzeit	über 2 Sekunden

Grenzwerte sollten regelmäßig überprüft werden.

Frühwarnungen

Nicht jeder Alarm muss sofort kritisch sein.

Beispiel:

Festplatte:

- 70 % → normal
- 85 % → Warnung
- 95 % → kritisch

Dadurch bleibt genügend Zeit,

Probleme zu beheben.

Alarmmüdigkeit (Alert Fatigue)

Zu viele unnötige Alarme führen dazu,

dass wichtige Warnungen übersehen werden.

Typische Ursachen:

- falsche Grenzwerte,
- doppelte Alarme,
- bekannte Ereignisse,
- ungefilterte Meldungen,
- schlecht abgestimmtes Monitoring.

Ein gutes Monitoring erzeugt möglichst wenige,

aber relevante Alarme.

Praxisbeispiel

Das Monitoring erkennt:

- Zertifikat läuft in 30 Tagen ab.

Es entsteht:

- Warnung,

- Ticket,
- Benachrichtigung an den Administrator.

Das Zertifikat wird rechtzeitig erneuert.

Ein späterer Incident wird vermieden.

Typische Fehler

Fehler 1

Monitoring existiert nur für Server,

nicht für Services.

Fehler 2

Grenzwerte sind ungeeignet.

Fehler 3

Zu viele unnötige Alarme.

Fehler 4

Alarme werden nicht bearbeitet.

Fehler 5

Backups werden nicht überwacht.

Fehler 6

Zertifikate werden nicht überwacht.

Fehler 7

Monitoring erkennt Fehler erst nach Benutzerbeschwerden.

Fehler 8

Monitoring wird nach Änderungen nicht angepasst.

Checkliste Monitoring

- [ ] kritische Services überwacht
- [ ] wichtige Configuration Items überwacht
- [ ] sinnvolle Grenzwerte definiert
- [ ] Alarme getestet
- [ ] Benachrichtigungen funktionieren
- [ ] Zertifikate überwacht
- [ ] Backups überwacht
- [ ] Dokumentation aktuell
- [ ] Verantwortlichkeiten festgelegt
- [ ] regelmäßige Überprüfung geplant

Bedeutung für Fachinformatiker für Systemintegration

Monitoring gehört zu den täglichen Aufgaben vieler Fachinformatiker.

Typische Tätigkeiten:

- Monitoring konfigurieren,
- Alarme auswerten,
- Grenzwerte anpassen,
- Logs analysieren,
- Events bewerten,
- Incidents erkennen,
- Automatisierungen entwickeln,
- Monitoring kontinuierlich verbessern.

Ein gut aufgebautes Monitoring verhindert viele Störungen, bevor Benutzer sie überhaupt bemerken.

Zusammenfassung

> Systeme und Services überwachen
>
> ↓

> Messwerte sammeln
>
> ↓
>
> Event erzeugen
>
> ↓
>
> Event bewerten
>
> ↓
>
> Alarm oder Automatisierung
>
> ↓
>
> Incident vermeiden oder bearbeiten
>
> ↓
>
> Service stabil halten

Merksätze

> “ Monitoring sammelt Informationen – Event Management bewertet sie.

> “ Nicht jedes Event ist ein Incident.

> “ Gutes Monitoring erkennt Probleme vor den Benutzern.

> “ Service Monitoring ist wichtiger als reine Serverüberwachung.

> “ Wenige aussagekräftige Alarme sind besser als viele unnötige Warnungen.

Verwandte Seiten

- 10.2 Event-Typen, Filter und Priorisierung
- 10.3 Alarme, Eskalationen und Automatisierung
- 10.4 Monitoring-Werkzeuge, Dashboards und Kennzahlen
- 10.5 Zusammenspiel mit Incident, Problem, Change und Service Configuration Management
- Incident Management
- Service Configuration Management
- Service Level Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Monitoring and Event Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Monitoring-Arten, Beispiele und Grenzwerte sind herstellerneutrale Praxisempfehlungen. ITIL schreibt keine konkreten Monitoring-Werkzeuge oder festen Schwellenwerte vor. Diese richten sich nach den überwachten Services, den Geschäftsanforderungen und der technischen Umgebung.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026

10.2 Event-Typen, Filter und Priorisierung

> **Kurz erklärt**
>
> Ein Monitoring-System erzeugt täglich oft tausende oder sogar Millionen von Events.
>
> Die Aufgabe des Event Managements besteht darin, diese Ereignisse zu bewerten, unwichtige Meldungen herauszufiltern und relevante Events an die richtigen Prozesse oder Personen weiterzuleiten.
>
> Nur so können Störungen früh erkannt werden, ohne Administratoren mit unnötigen Meldungen zu überlasten.

Warum Event Management notwendig ist

Ein modernes Rechenzentrum erzeugt ständig Ereignisse.

Beispiele:

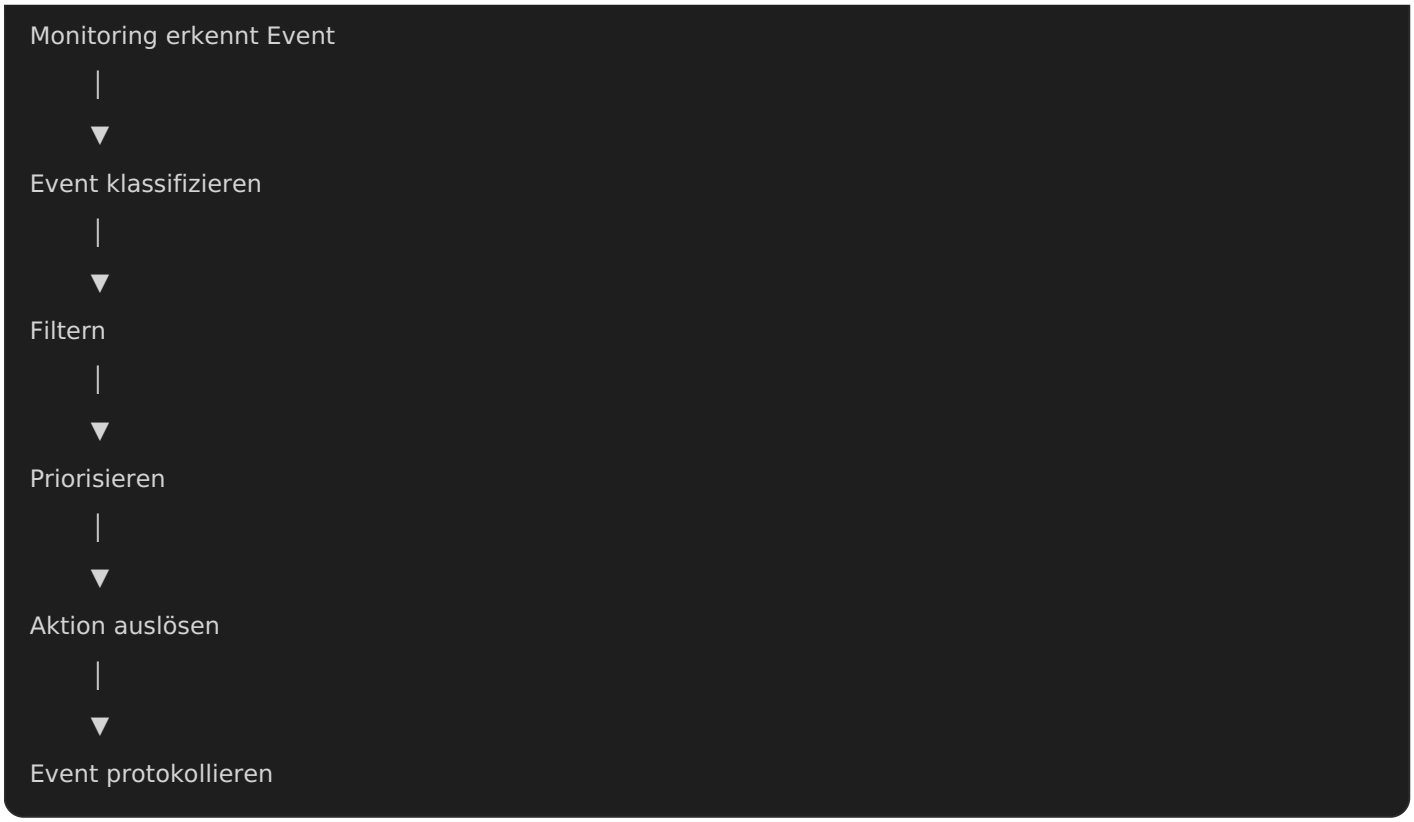
- Benutzer melden sich an.
- Dienste starten.
- Backups beginnen.
- Container werden erstellt.
- Zertifikate werden geprüft.
- CPU-Auslastung ändert sich.
- Netzwerkverkehr steigt.
- Datenbanken schreiben Logeinträge.

Ohne Bewertung wären diese Informationen kaum nutzbar.

Event Management entscheidet, welche Ereignisse wichtig sind.

Lebenszyklus eines Events

Ein typischer Ablauf:



Nicht jedes Event führt zu einer Aktion.

Kategorien von Events

ITIL unterscheidet grundsätzlich drei Event-Typen:

Event-Typ	Bedeutung
Informational	normale Information
Warning	mögliche zukünftige Störung
Exception	Fehler oder kritischer Zustand

Diese Einteilung hilft bei der automatischen Verarbeitung.

Informational Events

Informational Events dokumentieren normale Vorgänge.

Beispiele:

- Server gestartet
- Benutzer angemeldet
- Backup begonnen
- Dienst erfolgreich beendet
- Software installiert

- VM erstellt
- Zertifikat erfolgreich erneuert

Diese Events dienen hauptsächlich der Dokumentation.

Normalerweise ist keine Reaktion erforderlich.

Warning Events

Warning Events weisen auf mögliche zukünftige Probleme hin.

Beispiele:

- Festplatte zu 85 % belegt
- Zertifikat läuft in 30 Tagen ab
- CPU dauerhaft hoch ausgelastet
- Speicherverbrauch steigt
- Backup dauert ungewöhnlich lange
- Datenbank wächst stark

Noch besteht kein Incident.

Es sollte jedoch geprüft werden, ob Maßnahmen erforderlich sind.

Exception Events

Exception Events zeigen einen Fehler oder kritischen Zustand an.

Beispiele:

- Server nicht erreichbar
- Backup fehlgeschlagen
- RAID degradiert
- Zertifikat abgelaufen
- Datenbank gestoppt
- Webservice antwortet nicht
- Netzwerkschnittstelle ausgefallen
- Container abgestürzt

Diese Events führen häufig zu:

- Alarmen,
- Tickets,
- Automatisierungen,
- oder Incidents.

Warum nicht jedes Event gleich wichtig ist

Ein Drucker im Lager fällt aus.

Ein Domänencontroller fällt aus.

Beides sind Exception Events.

Die Auswirkungen unterscheiden sich jedoch erheblich.

Deshalb erfolgt zusätzlich eine Priorisierung.

Priorisierung von Events

Bei der Priorisierung werden unter anderem berücksichtigt:

- betroffener Service,
- Kritikalität,
- Anzahl betroffener Benutzer,
- geschäftliche Auswirkungen,
- Sicherheitsrelevanz,
- Wiederholungen,
- Tageszeit,
- vorhandene Redundanz.

Dadurch entstehen sinnvolle Reaktionen.

Beispiel

Zertifikat läuft in 30 Tagen ab.

→ Warning

Zertifikat heute abgelaufen.

→ Exception

Der Event-Typ ändert sich automatisch,

obwohl derselbe Sachverhalt überwacht wird.

Event-Filter

Nicht alle Events sollen angezeigt werden.

Filter unterdrücken beispielsweise:

- bekannte Wartungen,
- Testsysteme,
- doppelte Meldungen,
- bereits bestätigte Fehler,
- unwichtige Statusänderungen,
- Debug-Meldungen.

Dadurch sinkt die Anzahl unnötiger Alarme.

Warum Filter wichtig sind

Ohne Filter entstehen häufig:

- Alarmfluten,
- doppelte Tickets,
- unnötige Eskalationen,
- ignorierte Warnungen,
- überlastete Administratoren.

Ein gutes Monitoring meldet möglichst nur relevante Ereignisse.

Event-Korrelation

Mehrere Events können dieselbe Ursache besitzen.

Beispiel:

Ein Switch fällt aus.

Dadurch entstehen gleichzeitig:

- Server nicht erreichbar
- Drucker offline
- Access Points offline
- Telefonanlage nicht erreichbar
- NAS nicht erreichbar

Eine Event-Korrelation erkennt,

dass alle Meldungen wahrscheinlich dieselbe Ursache besitzen.

Dadurch entsteht nicht für jedes Event ein separates Incident-Ticket.

Deduplizierung

Viele Systeme melden denselben Fehler mehrfach.

Beispiel:

Alle 30 Sekunden:

> Datenbank nicht erreichbar

Die Deduplizierung fasst identische Events zusammen.

Dadurch entsteht nur ein Alarm.

Unterdrückung (Suppression)

Bei geplanten Wartungen können bestimmte Events unterdrückt werden.

Beispiel:

Während eines genehmigten Changes:

- Server wird neu gestartet.

Das Monitoring erkennt den Ausfall,

erzeugt jedoch keinen Incident,

da die Wartung bekannt ist.

Zeitliche Bewertung

Auch der Zeitpunkt spielt eine Rolle.

Beispiel:

CPU:

- 95 % für zwei Sekunden → normal
- 95 % für zwei Stunden → Warnung
- 100 % dauerhaft → Exception

Nicht jede kurzfristige Spitze stellt ein Problem dar.

Abhängigkeiten berücksichtigen

Ein Service kann aus vielen Komponenten bestehen.

Fällt der Router aus,

melden gleichzeitig:

- Webserver,
- Datenbank,
- Storage,
- Backup,
- Telefonie

Fehler.

Die eigentliche Ursache bleibt jedoch der Router.

Deshalb sollten Abhängigkeiten im Monitoring berücksichtigt werden.

Business Impact

Nicht jedes technische Problem besitzt dieselbe geschäftliche Bedeutung.

Beispiele:

Testsystem nicht erreichbar.

→ geringer Business Impact.

Produktionssystem ausgefallen.

→ hoher Business Impact.

Event Management sollte diese Unterschiede berücksichtigen.

Automatische Reaktionen

Je nach Event können unterschiedliche Aktionen erfolgen.

Beispiele:

Informational:

- protokollieren.

Warning:

- E-Mail,
- Dashboard,
- Ticket.

Exception:

- Alarm,
- Incident,
- SMS,
- Bereitschaft informieren,
- Automation starten.

Praxisbeispiel

Monitoring erkennt:

Festplatte:

84 %

→ Information.

85 %

→ Warning.

95 %

→ Exception.

98 %

→ automatischer Incident.

100 %

→ Notfallmaßnahmen.

Die Reaktion wird schrittweise intensiver.

Typische Fehler

Fehler 1

Alle Events besitzen dieselbe Priorität.

Fehler 2

Keine Event-Korrelation.

Fehler 3

Keine Filter vorhanden.

Fehler 4

Doppelte Events erzeugen mehrere Tickets.

Fehler 5

Warnungen werden ignoriert.

Fehler 6

Business Impact wird nicht berücksichtigt.

Fehler 7

Wartungen erzeugen unnötige Alarme.

Fehler 8

Grenzwerte werden nie überprüft.

Checkliste Event Management

- [ ] Event korrekt klassifiziert
- [ ] Priorität festgelegt
- [ ] Filter angewendet
- [ ] Event-Korrelation aktiv
- [ ] Deduplizierung vorhanden
- [ ] Wartungsfenster berücksichtigt
- [ ] Business Impact bewertet

- [ ] automatische Reaktion definiert
- [ ] Dokumentation aktuell
- [ ] Verantwortlichkeiten bekannt

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker konfigurieren häufig Monitoring- und Event-Management-Systeme.

Typische Aufgaben:

- Grenzwerte festlegen,
- Event-Regeln erstellen,
- Filter konfigurieren,
- Alarmfluten reduzieren,
- Event-Korrelation verbessern,
- Dashboards pflegen,
- Monitoring kontinuierlich optimieren.

Ein gutes Event Management reduziert unnötige Arbeit und sorgt dafür, dass kritische Ereignisse schnell erkannt werden.

Zusammenfassung

> Event entsteht
>
> ↓
>
> Event klassifizieren
>
> ↓
>
> Filter anwenden
>
> ↓
>
> Priorität bestimmen
>
> ↓
>
> Event korrelieren
>
> ↓

> Reaktion auslösen
>
> ↓
>
> Dokumentieren

Merksätze

> Nicht jedes Event ist wichtig.

> Informational Events dokumentieren normale Zustände.

> Warning Events weisen auf mögliche zukünftige Probleme hin.

> Exception Events erfordern häufig eine Reaktion.

> Gute Filter und Event-Korrelation verhindern Alarmfluten.

Verwandte Seiten

- 10.1 Ziele und Grundlagen des Monitoring and Event Management
- 10.3 Alarme, Eskalationen und Automatisierung
- 10.4 Monitoring-Werkzeuge, Dashboards und Kennzahlen
- 10.5 Zusammenspiel mit Incident, Problem, Change und Service Configuration Management
- Incident Management
- Problem Management
- Service Configuration Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Monitoring and Event Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Event-Kategorien (Informational, Warning und Exception) entsprechen der ITIL-Praxis. Die konkreten Prioritäten, Filterregeln, Schwellenwerte und Automatisierungen werden organisationsspezifisch definiert und hängen von den überwachten Services sowie den eingesetzten Monitoring-Werkzeugen ab.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026

10.3 Alarme, Eskalationen und Automatisierung

> **Kurz erklärt**
>
> Nicht jedes Event erfordert menschliches Eingreifen.
>
> Erst wenn ein Ereignis eine definierte Bedeutung besitzt, wird daraus ein Alarm.
>
> Je nach Schwere können anschließend automatische Aktionen, Eskalationen oder die Erstellung eines Incident-Tickets erfolgen.
>
> Ziel ist es, Probleme möglichst schnell und möglichst automatisch zu erkennen und zu behandeln.

Vom Event zum Alarm

Ein Monitoring-System erzeugt kontinuierlich Events.

Nur ein Teil dieser Events wird zu Alarmen.

Ein typischer Ablauf:

Die Bewertung erfolgt anhand definierter Regeln.

Was ist ein Alarm?

Ein Alarm ist eine Benachrichtigung über ein relevantes Ereignis.

Er informiert darüber, dass eine Reaktion erforderlich oder zumindest sinnvoll sein kann.

Ein Alarm bedeutet jedoch nicht automatisch, dass bereits ein Incident vorliegt.

Beispiele für Alarme

Typische Alarme:

- Festplatte fast voll
- Backup fehlgeschlagen
- RAID degradiert
- Server nicht erreichbar
- hohe CPU-Auslastung
- Zertifikat läuft bald ab
- Temperatur überschritten
- Netzwerkverbindung unterbrochen
- Container abgestürzt
- Datenbank antwortet nicht

Je nach Kritikalität unterscheiden sich die weiteren Maßnahmen.

Alarme priorisieren

Nicht jeder Alarm besitzt dieselbe Bedeutung.

Mögliche Kriterien:

- betroffener Service,
- Business Impact,
- Anzahl betroffener Benutzer,
- Kritikalität,
- Tageszeit,
- Sicherheitsrelevanz,
- vorhandene Redundanz,
- Wiederholungen.

Dadurch kann dieselbe technische Meldung unterschiedlich behandelt werden.

Alarmstufen

Viele Organisationen verwenden mehrere Alarmstufen.

Beispiel:

Stufe	Bedeutung
Information	nur protokollieren
Warnung	beobachten
Hoch	Administrator informieren
Kritisch	sofort reagieren
Notfall	Eskalation und Incident

Die genaue Anzahl der Stufen ist organisationsabhängig.

Alarmkanäle

Alarme können über verschiedene Wege zugestellt werden.

Beispiele:

- Dashboard
- E-Mail
- SMS
- Push-Nachricht
- Microsoft Teams
- Slack
- Telefonanruf
- Bereitschaftssystem
- Pager
- ITSM-System

Der Kanal richtet sich nach der Kritikalität.

Alarmrouting

Nicht jeder Alarm geht an dieselbe Person.

Beispiele:

Alarm	Empfänger
Netzwerk	Netzwerkteam

Alarm	Empfänger
Backup	Backup-Team
Datenbank	Datenbankadministration
Cloud	Cloud-Team
Active Directory	Windows-Team
Linux	Linux-Team
Container	Plattform-Team

Dadurch gelangen Informationen direkt an die zuständigen Personen.

Automatische Reaktionen

Viele Alarme können automatisiert verarbeitet werden.

Beispiele:

- Dienst neu starten
- Container neu starten
- VM verschieben
- Backup erneut starten
- Speicher bereinigen
- DNS umschalten
- Last verteilen
- Ticket erstellen
- Benutzer informieren

Nicht jede Situation erfordert sofort einen Administrator.

Selbstheilung (Self-Healing)

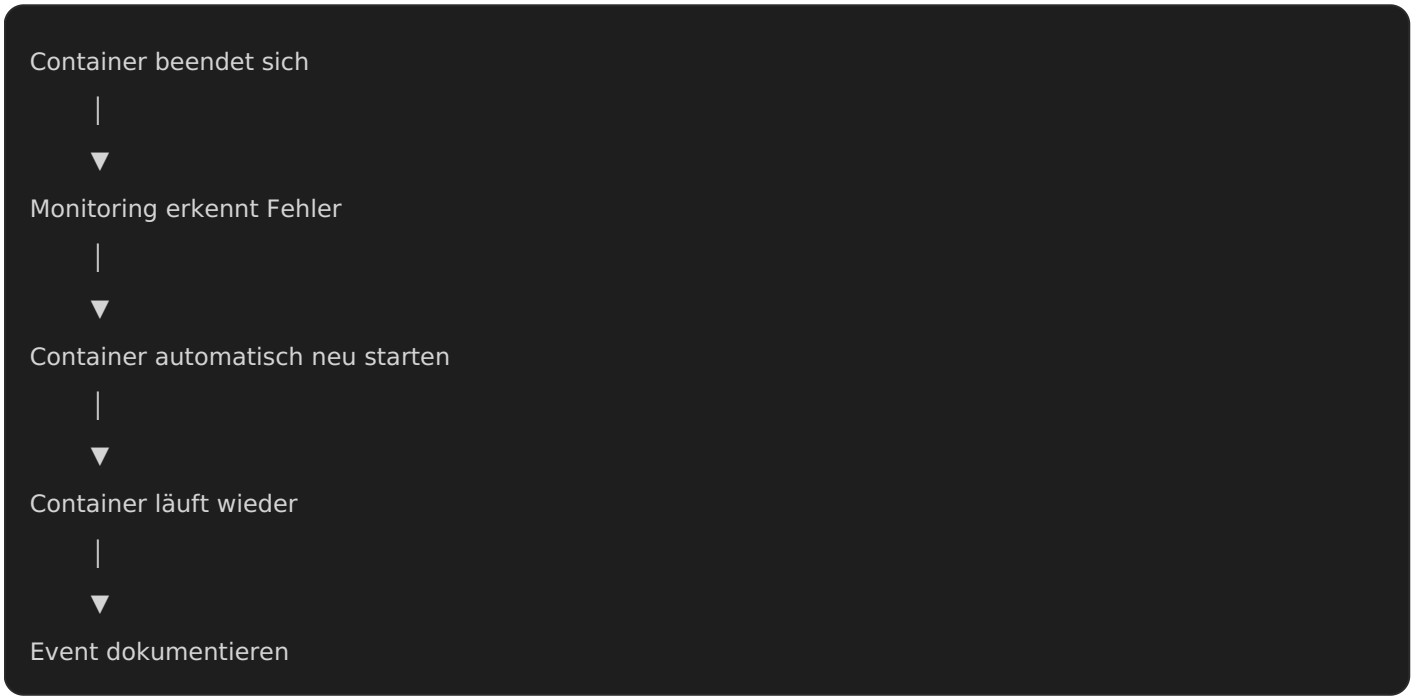
Self-Healing beschreibt automatische Korrekturmaßnahmen.

Beispiele:

- Dienst startet automatisch neu.
- Container wird neu erstellt.
- zweite Netzwerkverbindung wird aktiviert.
- Cluster übernimmt den Betrieb.
- Load Balancer entfernt fehlerhaften Server.
- Speicherplatz wird automatisch erweitert.

Dadurch entstehen viele Incidents gar nicht erst.

Beispiel



Der Benutzer bemerkt den Fehler möglicherweise überhaupt nicht.

Automatisierte Incident-Erstellung

Nicht jeder Alarm erzeugt automatisch einen Incident.

Sinnvoll ist dies beispielsweise bei:

- Serverausfall
- RAID-Fehler
- Datenbank nicht erreichbar
- Backup fehlgeschlagen
- Zertifikat abgelaufen
- Cloud-Service ausgefallen

Das ITSM-System erstellt automatisch ein Incident-Ticket.

Eskalationen

Kann ein Alarm nicht automatisch gelöst werden,

beginnt häufig eine Eskalation.

Beispiele:

- Bereitschaft informieren,

- Spezialisten hinzuziehen,
- Incident Manager benachrichtigen,
- Major Incident aktivieren.

Welche Eskalation erfolgt,

hängt von Schwere und Auswirkungen ab.

Zeitabhängige Eskalationen

Nicht jeder Alarm wird sofort eskaliert.

Beispiel:

CPU-Auslastung:

- über 90 % für 30 Sekunden → ignorieren
- über 90 % für 10 Minuten → Warnung
- über 95 % für 30 Minuten → Incident

Zeitfilter verhindern unnötige Reaktionen.

Mehrstufige Eskalationen

Ein möglicher Ablauf:

Nicht jeder Alarm erreicht automatisch die höchste Eskalationsstufe.

Alarmunterdrückung

Während geplanter Wartungen können Alarme unterdrückt werden.

Beispiel:

Ein Server wird bewusst neu gestartet.

Das Monitoring erkennt:

Server nicht erreichbar.

Da ein genehmigter Change aktiv ist,

wird kein Incident erzeugt.

Abhängigkeiten berücksichtigen

Ein ausgefallener Router verursacht häufig viele weitere Alarme.

Beispiele:

- Webserver offline
- NAS offline
- Drucker offline
- VoIP offline
- WLAN offline

Durch Abhängigkeitsregeln wird möglichst nur der eigentliche Fehler hervorgehoben.

Alarmkorrelation

Mehrere Alarme können zusammengehören.

Beispiel:

- Datenbank langsam
- Anwendung langsam
- Webserver meldet Fehler

Die eigentliche Ursache:

Storage reagiert verzögert.

Alarmkorrelation unterstützt Administratoren bei der Ursachenanalyse.

Alarmfluten vermeiden

Zu viele Warnungen führen häufig dazu,

dass wichtige Alarme übersehen werden.

Typische Maßnahmen:

- Filter,
- Korrelation,
- Deduplizierung,
- sinnvolle Grenzwerte,
- Wartungsfenster,
- Zeitfilter.

Dadurch steigt die Qualität der Alarmierung.

Dashboards

Dashboards stellen Alarme übersichtlich dar.

Typische Informationen:

- aktuelle Alarme,
- kritische Services,
- Verfügbarkeit,
- offene Incidents,
- Trends,
- Systemstatus,
- Karten,
- Serviceübersichten.

Sie ermöglichen einen schnellen Überblick.

Benachrichtigung außerhalb der Arbeitszeit

Kritische Systeme werden häufig rund um die Uhr überwacht.

Außerhalb der Geschäftszeiten können Alarme beispielsweise an:

- Rufbereitschaft,
- Bereitschaftsdienst,
- externen Dienstleister

gesendet werden.

Dadurch bleibt die Reaktionsfähigkeit erhalten.

Praxisbeispiel

Ein RAID-Verbund meldet:

Eine Festplatte ausgefallen.

Automatisch geschieht:

- Alarm erzeugen,
- Incident erstellen,
- Storage-Team informieren,
- Dashboard aktualisieren,
- Bereitschaft benachrichtigen,
- Dokumentation ergänzen.

Die eigentliche Reparatur erfolgt anschließend durch den zuständigen Administrator.

Typische Fehler

Fehler 1

Jeder Alarm erzeugt sofort einen Incident.

Fehler 2

Keine automatische Fehlerbehandlung.

Fehler 3

Keine Alarmkorrelation.

Fehler 4

Zu viele Benachrichtigungen.

Fehler 5

Keine Wartungsfenster berücksichtigt.

Fehler 6

Falsche Empfänger erhalten Alarme.

Fehler 7

Self-Healing wird nicht genutzt.

Fehler 8

Bereitschaft wird wegen unwichtiger Warnungen gestört.

Fehler 9

Alarmregeln werden nach Änderungen nicht angepasst.

Fehler 10

Fehlgeschlagene Automatisierungen bleiben unbemerkt.

Checkliste Alarmmanagement

- ☐ Event korrekt bewertet
- ☐ Alarmstufe festgelegt
- ☐ richtiger Empfänger definiert
- ☐ Eskalation vorhanden
- ☐ Wartungsfenster berücksichtigt
- ☐ Alarm dokumentiert
- ☐ Korrelation eingerichtet
- ☐ Deduplizierung aktiv
- ☐ Dashboard aktualisiert
- ☐ regelmäßige Überprüfung geplant

Checkliste Automatisierung

- ☐ Self-Healing möglich
- ☐ Workflow getestet
- ☐ Rollback vorhanden

- [ ] Fehler protokolliert
- [ ] Monitoring aktiv
- [ ] Incident-Erstellung geprüft
- [ ] Benachrichtigungen getestet
- [ ] Verantwortlichkeiten dokumentiert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker richten häufig Alarmierungen und automatische Reaktionen ein.

Typische Aufgaben:

- Alarmregeln definieren,
- Eskalationen konfigurieren,
- Dashboards pflegen,
- Automatisierungen entwickeln,
- Self-Healing testen,
- Benachrichtigungen überprüfen,
- Alarmfluten reduzieren,
- Monitoring kontinuierlich verbessern.

Ein gut abgestimmtes Alarmmanagement sorgt dafür, dass kritische Ereignisse schnell erkannt und möglichst automatisch behandelt werden.

Zusammenfassung

> Monitoring erkennt Event
>
> ↓
>
> Event bewerten
>
> ↓
>
> Alarm erzeugen
>
> ↓
>
> Priorisieren
>
> ↓

> Automatische Reaktion oder Eskalation
>
> ↓
>
> Incident (falls erforderlich)
>
> ↓
>
> Dokumentation

Merksätze

> Nicht jeder Alarm führt zu einem Incident.

> Gute Alarmregeln vermeiden unnötige Benachrichtigungen.

> Self-Healing kann viele Incidents vollständig verhindern.

> Alarmkorrelation unterstützt die Ursachenanalyse.

> Die richtige Information muss zur richtigen Zeit die richtige Person erreichen.

Verwandte Seiten

- 10.1 Ziele und Grundlagen des Monitoring and Event Management
- 10.2 Event-Typen, Filter und Priorisierung
- 10.4 Monitoring-Werkzeuge, Dashboards und Kennzahlen
- 10.5 Zusammenspiel mit Incident, Problem, Change und Service Configuration Management
- Incident Management
- Problem Management
- Service Configuration Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Monitoring and Event Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Alarmierungsstrategien, Eskalationsmodelle, Self-Healing-Beispiele und Automatisierungen sind herstellerneutrale Praxisempfehlungen. ITIL definiert die Grundprinzipien des Monitoring and Event Management, schreibt jedoch keine konkreten Alarmstufen, Eskalationswege oder Automatisierungsmechanismen vor. Diese werden organisationsabhängig anhand der Servicekritikalität und der eingesetzten Monitoring- und ITSM-Werkzeuge umgesetzt.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026

10.4 Monitoring-Werkzeuge, Dashboards und Kennzahlen

> **Kurz erklärt**
>
> Monitoring-Werkzeuge sammeln kontinuierlich Informationen über IT-Services, Anwendungen und Infrastruktur.
>
> Dashboards bereiten diese Informationen übersichtlich auf, während Kennzahlen (Metrics) helfen, Trends, Risiken und Verbesserungsmöglichkeiten zu erkennen.
>
> Ziel ist es, aus einer großen Menge technischer Daten verwertbare Informationen für den Betrieb und die Entscheidungsfindung zu gewinnen.

Warum Monitoring-Werkzeuge notwendig sind

Moderne IT-Umgebungen bestehen häufig aus:

- Servern,
- virtuellen Maschinen,
- Containern,
- Cloud-Diensten,
- Datenbanken,
- Netzwerkkomponenten,
- Firewalls,
- Anwendungen,
- APIs,
- Storage-Systemen,
- IoT-Geräten.

Eine manuelle Überwachung wäre praktisch unmöglich.

Monitoring-Werkzeuge sammeln und bewerten diese Informationen automatisch.

Aufgaben eines Monitoring-Werkzeugs

Ein Monitoring-System kann beispielsweise:

- Systeme überwachen,
- Services überwachen,
- Events sammeln,

- Alarme erzeugen,
- Kennzahlen berechnen,
- Trends erkennen,
- Dashboards bereitstellen,
- Berichte erzeugen,
- Benachrichtigungen versenden,
- Automatisierungen starten.

Das Werkzeug ersetzt dabei nicht die Analyse durch Fachpersonal.

Es liefert die notwendigen Informationen.

Was ist ein Dashboard?

Ein Dashboard fasst wichtige Informationen übersichtlich zusammen.

Es beantwortet Fragen wie:

- Welche Systeme sind gestört?
- Welche Services sind kritisch?
- Welche Alarme sind offen?
- Gibt es Kapazitätsprobleme?
- Welche Trends sind erkennbar?
- Welche Incidents bestehen aktuell?

Dadurch entsteht ein schneller Überblick über den aktuellen Zustand der IT.

Ziele eines Dashboards

Ein gutes Dashboard soll:

- übersichtlich sein,
- aktuelle Daten anzeigen,
- Prioritäten sichtbar machen,
- schnelle Entscheidungen unterstützen,
- Trends darstellen,
- unnötige Informationen vermeiden,
- verschiedene Zielgruppen unterstützen.

Nicht jede Person benötigt dieselben Informationen.

Verschiedene Dashboards

Je nach Rolle unterscheiden sich Dashboards erheblich.

Beispiele:

- Service Desk
- Netzwerkteam
- Serveradministration
- Datenbankadministration
- Cloud-Team
- Informationssicherheit
- Management
- Rufbereitschaft

Jede Zielgruppe benötigt andere Informationen.

Service-Desk-Dashboard

Typische Inhalte:

- offene Incidents,
- offene Service Requests,
- aktuelle Alarme,
- Major Incidents,
- Verfügbarkeit wichtiger Services,
- Ticketprioritäten,
- SLA-Verletzungen,
- Wartungsfenster.

Das Dashboard unterstützt die tägliche Bearbeitung.

Netzwerk-Dashboard

Mögliche Informationen:

- Bandbreitenauslastung,
- Switch-Status,
- Router,
- VPN,
- Firewalls,
- WLAN,
- Paketverluste,
- Latenzen,
- Verbindungsqualität.

Dadurch lassen sich Netzwerkprobleme schnell erkennen.

Server-Dashboard

Typische Kennzahlen:

- CPU,
- RAM,
- Festplatten,
- Prozesse,
- Dienste,
- Hardwarezustand,
- Temperaturen,
- Backupstatus,
- Virtualisierung.

Cloud-Dashboard

Beispiele:

- virtuelle Maschinen,
- Kubernetes,
- Container,
- Storage,
- Datenbanken,
- Kosten,
- Skalierung,
- Verfügbarkeit,
- Cloud-Regionen.

Cloud-Dashboards berücksichtigen häufig zusätzlich wirtschaftliche Kennzahlen.

Management-Dashboard

Das Management benötigt meist keine technischen Details.

Interessant sind beispielsweise:

- Serviceverfügbarkeit,
- SLA-Erfüllung,
- Major Incidents,
- Trends,
- Risiken,
- Benutzerzufriedenheit,
- Kosten,
- Verbesserungsmaßnahmen.

Dashboards sollten immer an die Zielgruppe angepasst werden.

Was sind Kennzahlen (Metrics)?

Kennzahlen beschreiben messbare Eigenschaften eines Systems oder Services.

Beispiele:

- CPU-Auslastung,
- Antwortzeit,
- Verfügbarkeit,
- Fehlerrate,
- Speicherauslastung,
- Netzwerkverkehr,
- Anzahl Incidents,
- Anzahl Alarme,
- Backup-Erfolg.

Kennzahlen bilden die Grundlage für Berichte und Entscheidungen.

Technische Kennzahlen

Typische technische Kennzahlen:

- CPU-Auslastung
- Arbeitsspeicher
- Festplattenbelegung
- Netzwerkdurchsatz
- Temperatur
- Antwortzeit
- IOPS
- Datenbankverbindungen
- Paketverlust
- DNS-Antwortzeit

Sie beschreiben den technischen Zustand.

Servicebezogene Kennzahlen

Beispiele:

- Serviceverfügbarkeit,
- Anmeldezeit,
- Transaktionsdauer,
- erfolgreiche Logins,

- API-Verfügbarkeit,
- Anzahl erfolgreicher Bestellungen,
- Wiederherstellungszeit,
- SLA-Erfüllung.

Sie beschreiben die Sicht des Benutzers.

Geschäftsbezogene Kennzahlen

Beispiele:

- Bestellungen pro Stunde,
- Zahlungsvorgänge,
- Produktionsmenge,
- Kundenanfragen,
- Umsatz,
- Bearbeitungszeit,
- Benutzerzufriedenheit.

Diese Kennzahlen verbinden IT und Geschäftsprozesse.

Wichtige Monitoring-Kennzahlen

Beispiele:

Kennzahl	Aussage
Verfügbarkeit	läuft der Service?
Antwortzeit	wie schnell reagiert der Service?
CPU	Prozessorauslastung
RAM	Speicherverbrauch
Festplatte	freier Speicher
Netzwerk	Datenverkehr
Fehlerrate	Anzahl Fehler
Backup	erfolgreich oder fehlgeschlagen
Zertifikate	verbleibende Laufzeit
Alarme	Anzahl kritischer Ereignisse

Nicht jede Kennzahl ist für jeden Service relevant.

KPIs (Key Performance Indicators)

KPIs sind besonders wichtige Kennzahlen.

Beispiele:

- Serviceverfügbarkeit 99,9 %
- Wiederherstellungszeit
- SLA-Erfüllung
- Incident-Anzahl
- Benutzerzufriedenheit
- erfolgreiche Deployments
- Fehlerrate
- Mean Time to Detect (MTTD)
- Mean Time to Restore (MTTR)

KPIs dienen häufig als Grundlage für Service Reviews.

MTTD

Mean Time to Detect beschreibt,

wie schnell eine Störung erkannt wird.

Je kleiner der Wert,

desto früher erkennt das Monitoring Probleme.

MTTR

Mean Time to Restore beschreibt,

wie lange die Wiederherstellung eines Services dauert.

Monitoring kann MTTR verkürzen,

weil Fehler früher erkannt werden.

Trendanalysen

Einzelne Messwerte sind oft wenig aussagekräftig.

Trends zeigen Entwicklungen.

Beispiele:

- Speicherverbrauch steigt seit Wochen.

- CPU-Auslastung nimmt kontinuierlich zu.
- Datenbank wächst täglich.
- Anzahl Incidents sinkt.

Trends unterstützen Kapazitätsplanung und Verbesserungen.

Historische Daten

Monitoring speichert häufig Messwerte über längere Zeit.

Dadurch lassen sich:

- Entwicklungen,
- Vergleiche,
- Kapazitätsplanungen,
- Ursachenanalysen

durchführen.

Historische Daten sind für Continual Improvement besonders wertvoll.

Visualisierung

Kennzahlen lassen sich unterschiedlich darstellen.

Beispiele:

- Tabellen,
- Balkendiagramme,
- Liniendiagramme,
- Kreisdiagramme,
- Heatmaps,
- Ampelsysteme,
- Service Maps.

Die Darstellung sollte zur jeweiligen Fragestellung passen.

Ampelsysteme

Viele Dashboards verwenden Farben.

Beispiel:

□□ Normal

□□ Warnung

□□ Kritisch

Farben ermöglichen eine schnelle Orientierung.

Sie sollten jedoch nicht die einzige Informationsquelle sein.

Service Maps

Service Maps zeigen,

welche Systeme zu einem Service gehören.

Beispiel:

```
Online-Shop
    │
    ├── Webserver
    ├── Datenbank
    ├── Storage
    ├── DNS
    ├── Load Balancer
    └── Payment API
```

Dadurch werden Abhängigkeiten sichtbar.

Kapazitätsplanung

Monitoring unterstützt Capacity Management.

Beispiele:

- Speicher wächst kontinuierlich.
- CPU dauerhaft hoch.
- Netzwerk regelmäßig ausgelastet.
- Datenbank benötigt mehr Ressourcen.

Kapazitätsprobleme können früh erkannt werden.

Monitoring und Reports

Aus Kennzahlen entstehen Reports.

Typische Inhalte:

- Verfügbarkeit,
- SLA-Erfüllung,
- Trends,
- Major Incidents,
- Alarme,
- Wartungsfenster,
- Verbesserungen.

Reports richten sich häufig an:

- Service Owner,
- Management,
- Kunden,
- Administratoren.

Praxisbeispiel

Das Dashboard zeigt:

- CPU 28 %
- RAM 61 %
- Storage 78 %
- Zertifikat 25 Tage
- Backup erfolgreich
- keine offenen kritischen Alarme

Administratoren erkennen sofort,

dass aktuell kein unmittelbarer Handlungsbedarf besteht.

Typische Fehler

Fehler 1

Zu viele Kennzahlen.

Fehler 2

Falsche Zielgruppe.

Fehler 3

Veraltete Dashboards.

Fehler 4

Nur technische Kennzahlen.

Fehler 5

Keine Trendanalyse.

Fehler 6

KPIs sind unklar definiert.

Fehler 7

Service Maps fehlen.

Fehler 8

Historische Daten werden nicht ausgewertet.

Fehler 9

Dashboards enthalten zu viele Details.

Fehler 10

Kennzahlen werden gemessen,

aber nicht genutzt.

Checkliste Monitoring-Dashboard

- [ ] aktuelle Daten
- [ ] passende Zielgruppe
- [ ] relevante KPIs
- [ ] kritische Services sichtbar
- [ ] Trends vorhanden

- [ ] Alarme sichtbar
- [ ] Service Maps integriert
- [ ] übersichtliche Darstellung
- [ ] regelmäßige Aktualisierung
- [ ] verständliche Visualisierung

Checkliste Kennzahlen

- [ ] eindeutig definiert
- [ ] messbar
- [ ] aktuell
- [ ] nachvollziehbar
- [ ] historisch auswertbar
- [ ] für Entscheidungen geeignet
- [ ] Servicebezug vorhanden
- [ ] regelmäßig überprüft

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker arbeiten regelmäßig mit Monitoring-Dashboards und Kennzahlen.

Typische Aufgaben:

- Dashboards erstellen,
- Kennzahlen definieren,
- Trends analysieren,
- Alarme auswerten,
- Reports erstellen,
- Service Maps pflegen,
- Monitoring verbessern,
- Kapazitäten planen.

Die Fähigkeit, Messwerte richtig zu interpretieren, ist im IT-Betrieb oft wichtiger als das reine Sammeln der Daten.

Zusammenfassung

> Monitoring sammelt Messwerte

↓

Kennzahlen berechnen

↓

Dashboard darstellen

↓

Trends erkennen

↓

Reports erstellen

↓

Entscheidungen treffen

↓

Services verbessern

Merksätze

> Ein Dashboard soll Entscheidungen unterstützen – nicht möglichst viele Daten anzeigen.

> KPIs konzentrieren sich auf die wichtigsten Kennzahlen.

> Trends sind häufig wertvoller als einzelne Messwerte.

> Servicebezogene Kennzahlen sind für Benutzer wichtiger als reine Hardwarewerte.

>

> Historische Daten ermöglichen Ursachenanalysen und Kapazitätsplanung.

Verwandte Seiten

- 10.1 Ziele und Grundlagen des Monitoring and Event Management
- 10.2 Event-Typen, Filter und Priorisierung
- 10.3 Alarme, Eskalationen und Automatisierung
- 10.5 Zusammenspiel mit Incident, Problem, Change und Service Configuration Management
- Service Level Management
- Continual Improvement
- Capacity and Performance Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Monitoring and Event Management
- PeopleCert – ITIL Practice Guide: Measurement and Reporting
- PeopleCert – ITIL Practice Guide: Capacity and Performance Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Dashboards, Kennzahlen, KPIs und Visualisierungen sind herstellerneutrale Praxisempfehlungen. ITIL definiert Grundprinzipien für Monitoring und Reporting, schreibt jedoch keine konkreten Werkzeuge, Dashboards oder Kennzahlen vor. Die Auswahl richtet sich nach den überwachten Services, den Geschäftsanforderungen und den Informationsbedürfnissen der jeweiligen Zielgruppen.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026

10.5 Zusammenspiel mit Incident, Problem, Change und Service Configuration Management

> **Kurz erklärt**
>
> Monitoring and Event Management arbeitet eng mit anderen ITIL Practices zusammen.
>
> Monitoring erkennt Ereignisse, bewertet deren Bedeutung und stellt Informationen für andere Prozesse bereit.
>
> Erst durch das Zusammenspiel mit Incident Management, Problem Management, Change Enablement und Service Configuration Management entsteht ein vollständiger und effizienter IT-Servicebetrieb.

Warum das Zusammenspiel wichtig ist

Monitoring allein behebt keine Fehler.

Es beantwortet zunächst nur die Frage:

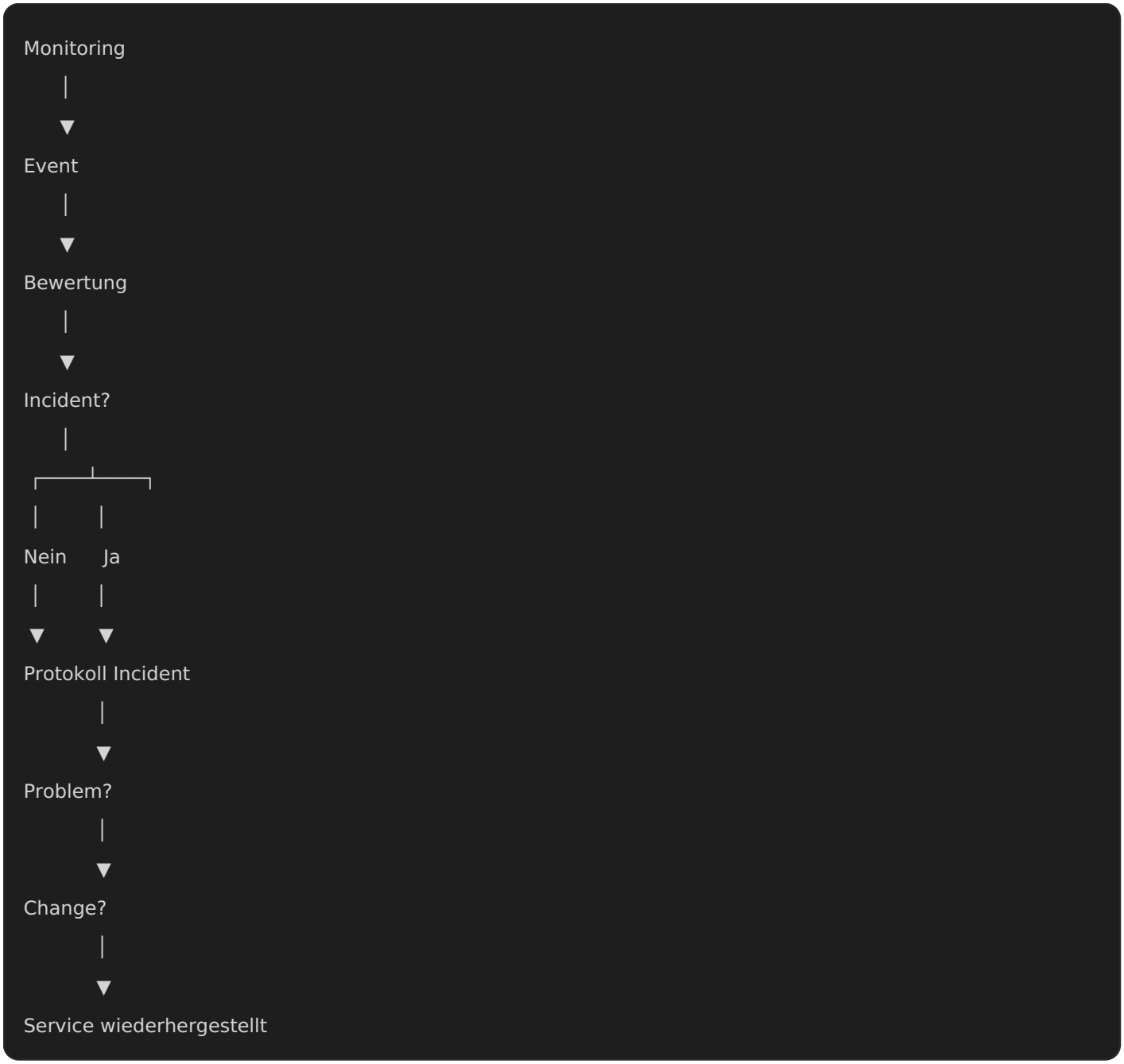
> **"Was ist passiert?"**

Andere ITIL Practices beantworten anschließend beispielsweise:

- Muss eine Störung behoben werden?
- Ist die Ursache bekannt?
- Muss eine Änderung durchgeführt werden?
- Welche Systeme sind betroffen?
- Welche Services sind beeinträchtigt?

Erst gemeinsam entsteht ein vollständiger Ablauf.

Zusammenspiel der Practices



Nicht jedes Event durchläuft alle Schritte.

Monitoring und Incident Management

Diese beiden Practices arbeiten besonders eng zusammen.

Monitoring erkennt Ereignisse.

Incident Management stellt den normalen Servicebetrieb wieder her.

Beispiel:

Monitoring erkennt:

> Webserver antwortet nicht.

Anschließend:

- Incident erstellen,
- Service Desk informieren,
- Administrator alarmieren,
- Service wiederherstellen.

Monitoring erkennt den Fehler.

Incident Management behebt ihn.

Monitoring erkennt häufig Incidents vor dem Benutzer

Ein großer Vorteil moderner Monitoring-Systeme:

Probleme werden oft erkannt,

bevor Benutzer sie melden.

Beispiele:

- Speicher fast voll
- Backup fehlgeschlagen
- Zertifikat läuft bald ab
- RAID degradiert
- Datenbank antwortet langsam

Viele Incidents können dadurch vollständig vermieden werden.

Monitoring und Problem Management

Ein einzelner Alarm ist häufig kein Problem.

Wiederholen sich jedoch dieselben Events regelmäßig,

kann daraus ein Problem Record entstehen.

Beispiel:

Jeden Dienstag:

- Datenbank langsam.

Incident wird jedes Mal behoben.

Monitoring zeigt jedoch,

dass dieselbe Ursache regelmäßig auftritt.

Jetzt beginnt Problem Management mit der Ursachenanalyse.

Trendanalysen unterstützen Problem Management

Monitoring speichert historische Daten.

Dadurch lassen sich erkennen:

- wiederkehrende Fehler,
- steigende Auslastung,
- häufige Neustarts,
- zunehmende Antwortzeiten,
- regelmäßig fehlschlagende Backups.

Diese Informationen helfen bei der Suche nach der eigentlichen Ursache.

Monitoring und Change Enablement

Viele Changes beeinflussen Monitoring.

Beispiele:

- neuer Server,
- zusätzliche VM,
- neue Firewall,
- neue Anwendung,
- Cloud-Service,
- Netzwerksegment.

Nach einem Change muss häufig auch das Monitoring angepasst werden.

Monitoring vor einem Change

Vor Änderungen liefern Monitoring-Daten wichtige Informationen.

Beispiele:

- aktuelle Auslastung,
- Antwortzeiten,

- Verfügbarkeit,
- Fehlerquote,
- Baseline.

Dadurch kann später geprüft werden,

ob der Change erfolgreich war.

Monitoring nach einem Change

Nach der Umsetzung wird kontrolliert:

- Service erreichbar?
- Fehler aufgetreten?
- Antwortzeiten verändert?
- CPU erhöht?
- neue Alarme?

Monitoring bestätigt,

ob die Änderung erfolgreich war.

Geplante Wartungen

Während eines genehmigten Changes sollten Monitoring-Systeme dies berücksichtigen.

Beispiel:

Server wird bewusst neu gestartet.

Ohne Wartungsfenster:

- Alarm,
- Incident,
- Eskalation.

Mit Wartungsfenster:

- Event wird dokumentiert,
- keine unnötigen Alarme entstehen.

Monitoring und Service Configuration Management

Monitoring überwacht Configuration Items.

Configuration Management beschreibt diese.

Beispiele:

- Server,
- Switch,
- Router,
- Datenbank,
- Container,
- VM,
- Storage,
- Firewall.

Monitoring liefert aktuelle Zustandsinformationen.

Configuration Management liefert Struktur und Beziehungen.

Warum Configuration Items wichtig sind

Ohne Configuration Management weiß Monitoring häufig nicht,

welche Bedeutung ein System besitzt.

Beispiel:

Server 01 fällt aus.

Monitoring erkennt den Ausfall.

Erst Configuration Management zeigt:

Dieser Server gehört zum:

- ERP-System,
- Produktionssteuerung,
- Rechnungswesen.

Dadurch lässt sich die Auswirkung deutlich besser bewerten.

Service Maps

Service Maps verbinden Monitoring und Configuration Management.

Beispiel:

```
Online-Shop
    │
    ├── Webserver
    ├── Load Balancer
    ├── Datenbank
    ├── Storage
    ├── DNS
    └── Payment API
```

Fällt eine Komponente aus,

werden die betroffenen Services sofort sichtbar.

Abhängigkeiten erkennen

Configuration Management zeigt,

welche Systeme voneinander abhängig sind.

Dadurch kann Monitoring:

- Alarme korrelieren,
- Prioritäten anpassen,
- Root Causes schneller erkennen.

Monitoring und Availability Management

Monitoring liefert die Grundlage zur Messung der Verfügbarkeit.

Beispiele:

- Server erreichbar?
- Anwendung verfügbar?
- API antwortet?
- Website erreichbar?

Availability Management nutzt diese Daten,

um Verfügbarkeitsziele zu bewerten.

Monitoring und Capacity Management

Monitoring misst kontinuierlich:

- CPU,
- RAM,
- Speicher,
- Netzwerk,
- Datenbankgröße.

Capacity Management nutzt diese Informationen,

um zukünftige Ressourcen zu planen.

Monitoring und Information Security Management

Monitoring erkennt auch sicherheitsrelevante Ereignisse.

Beispiele:

- ungewöhnliche Logins,
- fehlgeschlagene Anmeldungen,
- Malware,
- Firewall-Ereignisse,
- verdächtiger Netzwerkverkehr,
- Privilegienänderungen.

Diese Events können an Security-Prozesse weitergeleitet werden.

Monitoring und Continual Improvement

Monitoring liefert kontinuierlich Kennzahlen.

Dadurch lassen sich Verbesserungen erkennen.

Beispiele:

- MTTR sinkt,
- weniger Incidents,
- höhere Verfügbarkeit,
- kürzere Antwortzeiten,
- bessere Automatisierung.

Continual Improvement nutzt diese Daten,

um Prozesse weiterzuentwickeln.

Monitoring und Service Level Management

Viele SLA-Kennzahlen basieren direkt auf Monitoring-Daten.

Beispiele:

- Verfügbarkeit,
- Antwortzeit,
- Ausfallzeit,
- Servicequalität.

Ohne zuverlässiges Monitoring können SLA-Ziele kaum objektiv gemessen werden.

Praxisbeispiel

Das Monitoring erkennt:

Die Antwortzeit einer Datenbank steigt seit mehreren Wochen kontinuierlich an.

Zunächst entstehen einzelne Warning Events.

Später treten wiederholt Incidents auf.

Problem Management untersucht die Ursache und stellt fest,

dass der verfügbare Speicher nicht mehr ausreicht.

Ein genehmigter Change erweitert den Speicher.

Nach der Umsetzung zeigt das Monitoring wieder normale Antwortzeiten.

Die gewonnenen Erkenntnisse fließen anschließend in Continual Improvement ein.

Typische Fehler

Fehler 1

Monitoring arbeitet unabhängig von Incident Management.

Fehler 2

Configuration Items sind nicht aktuell.

Fehler 3

Changes berücksichtigen das Monitoring nicht.

Fehler 4

Monitoring-Daten werden nicht ausgewertet.

Fehler 5

Wiederkehrende Events führen nicht zu Problem Records.

Fehler 6

Service Maps fehlen.

Fehler 7

Abhängigkeiten werden nicht berücksichtigt.

Fehler 8

Monitoring misst keine SLA-Kennzahlen.

Fehler 9

Kapazitätsdaten werden nicht ausgewertet.

Fehler 10

Verbesserungen basieren nicht auf Messdaten.

Checkliste Zusammenspiel

- [ ] Monitoring aktiv
- [ ] Event korrekt bewertet
- [ ] Incident bei Bedarf erstellt
- [ ] Problem Record bei wiederkehrenden Fehlern geprüft
- [ ] Change berücksichtigt
- [ ] Configuration Items aktuell
- [ ] Service Maps gepflegt
- [ ] SLA-Kennzahlen verfügbar
- [ ] Sicherheitsereignisse erkannt

- [ ] Verbesserungspotenziale dokumentiert

Checkliste Monitoring-Prozess

- [ ] Events gesammelt
- [ ] Filter eingerichtet
- [ ] Alarmregeln geprüft
- [ ] Eskalationen definiert
- [ ] Dashboards aktuell
- [ ] KPIs überwacht
- [ ] Trends ausgewertet
- [ ] Reports erstellt
- [ ] Verantwortlichkeiten bekannt
- [ ] regelmäßige Optimierung geplant

Bedeutung für Fachinformatiker für Systemintegration

Monitoring gehört zu den wichtigsten Werkzeugen im Arbeitsalltag eines Fachinformatikers.

Typische Aufgaben:

- Monitoring konfigurieren,
- Alarme analysieren,
- Incidents früh erkennen,
- Monitoring nach Changes anpassen,
- Configuration-Daten pflegen,
- Service Maps aktualisieren,
- Trends auswerten,
- Automatisierungen verbessern.

Ein gut integriertes Monitoring unterstützt nahezu alle Bereiche des IT-Service-Managements.

Zusammenfassung

> Monitoring erkennt Events
>
> ↓
>
> Event bewerten

↓

Incident auslösen (falls erforderlich)

↓

Problem analysieren

↓

Change umsetzen

↓

Monitoring überprüft Ergebnis

↓

Erkenntnisse für Continual Improvement nutzen

Merksätze

> Monitoring erkennt Ereignisse – andere Practices reagieren darauf.

> Monitoring liefert die Grundlage für Incident, Problem und Change Management.

> Configuration Management erklärt die Beziehungen zwischen den überwachten Systemen.

> Gute Service Maps beschleunigen die Ursachenanalyse.

> Monitoring-Daten bilden die Grundlage für kontinuierliche Verbesserungen.

Verwandte Seiten

- 10.1 Ziele und Grundlagen des Monitoring and Event Management
- 10.2 Event-Typen, Filter und Priorisierung
- 10.3 Alarme, Eskalationen und Automatisierung
- 10.4 Monitoring-Werkzeuge, Dashboards und Kennzahlen
- Incident Management
- Problem Management
- Change Enablement
- Service Configuration Management
- Service Level Management
- Availability Management
- Capacity and Performance Management
- Information Security Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Monitoring and Event Management
- PeopleCert – ITIL Practice Guide: Incident Management
- PeopleCert – ITIL Practice Guide: Problem Management
- PeopleCert – ITIL Practice Guide: Change Enablement
- PeopleCert – ITIL Practice Guide: Service Configuration Management
- PeopleCert – ITIL Practice Guide: Availability Management
- PeopleCert – ITIL Practice Guide: Capacity and Performance Management
- PeopleCert – ITIL Practice Guide: Service Level Management
- PeopleCert – ITIL Practice Guide: Continual Improvement
- ITIL Foundation – Version 5

Einordnung

Die dargestellten Zusammenhänge orientieren sich an den offiziellen ITIL-Practices. Monitoring and Event Management stellt Informationen für zahlreiche weitere Practices bereit, übernimmt jedoch selbst weder die Fehlerbehebung noch die Ursachenanalyse oder die Planung von Änderungen. Die konkrete technische Umsetzung hängt von den eingesetzten Monitoring-, ITSM- und CMDB-Systemen der jeweiligen Organisation ab.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026