6.4 Discovery, Pflege und Datenqualität

> **Kurz erklärt**
>
> Discovery, Pflege und Datenqualität sorgen dafür, dass Configuration-Daten nicht nur einmal erfasst, sondern dauerhaft aktuell, korrekt und nutzbar bleiben.
>
> **Discovery** hilft, technische Informationen automatisch zu erkennen.
>
> **Pflege** sorgt dafür, dass fachliche, organisatorische und servicebezogene Informationen ergänzt und aktuell gehalten werden.
>
> **Datenqualität** entscheidet, ob Configuration-Daten im Betrieb wirklich verlässlich sind.

Warum dieses Thema wichtig ist

Eine CMDB oder ein Configuration Management System ist nur dann hilfreich, wenn die Daten stimmen.

Veraltete oder falsche Configuration-Daten führen zu Problemen:

- Incidents werden falsch priorisiert,
- Changes werden mit falscher Auswirkungsbewertung geplant,
- Problems werden nicht richtig eingegrenzt,
- Owner sind nicht erreichbar,
- Serviceabhängigkeiten fehlen,
- Sicherheitslücken werden übersehen,
- Zertifikate laufen ab,
- Cloud-Ressourcen bleiben unbekannt,
- und Dokumentation widerspricht der Realität.

Datenqualität ist deshalb kein Nebenthema.

Sie ist die Grundlage dafür, dass Service Configuration Management im Alltag funktioniert.

Discovery

Discovery bedeutet, technische Informationen automatisch oder halbautomatisch zu erkennen.

Mögliche Discovery-Ergebnisse:

- Server,
- Clients,
- Netzwerkgeräte,
- virtuelle Maschinen,
- Container-Plattformen,
- Cloud-Ressourcen,
- installierte Software,
- Betriebssystemversionen,
- IP-Adressen,
- offene Dienste,
- Zertifikate,
- Datenbankinstanzen,
- Schnittstellen,
- Hardwareinformationen,
- Speicher,
- Netzwerkverbindungen,
- und technische Abhängigkeiten.

Discovery kann Daten schneller und regelmäßiger erfassen als rein manuelle Pflege.

Was Discovery leisten kann

Discovery kann helfen bei:

- Erstbefüllung einer CMDB,
- Erkennung unbekannter Systeme,
- Aktualisierung technischer Daten,
- Abgleich von Ist-Zustand und Soll-Zustand,
- Erkennung veralteter Versionen,
- Sicherheitsbewertung,
- Audit-Vorbereitung,
- Cloud-Transparenz,
- Netzwerkübersicht,
- und Datenqualitätsprüfung.

Beispiel:

Ein Discovery-Werkzeug erkennt, dass ein Server noch eine alte Softwareversion verwendet.

Diese Information kann für Security, Change Enablement und Problem Management wichtig sein.

Was Discovery nicht automatisch leisten kann

Discovery erkennt Technik.

Discovery versteht aber nicht automatisch den fachlichen Zusammenhang.

Ein Discovery-Werkzeug erkennt vielleicht:

- ein Server existiert,
- ein Dienst läuft,
- eine IP-Adresse ist erreichbar,
- ein Zertifikat ist installiert,
- eine Anwendung kommuniziert mit einer Datenbank.

Es weiß aber nicht automatisch:

- welcher Geschäftsprozess betroffen ist,
- welcher Service Owner verantwortlich ist,
- welche Kritikalität der Service besitzt,
- ob eine Abweichung genehmigt ist,
- ob ein System bald abgelöst wird,
- welcher Lieferant verantwortlich ist,
- oder welcher Workaround im Störungsfall gilt.

Deshalb muss Discovery durch fachliche Pflege ergänzt werden.

Discovery und manuelle Pflege ergänzen sich

Discovery	Manuelle oder organisatorische Pflege
erkennt technische Objekte	ergänzt fachliche Bedeutung
erkennt Versionen und Eigenschaften	ergänzt Owner und Verantwortlichkeiten
erkennt Verbindungen	bewertet Serviceabhängigkeiten
erkennt neue Ressourcen	prüft, ob sie relevant sind
erkennt Abweichungen	klärt, ob sie erlaubt sind
liefert Ist-Daten	ergänzt Soll-Zustand und Kontext

Gute Configuration-Daten entstehen durch Kombination aus Automatisierung, Prozessen und Verantwortung.

Typische Discovery-Quellen

Mögliche Quellen für technische Erkennung:

- Netzwerk-Scanner,
- Monitoring-System,
- Endpoint-Management,
- Softwareverteilung,

- Virtualisierungsplattform,
- Cloud-Plattform,
- Container-Plattform,
- Verzeichnisdienst,
- Zertifikatsverwaltung,
- Backup-System,
- Datenbankverwaltung,
- Sicherheitswerkzeuge,
- ITSM-System,
- Asset-Management-System.

Keine Quelle ist vollständig.

Mehrere Quellen müssen oft abgeglichen werden.

Discovery in Cloud-Umgebungen

Cloud-Umgebungen verändern sich häufig schneller als klassische Rechenzentrumsumgebungen.

Typische Herausforderungen:

- Ressourcen werden automatisch erstellt,
- Ressourcen werden schnell gelöscht,
- Tags fehlen oder sind uneinheitlich,
- Owner sind unklar,
- Kostenstellen fehlen,
- Testressourcen bleiben aktiv,
- Dienste werden extern bereitgestellt,
- Abhängigkeiten sind dynamisch.

Wichtige Maßnahmen:

- verbindliche Tags definieren,
- Owner erfassen,
- Umgebung kennzeichnen,
- automatische Exporte nutzen,
- Cloud-Ressourcen regelmäßig prüfen,
- nicht genutzte Ressourcen bereinigen,
- Sicherheitsklassifizierung ergänzen.

Discovery in Container-Umgebungen

In Container-Umgebungen entstehen und verschwinden technische Instanzen oft sehr schnell.

Nicht jeder kurzlebige Container sollte dauerhaft als einzelnes CI gepflegt werden.

Sinnvoller können sein:

- Service,
- Anwendung,
- Namespace,
- Cluster,
- Deployment,
- Image-Version,
- Datenbank,
- Volume,
- Ingress,
- Zertifikat,
- Secret,
- Monitoring,
- Backupjob.

Wichtig ist, servicebezogene Informationen zu erfassen, nicht jede temporäre Laufzeitinstanz dauerhaft zu dokumentieren.

Discovery und Sicherheitsbewertung

Discovery ist für Sicherheit besonders wichtig.

Beispiele:

- unbekannte Systeme erkennen,
- veraltete Software finden,
- offene Dienste prüfen,
- externe Erreichbarkeit feststellen,
- ablaufende Zertifikate identifizieren,
- nicht genehmigte Cloud-Ressourcen erkennen,
- nicht inventarisierte Geräte finden,
- Systeme mit kritischen Schwachstellen priorisieren.

Aber auch hier gilt:

Discovery zeigt technische Tatsachen.

Die Risikobewertung benötigt zusätzlich Kritikalität, Datenklasse, Servicebezug und Owner.

Discovery-Ergebnisse prüfen

Discovery-Daten sollten nicht ungeprüft übernommen werden.

Zu prüfen ist:

- Ist das erkannte Objekt wirklich relevant?
- Ist es bereits als CI vorhanden?
- Ist der Name eindeutig?
- Ist die Umgebung korrekt?
- Ist der Status korrekt?
- Gibt es Dubletten?
- Ist der Owner bekannt?
- Gibt es einen Servicebezug?
- Handelt es sich um ein temporäres Objekt?
- Muss das Objekt dauerhaft gepflegt werden?

Automatisierung ohne Prüfung kann die CMDB mit unbrauchbaren Daten füllen.

Datenpflege

Datenpflege bedeutet, Configuration-Daten aktuell und nutzbar zu halten.

Dazu gehören:

- neue CIs erfassen,
- veränderte CIs aktualisieren,
- entfernte CIs archivieren,
- Beziehungen pflegen,
- Owner aktualisieren,
- Versionen korrigieren,
- Dokumentationslinks prüfen,
- Dubletten bereinigen,
- Statuswerte anpassen,
- und veraltete Einträge entfernen oder archivieren.

Pflege ist keine einmalige Projektarbeit.

Sie ist eine laufende Betriebsaufgabe.

Pflege durch Change Enablement

Changes sind eine der wichtigsten Quellen für CMDB-Aktualisierungen.

Nach einem Change kann sich ändern:

- Version,
- Konfiguration,
- Beziehung,
- Status,
- Umgebung,

- Owner,
- Lieferant,
- Zertifikat,
- Schnittstelle,
- Monitoring,
- Backup,
- Dokumentation.

Deshalb sollte jeder relevante Change prüfen:

> Müssen Configuration-Daten aktualisiert werden?

Wenn Changes die CMDB nicht pflegen, veraltet sie automatisch.

Pflege durch Release und Deployment

Nach Releases und Deployments ändern sich häufig:

- Anwendungsversionen,
- Zielumgebungen,
- Infrastrukturkomponenten,
- Schnittstellen,
- Datenbankversionen,
- Konfigurationsstände,
- Container-Images,
- Dokumentation,
- Known Errors,
- Monitoringregeln.

Release- und Deployment-Prozesse sollten deshalb CMDB-Aktualisierung als Abschlussaufgabe enthalten.

Pflege durch Incident und Problem Management

Incident und Problem Management können Datenfehler sichtbar machen.

Beispiele:

- falscher Owner im Incident,
- fehlende Servicebeziehung,
- veraltete Version,
- nicht dokumentierte Schnittstelle,
- fehlender Known Error,

- nicht erfasstes Zertifikat,
- falsche Kritikalität,
- fehlender Dokumentationslink.

Solche Erkenntnisse sollten nicht im Ticket verschwinden.

Sie sollten zur Verbesserung der Configuration-Daten genutzt werden.

Pflege durch Service Owner

Service Owner sind wichtig für fachliche Informationen.

Sie können prüfen:

- stimmt die Servicebeschreibung,
- stimmt die Kritikalität,
- stimmt der Geschäftsprozessbezug,
- stimmen Supportzeiten,
- stimmen Benutzergruppen,
- stimmen Lieferanten,
- sind Abhängigkeiten vollständig,
- sind Dokumentationslinks aktuell,
- sind Risiken korrekt bewertet.

Technische Teams kennen oft die Komponenten.

Service Owner kennen den fachlichen Wert und die Auswirkung.

Pflege durch technische Teams

Technische Teams pflegen vor allem technische Informationen.

Beispiele:

- Versionen,
- Konfigurationsstände,
- Plattformen,
- Server,
- Datenbanken,
- Netzwerkkomponenten,
- Zertifikate,
- Backupjobs,
- Monitoringregeln,
- Schnittstellen,
- Betriebshandbücher.

Wichtig ist, dass technische Änderungen nicht nur umgesetzt, sondern auch nachvollziehbar dokumentiert werden.

Pflegeintervalle

Nicht alle Configuration-Daten müssen gleich häufig geprüft werden.

Mögliche Prüfintervalle:

CI oder Information	mögliches Prüfintervall
geschäftskritischer Service	regelmäßig und nach jedem relevanten Change
Zertifikate	automatisch überwachen und regelmäßig prüfen
produktive Server	nach Changes und in festen Reviews
Cloud-Ressourcen	häufig oder automatisiert
Dokumentationslinks	regelmäßig oder nach Releases
Testsysteme	nach Bedarf
archivierte CIs	selten, aber nachvollziehbar

Die Häufigkeit sollte sich nach Risiko, Kritikalität und Änderungsdynamik richten.

Datenqualität

Datenqualität beschreibt, ob Configuration-Daten für den vorgesehenen Zweck geeignet sind.

Wichtige Qualitätsmerkmale:

- korrekt,
- aktuell,
- vollständig genug,
- eindeutig,
- konsistent,
- nachvollziehbar,
- auffindbar,
- verantwortet,
- und nutzbar.

Perfekte Vollständigkeit ist selten realistisch.

Entscheidend ist, dass die Daten für Incident, Problem, Change, Release und Security Management ausreichend zuverlässig sind.

Korrektheit

Korrektheit bedeutet:

> Die Information stimmt mit der Realität überein.

Beispiele:

- Version ist richtig,
- Owner ist richtig,
- Status ist richtig,
- Servicezuordnung ist richtig,
- Beziehung ist richtig.

Falsche Daten sind gefährlich.

Beispiel:

Die CMDB zeigt einen Server als Testsystem, obwohl er produktiv genutzt wird.

Ein Change wird dadurch zu niedrig bewertet.

Aktualität

Aktualität bedeutet:

> Die Information ist nicht veraltet.

Beispiele für veraltete Daten:

- alter Owner,
- alte Version,
- außer Betrieb genommener Server ist noch aktiv,
- neues Zertifikat fehlt,
- Cloud-Ressource wurde gelöscht,
- Schnittstelle wurde geändert,
- Dokumentationslink zeigt auf alte Anleitung.

Aktualität entsteht durch Anbindung an Changes, Releases, Reviews und Discovery.

Vollständigkeit

Vollständigkeit bedeutet nicht, jedes Detail zu erfassen.

Sie bedeutet:

> Die für den Zweck notwendigen Informationen sind vorhanden.

Beispiel:

Für einen kritischen Service sollten mindestens vorhanden sein:

- Service Owner,
- technische Supportgruppe,
- wichtigste CIs,
- zentrale Abhängigkeiten,
- Kritikalität,
- Dokumentation,
- Notfallinformationen,
- und relevante Lieferanten.

Ein unwichtiger Testdienst benötigt weniger Detailtiefe.

Eindeutigkeit

Eindeutigkeit bedeutet:

> Jedes CI ist klar identifizierbar.

Probleme entstehen durch:

- doppelte Einträge,
- unterschiedliche Schreibweisen,
- unklare Namen,
- fehlende IDs,
- unklare Umgebungen,
- ähnliche Systemnamen.

Beispiel:

- APP01
- app01
- Portal-App-Server
- PRD-APP-PORTAL-01

könnten dasselbe CI beschreiben.

Solche Dubletten erschweren Reports, Changes und Incidents.

Konsistenz

Konsistenz bedeutet:

> Daten folgen einheitlichen Regeln.

Beispiele:

- gleiche Statuswerte,
- gleiche CI-Typen,
- einheitliche Namenskonvention,
- klare Beziehungstypen,
- einheitliche Umgebungsbezeichnungen,
- definierte Pflichtfelder.

Ungeeignet:

- Produktion,
- Prod,
- PRD,
- Live

werden ohne Regel gemischt.

Besser:

> Eine definierte Schreibweise wird verbindlich genutzt.

Nachvollziehbarkeit

Nachvollziehbarkeit bedeutet:

> Es ist erkennbar, woher eine Information kommt und warum sie geändert wurde.

Hilfreich sind:

- letzter Änderungszeitpunkt,
- letzter Prüftermin,
- verknüpfter Change,
- verantwortliche Person oder Gruppe,
- Datenquelle,
- Kommentar zur Abweichung,
- Archivierung statt unkontrollierter Löschung.

Nachvollziehbarkeit ist besonders wichtig bei Sicherheitsfragen, Audits und Major Incidents.

Nutzbarkeit

Daten sind nur wertvoll, wenn sie genutzt werden können.

Nutzbarkeit bedeutet:

- Informationen sind auffindbar,
- Felder sind verständlich,
- Beziehungen sind sichtbar,
- Reports sind sinnvoll,
- Daten passen zu Prozessen,
- Teams vertrauen den Informationen,
- Pflegeaufwand ist realistisch.

Eine CMDB mit vielen Feldern, die niemand versteht oder nutzt, erzeugt keinen Mehrwert.

Datenqualitätsregeln

Eine Organisation sollte Datenqualitätsregeln definieren.

Beispiele:

- produktive CIs benötigen Owner,
- produktive CIs benötigen Servicezuordnung,
- kritische Services benötigen dokumentierte Abhängigkeiten,
- Zertifikate benötigen Ablaufdatum und Owner,
- Changes müssen betroffene CIs verknüpfen,
- außer Betrieb genommene CIs werden archiviert,
- Dubletten werden regelmäßig bereinigt,
- Cloud-Ressourcen benötigen Tags.

Solche Regeln müssen kontrollierbar sein.

Datenqualitätskennzahlen

Mögliche Kennzahlen:

Kennzahl	Aussage
CIs ohne Owner	Verantwortlichkeit fehlt
CIs ohne Servicezuordnung	Servicebezug unklar
kritische Services ohne Abhängigkeiten	Auswirkungsanalyse unvollständig
CIs ohne letzte Prüfung	Aktualität unsicher
ablaufende Zertifikate ohne Owner	hohes Betriebsrisiko
Dublettenquote	Eindeutigkeit schlecht
Changes ohne CI-Verknüpfung	CMDB wird im Change-Prozess nicht genutzt
Incidents ohne CI-Verknüpfung	Trendanalyse erschwert

Kennzahlen sollten nicht nur gemessen werden.

Sie müssen zu Verbesserungen führen.

Datenqualitätsberichte

Datenqualitätsberichte können regelmäßig zeigen:

- welche CIs unvollständig sind,
- welche Owner fehlen,
- welche Beziehungen fehlen,
- welche CIs lange nicht geprüft wurden,
- welche Zertifikate bald ablaufen,
- welche Versionen veraltet sind,
- welche Changes keine CIs verknüpft haben,
- welche Services keine Abhängigkeiten besitzen.

Solche Berichte sollten Verantwortliche und nächste Schritte enthalten.

Nur eine Fehlerliste reicht nicht aus.

Dubletten bereinigen

Dubletten entstehen leicht durch manuelle Pflege, Discovery oder unterschiedliche Datenquellen.

Vorgehen:

1. mögliche Dubletten erkennen
2. prüfen, ob es wirklich dasselbe CI ist
3. führenden Datensatz festlegen

4. Beziehungen zusammenführen
5. verknüpfte Incidents, Problems und Changes prüfen
6. Dublette archivieren oder löschen
7. Ursache der Dublette beheben

Wichtig ist, nicht versehentlich unterschiedliche CIs zusammenzuführen.

Veraltete CIs archivieren

Nicht mehr aktive CIs sollten nicht einfach unkontrolliert gelöscht werden.

Besser ist oft:

- Status auf außer Betrieb setzen,
- Beziehungen prüfen,
- offene Incidents oder Changes prüfen,
- Dokumentation verknüpft lassen,
- Archivierungsdatum erfassen,
- Grund der Außerbetriebnahme dokumentieren.

So bleibt Nachvollziehbarkeit erhalten.

Das ist besonders wichtig bei Audits, Sicherheitsvorfällen oder späteren Analysen.

Abweichungen erkennen

Abweichungen zwischen CMDB und Realität können entstehen durch:

- ungeplante Changes,
- manuelle Änderungen,
- Notfallmaßnahmen,
- nicht dokumentierte Deployments,
- fehlerhafte Discovery,
- Cloud-Automatisierung,
- Lieferantenänderungen,
- Schatten-IT,
- vergessene Außerbetriebnahmen.

Abweichungen sollten nicht nur korrigiert werden.

Es sollte geprüft werden, warum sie entstanden sind.

Soll-Zustand und Ist-Zustand unterscheiden

Zustand	Bedeutung

Soll-Zustand	freigegebener oder erwarteter Zustand
Ist-Zustand	tatsächlich erkannter Zustand
Abweichung	Unterschied zwischen Soll und Ist

Beispiel:

Soll-Zustand:

> Server nutzt Version 5.9.

Ist-Zustand:

> Discovery erkennt Version 5.8.

Mögliche Ursachen:

- Update fehlgeschlagen,
- CMDB nicht aktualisiert,
- falscher Server gescannt,
- Rollback wurde durchgeführt,
- Change nicht dokumentiert.

Configuration Drift

Configuration Drift bedeutet, dass Systeme im Laufe der Zeit vom gewünschten Zustand abweichen.

Ursachen:

- manuelle Änderungen,
- Hotfixes,
- ungeplante Anpassungen,
- unterschiedliche Skriptversionen,
- nicht dokumentierte Changes,
- Lieferantenarbeiten,
- fehlende Automatisierung,
- fehlende Kontrolle.

Folgen:

- Tests werden unzuverlässig,

- Deployments schlagen fehl,
- Sicherheitslücken entstehen,
- Incidents werden schwerer analysiert,
- Rollback wird schwieriger.

Baselines und regelmäßige Prüfungen helfen gegen Configuration Drift.

Baselines nutzen

Eine Configuration Baseline beschreibt einen bekannten, freigegebenen Zustand.

Beispiele:

- Standardserverkonfiguration,
- genehmigte Anwendungsversion,
- geprüfte Firewall-Regelbasis,
- Standardclient,
- freigegebene Cloud-Tag-Struktur,
- definierter Release-Stand.

Baselines helfen bei:

- Vergleich mit aktuellem Zustand,
- Fehleranalyse,
- Rollback,
- Audit,
- Standardisierung,
- Sicherheitsprüfung.

Wenn ein System von der Baseline abweicht, sollte dokumentiert sein, warum.

Pflegeverantwortung

Ohne Verantwortung veraltet Configuration Management.

Zu klären ist:

- Wer ist für das CMDB-Modell verantwortlich?
- Wer pflegt technische Daten?
- Wer pflegt Serviceinformationen?
- Wer prüft Datenqualität?
- Wer bereinigt Dubletten?
- Wer kontrolliert Reports?
- Wer aktualisiert Daten nach Changes?
- Wer genehmigt Modelländerungen?

- Wer ist Ansprechpartner bei Datenfehlern?

Verantwortung kann auf mehrere Rollen verteilt sein.

Sie muss aber eindeutig sein.

Rollen bei der Datenpflege

Rolle	mögliche Aufgabe
Configuration Manager	Regeln, Modell und Datenqualität steuern
Service Owner	Servicebezug, Kritikalität und fachliche Daten prüfen
CI Owner	einzelne CIs verantworten
Supportgruppe	technische Informationen ergänzen
Change Manager	CMDB-Aktualisierung nach Changes sicherstellen
Security Team	sicherheitsrelevante Daten nutzen und prüfen
Asset Manager	Asset-, Vertrags- und Lebenszyklusdaten liefern

Die Rollen müssen nicht immer eigene Stellen sein.

Wichtig sind klare Zuständigkeiten.

Pflege durch Prozesse erzwingen

Datenpflege funktioniert besser, wenn sie in Prozesse eingebaut ist.

Beispiele:

- Change kann nicht abgeschlossen werden, ohne betroffene CIs zu prüfen.
- Release-Abschluss enthält CMDB-Aktualisierung.
- Incident-Kategorien verlangen Servicezuordnung.
- Zertifikate benötigen Owner und Ablaufdatum.
- Cloud-Ressourcen ohne Pflicht-Tags werden gemeldet.
- Außerbetriebnahme verlangt Abhängigkeitsprüfung.

Datenpflege sollte nicht nur auf freiwilliger Erinnerung beruhen.

Pflegeaufwand realistisch halten

Zu viel Detailtiefe führt zu Pflegeproblemen.

Fragen zur Begrenzung:

- Wird dieses Feld wirklich genutzt?
- Wer pflegt es?
- Wie oft ändert es sich?
- Kann es automatisiert werden?
- Hilft es bei Incident, Problem, Change oder Security?
- Ist es für einen kritischen Service notwendig?
- Entsteht durch fehlende Information ein echtes Risiko?

Daten, die niemand nutzt und niemand pflegt, sollten kritisch hinterfragt werden.

Toolunterstützung

Tools können unterstützen durch:

- Pflichtfelder,
- Workflows,
- Discovery-Import,
- Dublettenprüfung,
- Beziehungsansichten,
- Reports,
- Erinnerungen,
- Datenqualitätsregeln,
- Rollenrechte,
- Schnittstellen zu Monitoring und Cloud,
- Verknüpfung mit Incidents und Changes.

Ein Tool löst aber kein organisatorisches Problem.

Ohne klare Regeln und Verantwortung wird auch ein gutes Tool schlechte Daten enthalten.

Integration mit ITSM-Prozessen

Configuration-Daten sollten nicht isoliert gepflegt werden.

Sie sollten aktiv genutzt werden in:

- Incident Management,
- Problem Management,
- Change Enablement,
- Release Management,
- Service Level Management,
- Information Security Management,
- IT Asset Management,
- Continual Improvement.

Je häufiger die Daten im Alltag genutzt werden, desto eher fallen Fehler auf.

Ungenutzte Daten veralten schneller.

Datenqualität durch Nutzung verbessern

Wenn Service Desk und Fachteams CMDB-Daten aktiv nutzen, werden Fehler sichtbar.

Beispiele:

- falscher Owner wird bei Eskalation bemerkt,
- fehlende Beziehung fällt bei Change-Bewertung auf,
- veraltete Version fällt bei Security-Prüfung auf,
- fehlender Dokumentationslink fällt im Incident auf.

Wichtig ist, dass solche Fehler einfach gemeldet und korrigiert werden können.

Praxisbeispiel: Discovery findet unbekannten Server

Situation

Discovery erkennt einen Server, der nicht in der CMDB steht.

Prüfung

- Ist der Server produktiv?
- Welcher Service nutzt ihn?
- Wer ist Owner?
- Warum wurde er nicht erfasst?
- Gibt es Sicherheitsrisiken?
- Gibt es offene Ports?
- Muss ein Change oder Problem erstellt werden?

Ergebnis

Der Server wird entweder als CI aufgenommen oder kontrolliert außer Betrieb genommen.

Praxisbeispiel: Falscher Owner

Situation

Ein Incident betrifft eine Anwendung.

Die CMDB nennt ein Team, das seit Monaten nicht mehr zuständig ist.

Folge

Eskalation verzögert sich.

Verbesserung

- Owner wird korrigiert,
- ähnliche Services werden geprüft,
- regelmäßige Owner-Reviews werden eingeführt,
- Change-Übergaben müssen künftig Owner aktualisieren.

Praxisbeispiel: Veraltete Version

Situation

Die CMDB zeigt Version 5.9.

Discovery erkennt Version 5.8.

Mögliche Ursachen

- Update wurde nicht durchgeführt,
- Rollback wurde nicht dokumentiert,
- CMDB wurde zu früh aktualisiert,
- Discovery scannt falsches System,
- mehrere ähnliche CIs existieren.

Verbesserung

Abweichung wird geprüft, korrigiert und mit Change Record verknüpft.

Praxisbeispiel: Fehlende Zertifikatsdaten

Situation

Ein produktives Zertifikat läuft ab.

In der CMDB ist kein Ablaufdatum gepflegt.

Folge

Anmeldedienst fällt aus.

Verbesserung

- Zertifikate als eigene CIs erfassen,

- Ablaufdatum verpflichtend machen,
- Owner festlegen,
- Monitoring für Ablaufdaten einführen,
- Erneuerungs-Runbook verknüpfen.

Praxisbeispiel: Cloud-Ressource ohne Tags

Situation

Cloud-Report zeigt mehrere Ressourcen ohne Service-Tag.

Risiko

- Kostenstelle unklar,
- Owner unbekannt,
- Kritikalität unbekannt,
- Sicherheitsbewertung schwierig,
- Außerbetriebnahme riskant.

Verbesserung

- Pflicht-Tags definieren,
- Erstellung ohne Tags technisch verhindern oder melden,
- Ressourcen nachträglich zuordnen,
- Verantwortliche benennen.

Typische Fehler

Fehler 1

Discovery-Daten werden ungeprüft in die CMDB übernommen.

Fehler 2

Manuelle Pflege wird niemandem eindeutig zugeordnet.

Fehler 3

CMDB wird einmal aufgebaut und danach nicht gepflegt.

Fehler 4

Changes aktualisieren Configuration-Daten nicht.

Fehler 5

Dubletten werden ignoriert.

Fehler 6

Datenqualitätsberichte werden erstellt, aber nicht bearbeitet.

Fehler 7

Zu viele Felder werden gepflegt, obwohl sie niemand nutzt.

Fehler 8

Wichtige Beziehungen fehlen trotz vieler technischer Details.

Fehler 9

Cloud- und Container-Ressourcen werden nicht berücksichtigt.

Fehler 10

Zertifikate, Schnittstellen und Benutzergruppen werden vergessen.

Fehler 11

Owner und Kritikalität werden nicht regelmäßig geprüft.

Fehler 12

Teams vertrauen der CMDB nicht mehr, weil Fehler nicht korrigiert werden.

Checkliste Discovery

- [ ] relevante Datenquellen festgelegt
- [ ] Discovery-Bereich definiert
- [ ] erkannte Objekte geprüft
- [ ] Dublettenprüfung durchgeführt
- [ ] Servicebezug ergänzt

- [ ] Owner ergänzt
- [ ] Umgebung geprüft
- [ ] Status geprüft
- [ ] technische Versionen übernommen
- [ ] fachliche Kritikalität ergänzt
- [ ] Sicherheitsrelevanz geprüft
- [ ] nicht relevante Objekte ausgeschlossen oder archiviert

Checkliste Datenpflege

- [ ] neue CIs werden erfasst
- [ ] geänderte CIs werden aktualisiert
- [ ] entfernte CIs werden archiviert
- [ ] Owner werden gepflegt
- [ ] Versionen werden aktualisiert
- [ ] Beziehungen werden gepflegt
- [ ] Dokumentationslinks werden geprüft
- [ ] Zertifikate werden überwacht
- [ ] Cloud-Tags werden geprüft
- [ ] Changes aktualisieren CMDB-Daten
- [ ] Releases aktualisieren CMDB-Daten
- [ ] Datenfehler können einfach gemeldet werden

Checkliste Datenqualität

- [ ] produktive CIs haben Owner
- [ ] produktive CIs haben Servicezuordnung
- [ ] kritische Services haben Abhängigkeiten
- [ ] CIs haben eindeutige Namen
- [ ] Dubletten werden bereinigt
- [ ] Statuswerte sind aktuell
- [ ] Versionen sind korrekt
- [ ] Umgebungen sind eindeutig
- [ ] Zertifikate besitzen Ablaufdatum
- [ ] Dokumentationslinks funktionieren

- [ ] letzte Prüfung ist dokumentiert
- [ ] Datenqualitätsberichte werden bearbeitet

Checkliste nach Changes

- [ ] betroffene CIs geprüft
- [ ] neue CIs angelegt
- [ ] entfernte CIs archiviert
- [ ] Versionen aktualisiert
- [ ] Beziehungen aktualisiert
- [ ] Owner geprüft
- [ ] Dokumentationslinks aktualisiert
- [ ] Monitoringbeziehungen geprüft
- [ ] Backupbeziehungen geprüft
- [ ] Known Errors aktualisiert, falls relevant
- [ ] Service Map aktualisiert
- [ ] Change Record mit CMDB verknüpft

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker tragen stark zur Datenqualität bei.

Im Alltag bedeutet das:

- technische Änderungen nicht nur ausführen, sondern dokumentieren,
- Versionen und Konfigurationen korrekt erfassen,
- falsche CMDB-Daten melden,
- Owner und Servicebezug prüfen,
- Discovery-Ergebnisse fachlich einordnen,
- nach Changes Beziehungen aktualisieren,
- Zertifikate, Schnittstellen und Monitoring berücksichtigen,
- und Datenqualität als Teil stabilen Betriebs verstehen.

Eine gute CMDB entsteht nicht allein durch ein Tool.

Sie entsteht durch saubere technische Arbeit, klare Prozesse und konsequente Pflege.

Zusammenfassung

> technische Daten durch Discovery erkennen
> ↓
> Ergebnisse prüfen und Dubletten vermeiden
> ↓
> fachliche Informationen ergänzen
> ↓
> Owner, Servicebezug und Kritikalität pflegen
> ↓
> Changes und Releases zur Aktualisierung nutzen
> ↓
> Datenqualität regelmäßig messen
> ↓
> Abweichungen korrigieren
> ↓
> Pflegeaufwand realistisch halten
> ↓
> Configuration-Daten aktiv in ITSM-Prozessen nutzen

Merksätze

> Discovery erkennt Technik, aber nicht automatisch fachliche Bedeutung.

> Datenpflege ist eine laufende Betriebsaufgabe, kein einmaliges Projekt.

> Eine CMDB veraltet automatisch, wenn Changes sie nicht aktualisieren.

> Datenqualität ist wichtiger als Datenmenge.

> Schlechte Configuration-Daten können zu falschen Entscheidungen führen.

> Je häufiger Daten im Alltag genutzt werden, desto schneller fallen Fehler auf.

> Gute Datenqualität braucht Toolunterstützung, Prozesse und Verantwortung.

Verwandte Seiten

- 6.1 Service Configuration Management – Ziele, Begriffe und Grundlagen
- 6.2 Configuration Items und Beziehungen
- 6.3 Configuration Management Database
- 6.5 Service Configuration Management im Zusammenspiel mit Incident-, Problem- und Change-Management
- Change Enablement
- Release Management
- Incident Management
- Problem Management
- Information Security Management
- IT Asset Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Service Configuration Management
- PeopleCert – ITIL Practice Guide: IT Asset Management
- PeopleCert – ITIL Practice Guide: Change Enablement
- PeopleCert – ITIL Practice Guide: Release Management
- PeopleCert – ITIL Practice Guide: Information Security Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten:

- Discovery-Beispiele,
- Datenqualitätsregeln,
- Pflegeprozesse,
- Prüffragen,
- Checklisten,
- Datenqualitätskennzahlen,
- und Praxisbeispiele

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Discovery-Methode,

- Datenqualitätskennzahl,
- Pflegefrequenz,
- Pflichtfeldstruktur,
- Toolintegration,
- oder CMDB-Governance-Struktur

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- Organisationsgröße,
- Toollandschaft,
- Datenquellen,
- Cloud-Nutzung,
- Sicherheitsanforderungen,
- Change-Modell,
- Supportmodell,
- und verfügbare Fähigkeiten

angepasst werden.

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