7.5 Knowledge Management im Zusammenspiel mit Incident, Problem und Change

> **Kurz erklärt**
>
> Knowledge Management wirkt nicht isoliert.
>
> Es verbindet Wissen aus Incident Management, Problem Management, Change Enablement, Release Management, Service Desk und Service Configuration Management.
>
> Ziel ist, dass Erfahrungen, Lösungen, Workarounds, Known Errors, Runbooks und Benutzerinformationen im richtigen Moment verfügbar sind.

Warum das Zusammenspiel wichtig ist

Im IT-Betrieb entsteht Wissen ständig während der Arbeit.

Beispiele:

- ein Incident wird gelöst,
- ein Workaround wird gefunden,
- ein Problem wird analysiert,
- ein Known Error wird dokumentiert,
- ein Change verändert einen Service,
- ein Release bringt neue Funktionen,
- ein Benutzerartikel wird nach Rückfragen verbessert,
- ein Major Incident erzeugt Lessons Learned.

Wenn dieses Wissen nicht in Knowledge Management einfließt, geht es verloren.

Dann entstehen typische Probleme:

- gleiche Incidents werden immer wieder neu analysiert,
- Service Desk nutzt veraltete Informationen,
- Workarounds stehen nur in einzelnen Tickets,
- Benutzer erhalten widersprüchliche Antworten,
- Knowledge-Artikel passen nicht mehr zur aktuellen Version,
- Known Errors sind nicht auffindbar,
- Changes verändern Services, aber Artikel bleiben alt,
- und Verbesserungen werden nicht dauerhaft übernommen.

Grundidee des Zusammenspiels

```
Incident tritt auf
        ↓
Service Desk nutzt vorhandenes Wissen
        ↓
fehlendes oder falsches Wissen wird erkannt
        ↓
Problem Management analysiert wiederkehrende Ursachen
        ↓
Known Error oder Workaround entsteht
        ↓
Knowledge Management macht dieses Wissen nutzbar
        ↓
Change oder Release setzt dauerhafte Lösung um
        ↓
Knowledge-Artikel, Runbooks und FAQ werden aktualisiert
        ↓
zukünftige Incidents werden schneller gelöst oder vermieden
```

Knowledge Management ist damit ein Lernkreislauf im IT-Betrieb.

Zusammenspiel mit Incident Management

Incident Management nutzt Knowledge, um Services schneller wiederherzustellen.

Nützlich sind:

- bekannte Symptome,
- Standardlösungen,
- Workarounds,
- Known Errors,
- Prüfschritte,
- Eskalationsregeln,
- Benutzerinformationen,
- Runbooks,
- Service-Desk-Artikel.

Beispiel:

Ein Benutzer meldet, dass VPN nach dem Ruhezustand nicht mehr verbindet.

Der Service Desk findet einen Knowledge-Artikel mit:

- betroffenem VPN-Client,
- Prüfschritten,
- Workaround,
- Eskalationskriterien,
- Verknüpfung zum Known Error.

Dadurch muss der Incident nicht neu analysiert werden.

Incident Management liefert Wissen zurück

Incidents zeigen, welches Wissen fehlt oder verbessert werden muss.

Beispiele:

- neuer Fehler tritt auf,
- bekannter Workaround funktioniert nicht mehr,
- Artikel ist unverständlich,
- Suchbegriff fehlt,
- Eskalationsweg ist falsch,
- Benutzer stellen immer dieselbe Frage,
- Service Desk ergänzt immer wieder dieselbe Erklärung,
- Artikel passt nicht zur aktuellen Version.

Diese Erkenntnisse sollten in Knowledge Management zurückfließen.

Ein gelöster Incident kann also Auslöser für einen neuen oder verbesserten Knowledge-Artikel sein.

Wann aus einem Incident ein Knowledge-Artikel entstehen sollte

Ein Knowledge-Artikel ist sinnvoll, wenn:

- der Incident wahrscheinlich wieder auftreten wird,
- die Lösung wiederverwendbar ist,
- ein Workaround gefunden wurde,
- Service Desk künftig schneller reagieren kann,
- Benutzer eine Anleitung benötigen,
- eine Fehlermeldung häufig vorkommt,
- eine Eskalation klar geregelt werden muss,
- oder der Fall für neue Mitarbeitende hilfreich ist.

Nicht jeder einzelne Incident benötigt einen eigenen Artikel.

Aber wiederkehrende oder typische Fälle sollten dokumentiert werden.

Zusammenspiel mit Problem Management

Problem Management erzeugt besonders wertvolles Wissen.

Beispiele:

- Ursachenanalyse,
- bekannte Ursachen,
- beitragende Faktoren,
- Known Errors,
- Workarounds,
- dauerhafte Lösungen,
- Risiken,
- Lessons Learned,
- wiederkehrende Muster.

Dieses Wissen darf nicht nur im Problem Record bleiben.

Es muss so aufbereitet werden, dass es im Service Desk, in Fachgruppen oder im Self-Service genutzt werden kann.

Problem Management liefert Known Errors

Ein Known Error sollte in Knowledge Management nutzbar gemacht werden.

Ein guter Known-Error-Artikel enthält:

- Fehlerbild,
- betroffener Service,
- betroffene Version,
- Ursache oder wahrscheinliche Ursache,
- Workaround,
- Risiken,
- Einschränkungen,
- Eskalationskriterien,
- dauerhafte Lösung,
- Status,
- Problem-Verknüpfung,
- Review-Datum.

Dadurch kann Incident Management neue Incidents schneller zuordnen.

Knowledge Management unterstützt Problem Management

Knowledge Management hilft Problem Management durch:

- historische Artikel,
- bekannte Workarounds,
- häufig genutzte Lösungen,
- Suchanfragen ohne Treffer,
- wiederkehrende Service-Desk-Fragen,
- schlecht bewertete Artikel,
- Hinweise auf Knowledge-Lücken,
- Nutzungsauswertungen,
- Verknüpfungen zu Incidents.

Beispiel:

Ein Workaround-Artikel wird sehr häufig genutzt.

Das kann ein Hinweis sein, dass ein Problem weiterhin offen ist und eine dauerhafte Lösung benötigt.

Zusammenspiel mit Change Enablement

Changes verändern Services, Systeme, Prozesse und Benutzeroberflächen.

Dadurch kann Wissen veralten.

Nach einem Change müssen möglicherweise aktualisiert werden:

- Benutzerartikel,
- FAQ,
- Service-Desk-Artikel,
- Runbooks,
- Known Errors,
- Workarounds,
- Screenshots,
- Menüpfade,
- Eskalationswege,
- Servicebeschreibungen,
- Monitoringhinweise,
- CMDB-Verknüpfungen.

Ein Change ist nicht vollständig abgeschlossen, wenn die technische Änderung umgesetzt wurde, aber das Wissen noch den alten Zustand beschreibt.

Knowledge-Prüfung im Change-Abschluss

Beim Abschluss eines Changes sollte geprüft werden:

- Welche Artikel sind betroffen?
- Haben sich Schritte, Screenshots oder Menüpfade geändert?
- Gibt es neue Fehlermeldungen?
- Wurde ein Known Error behoben?
- Ist ein Workaround noch gültig?
- Muss ein Benutzerartikel erstellt werden?
- Muss der Service Desk informiert werden?
- Muss ein Runbook angepasst werden?
- Muss ein Artikel archiviert werden?

Diese Prüfung sollte Teil der Change-Nachbereitung sein.

Beispiel: Change macht Wissen veraltet

Situation

Eine Anwendung erhält eine neue Oberfläche.

Problem

Der Self-Service-Artikel beschreibt noch die alte Menüführung.

Folge

Benutzer finden die Funktion nicht und erstellen Tickets.

Verbesserung

Change Enablement ergänzt eine Pflichtprüfung:

> Welche Knowledge-Artikel, FAQ und Screenshots müssen vor oder nach dem Change aktualisiert werden?

Zusammenspiel mit Release Management

Releases bringen neue oder geänderte Funktionen.

Knowledge Management unterstützt Releases durch:

- Release Notes,
- Benutzerinformationen,
- FAQ,
- Service-Desk-Briefings,

- Known Errors,
- bekannte Einschränkungen,
- neue Workarounds,
- aktualisierte Runbooks,
- Schulungsunterlagen.

Ein Release ohne Knowledge-Vorbereitung führt häufig zu unnötigen Rückfragen.

Release-Wissen vor Veröffentlichung vorbereiten

Vor einem Release sollte geklärt werden:

- Was ändert sich für Benutzer?
- Welche Funktionen sind neu?
- Welche Funktionen ändern sich?
- Welche bekannten Einschränkungen bestehen?
- Welche Artikel müssen aktualisiert werden?
- Welche Fragen werden Benutzer wahrscheinlich stellen?
- Welche Informationen benötigt der Service Desk?
- Welche Known Errors sind relevant?
- Welche Workarounds sind freigegeben?

Gute Release-Kommunikation reduziert Incidents nach der Einführung.

Zusammenspiel mit Service Desk

Der Service Desk ist gleichzeitig Nutzer und Lieferant von Wissen.

Er nutzt Knowledge für:

- schnelle Antworten,
- Standardlösungen,
- Workarounds,
- Eskalation,
- Benutzerkommunikation,
- Ticketdokumentation.

Er liefert Knowledge zurück durch:

- Feedback zu Artikeln,
- Hinweise auf fehlende Inhalte,
- wiederkehrende Fragen,
- neue Fehlerbilder,
- Suchbegriffe aus der Praxis,
- Rückmeldungen von Benutzern.

Der Service Desk sollte deshalb aktiv in Knowledge Management eingebunden sein.

Service Desk als Qualitätsfilter

Der Service Desk erkennt schnell, ob Wissen im Alltag funktioniert.

Typische Rückmeldungen:

- Artikel wird nicht gefunden,
- Titel passt nicht zur Benutzersprache,
- Schritte sind unklar,
- Screenshot ist veraltet,
- Workaround funktioniert nicht,
- Eskalationsweg stimmt nicht,
- Artikel ist zu lang,
- Artikel enthält zu viele interne Details,
- Benutzer verstehen die Anleitung nicht.

Diese Rückmeldungen sollten regelmäßig ausgewertet werden.

Zusammenspiel mit Self-Service

Self-Service nutzt Knowledge, damit Benutzer einfache Anliegen selbst lösen können.

Geeignet sind:

- FAQ,
- Benutzeranleitungen,
- Statusinformationen,
- bekannte sichere Workarounds,
- Servicebeschreibungen,
- Request-Formulare,
- Passwort- und MFA-Hilfen,
- Softwarekatalog,
- Onboarding-Anleitungen.

Self-Service braucht besonders klare Sprache.

Interne Diagnoseschritte oder sicherheitskritische Informationen gehören nicht ungeprüft in Benutzerartikel.

Self-Service liefert Wissen zurück

Self-Service-Daten zeigen, was Benutzer wirklich suchen.

Wichtige Signale:

- häufige Suchanfragen,
- Suchanfragen ohne Treffer,
- häufig geöffnete Artikel,
- schlechte Bewertungen,
- hohe Formularabbrüche,
- viele Tickets nach Artikelaufruf,
- wiederkehrende Begriffe aus Benutzertexten.

Diese Daten helfen, Knowledge-Artikel, FAQ und Formulare zu verbessern.

Zusammenspiel mit Service Configuration Management

Service Configuration Management liefert Kontext zu Wissen.

Wichtige Verknüpfungen:

- Artikel zu Service,
- Artikel zu Configuration Item,
- Known Error zu betroffener Version,
- Runbook zu Anwendung oder Server,
- Workaround zu Problem Record,
- Benutzerartikel zu Servicekatalogeintrag,
- Release Notes zu Anwendungsversion.

Ohne Service- oder CI-Bezug sind Knowledge-Artikel schwerer auffindbar und schwerer aktuell zu halten.

Beispiel: Knowledge und CI-Verknüpfung

Ein Artikel beschreibt einen VPN-Fehler.

Sinnvolle Verknüpfungen:

- Service: VPN-Zugang,
- CI: VPN-Client-Version,
- CI: VPN-Gateway,
- CI: Identity Provider,
- Problem Record,
- Known Error,
- geplanter Change.

Dadurch kann der Artikel bei passenden Incidents automatisch vorgeschlagen oder schneller gefunden werden.

Zusammenspiel mit Information Security Management

Sicherheitswissen muss besonders sorgfältig gepflegt werden.

Beispiele:

- Phishing melden,
- verlorenes Gerät melden,
- MFA-Probleme,
- Passwortregeln,
- verdächtige Anmeldung,
- Sicherheitsvorfall-Prozess,
- Umgang mit sensiblen Daten,
- sichere Dateiablage,
- Meldewege,
- Verhalten bei Verdacht auf Kompromittierung.

Sicherheitsartikel müssen verständlich sein.

Sie dürfen aber keine internen Schutzmechanismen unnötig offenlegen.

Sicherheitsrelevante Knowledge-Artikel

Bei sicherheitsrelevanten Artikeln ist zu prüfen:

- Wer darf den Artikel sehen?
- Sind interne Details geschützt?
- Ist die Anleitung missbrauchssicher?
- Gibt es klare Meldewege?
- Ist der Inhalt fachlich freigegeben?
- Ist der Artikel aktuell?
- Ist ein Review-Datum gesetzt?
- Sind Datenschutzanforderungen berücksichtigt?

Falsches Sicherheitswissen kann zu hohen Risiken führen.

Zusammenspiel mit Continual Improvement

Knowledge Management liefert viele Hinweise für Verbesserungen.

Beispiele:

- häufige Suchanfragen ohne Treffer,
- viele Tickets trotz Self-Service-Artikel,
- häufig genutzte Workarounds,

- schlecht bewertete Artikel,
- veraltete Runbooks,
- fehlende Knowledge-Verknüpfungen,
- wiederkehrende Benutzerfragen,
- viele Eskalationen wegen fehlender Prüfschritte.

Diese Erkenntnisse sollten in Continual Improvement einfließen.

Knowledge als Verbesserungsquelle

Ein sinnvoller Verbesserungsablauf:

```
Knowledge-Daten auswerten
        ↓
Lücken und Qualitätsprobleme erkennen
        ↓
Verbesserungsmaßnahme definieren
        ↓
Owner und Termin festlegen
        ↓
Artikel, Formular oder Prozess verbessern
        ↓
Wirkung messen
        ↓
Ergebnis in Continual Improvement übernehmen
```

Knowledge Management ist damit nicht nur Dokumentation, sondern auch Verbesserungsarbeit.

Wissen im Lebenszyklus eines Incidents

```
Benutzer meldet Störung
        ↓
Service Desk sucht passenden Artikel
        ↓
Artikel liefert Prüfschritte oder Workaround
        ↓
Incident wird gelöst oder eskaliert
        ↓
Ticket wird mit Artikel verknüpft
```

```
↓
Feedback zum Artikel wird dokumentiert
↓
Knowledge-Artikel wird bei Bedarf verbessert
```

So wird jeder Incident zu einer möglichen Quelle für besseres Wissen.

Wissen im Lebenszyklus eines Problems

```
Wiederkehrende Incidents werden erkannt
↓
Problem Record wird erstellt
↓
Ursache oder wahrscheinliche Ursache wird analysiert
↓
Known Error wird dokumentiert
↓
Workaround wird erstellt
↓
Service Desk erhält Knowledge-Artikel
↓
dauerhafte Lösung wird verfolgt
↓
Artikel wird nach Lösung aktualisiert oder archiviert
```

Problem Management und Knowledge Management arbeiten hier besonders eng zusammen.

Wissen im Lebenszyklus eines Changes

```
Change wird geplant
↓
betroffene Artikel und Runbooks werden identifiziert
↓
Service Desk und Benutzerinformationen werden vorbereitet
↓
Change wird umgesetzt
↓
Wissen wird aktualisiert
```

```
        ↓
alte Artikel werden angepasst oder archiviert
        ↓
Feedback nach Change wird ausgewertet
```

So verhindert Change Enablement, dass veraltetes Wissen im Betrieb bleibt.

Wissensverknüpfungen

Gute Knowledge-Arbeit nutzt Verknüpfungen.

Mögliche Verknüpfungen:

- Knowledge-Artikel zu Incident,
- Knowledge-Artikel zu Problem,
- Known Error zu Workaround,
- Workaround zu Service,
- Runbook zu CI,
- Benutzerartikel zu Servicekatalog,
- Artikel zu Change,
- Artikel zu Release,
- Artikel zu Sicherheitsmeldung.

Verknüpfungen verbessern Auffindbarkeit, Auswertung und Pflege.

Rollen im Zusammenspiel

Rolle	Beitrag zu Knowledge Management
Service Desk	nutzt Artikel, gibt Feedback, erkennt Wissenslücken
Fachteam	liefert technische Inhalte und prüft Richtigkeit
Problem Management	liefert Known Errors, Ursachen und Workarounds
Change Management	sorgt für Aktualisierung nach Changes
Release Management	liefert Release Notes und neue Informationen
Service Owner	prüft fachliche Richtigkeit und Benutzerwirkung
Security Team	prüft sicherheitsrelevante Inhalte
Knowledge Owner	verantwortet Pflege, Review und Qualität

Rollen können je nach Organisation unterschiedlich benannt sein.

Wichtig sind klare Verantwortlichkeiten.

Knowledge Owner

Ein Knowledge Owner sorgt dafür, dass ein Artikel aktuell und korrekt bleibt.

Aufgaben:

- fachliche Prüfung,
- Review,
- Feedback bewerten,
- Änderungen einarbeiten,
- veraltete Inhalte archivieren,
- Zielgruppe prüfen,
- Zugriffsschutz prüfen,
- Verknüpfungen pflegen.

Ohne Knowledge Owner veralten wichtige Artikel schnell.

Service Owner

Der Service Owner hilft zu klären:

- welche Benutzerinformationen nötig sind,
- welche Servicebeschreibung korrekt ist,
- welche Auswirkungen ein Known Error hat,
- welche Kommunikation freigegeben ist,
- welche Artikel für den Service wichtig sind,
- welche Inhalte nach Changes angepasst werden müssen.

Service Owner verbinden technische Informationen mit Service- und Benutzerperspektive.

Fachteams

Fachteams liefern technisches Wissen.

Beispiele:

- Logauswertung,
- Ursache,
- Konfigurationsdetails,
- technische Risiken,
- sichere Workarounds,
- Runbooks,
- Abbruchkriterien,
- Rollback-Hinweise,
- technische Prüfungen nach Changes.

Fachteams sollten Wissen so aufbereiten, dass Service Desk oder Benutzer es passend nutzen können.

Wissen für verschiedene Zielgruppen aufbereiten

Ein technischer Sachverhalt kann mehrere Wissensformen benötigen.

Beispiel: Zertifikatsproblem

Benutzerartikel

- aktuelle Einschränkung,
- was Benutzer tun können,
- wo Statusinformationen stehen.

Service-Desk-Artikel

- Symptome,
- betroffener Service,
- Eskalationsweg,
- freigegebene Kommunikation.

Fachteam-Runbook

- Zertifikat prüfen,
- Zertifikat erneuern,
- Dienst testen,
- Monitoring prüfen,
- CMDB aktualisieren.

Ein Artikel für alle Zielgruppen ist oft nicht sinnvoll.

Wissensqualität im Zusammenspiel

Wissensqualität bedeutet:

- richtige Information,
- richtige Zielgruppe,
- richtige Detailtiefe,
- richtige Sichtbarkeit,
- richtige Verknüpfung,
- aktueller Stand,
- klarer Owner,
- regelmäßiger Review.

Ein fachlich richtiger Artikel kann trotzdem schlecht sein, wenn er für die Zielgruppe unverständlich ist.

Ein gut geschriebener Artikel kann gefährlich sein, wenn er veraltet ist.

Kennzahlen für das Zusammenspiel

Mögliche Kennzahlen:

Kennzahl	mögliche Aussage
Incidents mit verknüpftem Knowledge-Artikel	Nutzung im Service Desk
häufig genutzte Artikel	wichtige Supportthemen
Suchanfragen ohne Treffer	Wissenslücken
Artikel ohne Owner	Pflegeproblem
Artikel ohne Review-Datum	Aktualitätsrisiko
Tickets trotz Self-Service-Artikel	Artikel nicht hilfreich oder Problem weiterhin häufig
Known Errors ohne Artikel	Wissen fehlt im Support
Changes ohne Knowledge-Prüfung	Risiko veralteter Artikel
schlechte Artikelbewertungen	Qualitätsproblem

Kennzahlen müssen mit Feedback und fachlicher Bewertung kombiniert werden.

Wissen aus Major Incidents

Major Incidents liefern besonders wichtige Erkenntnisse.

Nach einem Major Incident sollte geprüft werden:

- Welche Informationen haben gefehlt?
- Welche Runbooks waren hilfreich?
- Welche Artikel waren veraltet?
- Welche Kommunikation war unklar?
- Welche Workarounds wurden genutzt?
- Welche Known Errors entstanden?
- Welche Benutzerinformationen müssen verbessert werden?
- Welche Lessons Learned gehören in Knowledge Management?

Ein Major Incident Review ohne Knowledge-Aktualisierung verschenkt Lernpotenzial.

Wissen aus Failed Changes

Auch fehlgeschlagene Changes liefern wertvolles Wissen.

Zu prüfen ist:

- War das Runbook ausreichend?
- War der Rollback beschrieben?
- Waren Tests vollständig?
- Gab es fehlende Abhängigkeiten?
- War Service Desk informiert?
- Müssen Change-Vorlagen verbessert werden?
- Muss ein neuer Knowledge-Artikel entstehen?
- Muss ein Known Error dokumentiert werden?

Failed Changes sollten nicht nur technisch korrigiert, sondern auch wissensseitig ausgewertet werden.

Wissen aus Releases

Nach Releases sollte beobachtet werden:

- Welche Fragen stellen Benutzer?
- Welche Incidents entstehen?
- Welche Artikel werden häufig aufgerufen?
- Welche Suchbegriffe führen zu keinem Treffer?
- Welche Screenshots sind veraltet?
- Welche Known Errors sind neu?
- Welche FAQ fehlt?

So kann Knowledge Management schnell auf reale Benutzerfragen reagieren.

Praxisbeispiel: VPN-Problem

Incident Management

Benutzer melden VPN-Abbrüche.

Problem Management

Mehrere Incidents zeigen dasselbe Muster.

Ursache ist eine fehlerhafte Client-Version.

Knowledge Management

Ein Known-Error-Artikel wird erstellt.

Er enthält:

- Fehlerbild,
- betroffene Version,
- Prüfschritte,
- Workaround,
- Eskalationskriterien,
- geplanten Change.

Change Enablement

Neue Client-Version wird ausgerollt.

Nachbereitung

Knowledge-Artikel wird aktualisiert oder archiviert.

Praxisbeispiel: MFA-Self-Service

Beobachtung

Viele Tickets entstehen wegen neuem Smartphone.

Knowledge Management

Benutzerartikel wird erstellt:

- MFA auf neuem Smartphone einrichten,
- Screenshots,
- Hinweis bei fehlendem Zugriff,
- Ticketlink.

Service Desk

Tickets enthalten bessere Informationen.

Continual Improvement

Suchbegriffe und Formular werden verbessert.

Nutzen

Weniger Standardtickets und bessere Benutzererfahrung.

Praxisbeispiel: Change veraltet Runbook

Situation

Nach einem Update ändert sich der Neustartprozess eines Dienstes.

Problem

Das alte Runbook beschreibt noch die frühere Reihenfolge.

Risiko

Administratoren führen im Störungsfall falsche Schritte aus.

Verbesserung

Change-Abschluss enthält künftig:

- betroffene Runbooks prüfen,
- Knowledge Owner informieren,
- Review-Datum setzen,
- alte Version archivieren.

Praxisbeispiel: Major Incident

Situation

Ein zentraler Dienst fällt aus.

Review zeigt

- Service Desk hatte keinen passenden Artikel,
- Eskalationsweg war unklar,
- Statusmeldung musste improvisiert werden,
- Runbook für Wiederherstellung war veraltet.

Knowledge-Verbesserungen

- Service-Desk-Artikel erstellen,
- Major-Incident-Kommunikationsvorlage ergänzen,
- Runbook aktualisieren,
- Known Error dokumentieren,
- Service Map verlinken.

Typische Fehler

Fehler 1

Knowledge Management wird getrennt von Incident, Problem und Change betrieben.

Fehler 2

Gelöste Incidents erzeugen kein wiederverwendbares Wissen.

Fehler 3

Problem Management dokumentiert Known Errors, aber der Service Desk findet sie nicht.

Fehler 4

Workarounds bleiben nur in Tickets.

Fehler 5

Changes aktualisieren Knowledge-Artikel nicht.

Fehler 6

Releases erzeugen neue Benutzerfragen, aber keine FAQ.

Fehler 7

Service Desk Feedback wird nicht bearbeitet.

Fehler 8

Knowledge-Artikel sind nicht mit Services oder CIs verknüpft.

Fehler 9

Sicherheitsartikel sind entweder zu offen oder zu stark eingeschränkt.

Fehler 10

Major-Incident-Lessons-Learned werden nicht in Knowledge übernommen.

Fehler 11

Artikel werden an Kennzahlen gemessen, aber nicht fachlich geprüft.

Fehler 12

Niemand ist verantwortlich für Pflege und Review.

Checkliste Incident zu Knowledge

- [ ] wiederkehrendes Fehlerbild erkannt
- [ ] Lösung oder Workaround dokumentiert
- [ ] passende Suchbegriffe ergänzt
- [ ] Zielgruppe festgelegt
- [ ] Service oder CI verknüpft
- [ ] Eskalationsweg beschrieben
- [ ] Risiken genannt
- [ ] Knowledge-Artikel im Ticket verknüpft
- [ ] Feedback aus Ticket berücksichtigt
- [ ] Problem Management informiert, falls Muster erkennbar

Checkliste Problem zu Knowledge

- [ ] Known Error dokumentiert
- [ ] Ursache oder wahrscheinliche Ursache beschrieben
- [ ] Workaround beschrieben oder als nicht verfügbar gekennzeichnet
- [ ] betroffene Versionen genannt
- [ ] betroffene Services und CIs verknüpft
- [ ] Service Desk Artikel erstellt
- [ ] Benutzerinformation geprüft
- [ ] dauerhafte Lösung oder Change vermerkt
- [ ] Review-Datum gesetzt
- [ ] Artikel nach Lösung aktualisiert oder archiviert

Checkliste Change zu Knowledge

- [ ] betroffene Artikel identifiziert
- [ ] Benutzerartikel geprüft

- [ ] Service-Desk-Artikel geprüft
- [ ] Runbooks geprüft
- [ ] Screenshots geprüft
- [ ] Menüpfade geprüft
- [ ] Known Errors geprüft
- [ ] Workarounds geprüft
- [ ] Service Desk informiert
- [ ] Artikel nach Umsetzung aktualisiert
- [ ] veraltete Artikel archiviert
- [ ] Knowledge-Prüfung im Change Record dokumentiert

Checkliste Release zu Knowledge

- [ ] Release Notes erstellt
- [ ] neue Funktionen beschrieben
- [ ] geänderte Funktionen beschrieben
- [ ] bekannte Einschränkungen dokumentiert
- [ ] FAQ vorbereitet
- [ ] Service Desk gebrieft
- [ ] Benutzerkommunikation vorbereitet
- [ ] neue Known Errors dokumentiert
- [ ] alte Workarounds geprüft
- [ ] Suchanfragen nach Release ausgewertet
- [ ] Artikel nach Feedback verbessert

Checkliste Knowledge-Qualität im Zusammenspiel

- [ ] Artikel besitzt Owner
- [ ] Review-Datum vorhanden
- [ ] Zielgruppe klar
- [ ] Service oder CI verknüpft
- [ ] Problem oder Known Error verknüpft, falls relevant
- [ ] Change oder Release verknüpft, falls relevant
- [ ] Risiken und Grenzen beschrieben
- [ ] Eskalationsweg klar

- [ ] Zugriffsschutz passend
- [ ] Feedbackmöglichkeit vorhanden
- [ ] Nutzung wird ausgewertet
- [ ] veraltete Inhalte werden archiviert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker stehen häufig genau dort, wo wichtiges Wissen entsteht.

Im Arbeitsalltag bedeutet das:

- Lösungen aus Incidents dokumentieren,
- wiederkehrende Fehler an Problem Management melden,
- Workarounds sicher beschreiben,
- Known Errors technisch prüfen,
- Runbooks für Betriebsaufgaben erstellen,
- nach Changes Artikel aktualisieren,
- Service Desk mit Prüfschritten unterstützen,
- Configuration Items verknüpfen,
- Benutzerwissen technisch korrekt vorbereiten,
- und Lessons Learned in nutzbare Dokumentation übertragen.

Gute Wissensarbeit macht technischen Betrieb stabiler, schneller und weniger abhängig von Einzelpersonen.

Zusammenfassung

> Incident liefert Erfahrung
> ↓
> Problem Management erkennt Ursachen und Known Errors
> ↓
> Knowledge Management macht Wissen nutzbar
> ↓
> Service Desk verwendet Artikel im Alltag
> ↓
> Change und Release verändern Services
> ↓
> Knowledge wird aktualisiert
> ↓
> Self-Service stellt Benutzerwissen bereit
> ↓

> Feedback und Kennzahlen zeigen Lücken
> ↓
> Continual Improvement verbessert Inhalte und Prozesse

Merksätze

> Knowledge Management verbindet Lernen mit täglichem IT-Betrieb.

> Incidents liefern Hinweise auf fehlendes oder falsches Wissen.

> Problem Management liefert Known Errors, Ursachen und Workarounds.

> Changes und Releases müssen Knowledge aktualisieren.

> Service Desk Feedback ist eine der wichtigsten Quellen für Wissensqualität.

> Knowledge ohne Service- oder CI-Bezug ist schwerer nutzbar.

> Lessons Learned sind nur wertvoll, wenn sie in nutzbares Wissen überführt werden.

Verwandte Seiten

- 7.1 Knowledge Management – Ziele, Begriffe und Grundlagen
- 7.2 Knowledge-Artikel, Runbooks und FAQ
- 7.3 Known Errors, Workarounds und Wissensnutzung im Service Desk
- 7.4 Self-Service und Benutzerwissen
- Incident Management

- Problem Management
- Change Enablement
- Release Management
- Service Configuration Management
- Continual Improvement

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Knowledge Management
- PeopleCert – ITIL Practice Guide: Incident Management
- PeopleCert – ITIL Practice Guide: Problem Management
- PeopleCert – ITIL Practice Guide: Change Enablement
- PeopleCert – ITIL Practice Guide: Release Management
- PeopleCert – ITIL Practice Guide: Service Configuration Management
- PeopleCert – ITIL Practice Guide: Continual Improvement
- ITIL Foundation – Version 5

Ergänzende Praxiseinordnung

- Knowledge-Centered Service als verbreiteter Praxisansatz für Wissensarbeit im Support

Einordnung

Die dargestellten:

- Schnittstellen,
- Informationsflüsse,
- Rollenhinweise,
- Checklisten,
- Praxisbeispiele,
- und Qualitätskriterien

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Knowledge-Prozessschnittstelle,
- Artikelpflicht nach jedem Incident,
- Review-Frequenz,
- Verknüpfungsregel,
- Rollenmatrix,
- oder Toolintegration

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Zielgruppen,
- Supportmodell,
- Knowledge-Werkzeuge,
- Sicherheitsanforderungen,
- Problem Management,
- Change-Modell,
- Release-Modell,
- Service Configuration Management,
- und verfügbare Fähigkeiten

angepasst werden.

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