

IDENTITY GOVERNANCE & ADMINISTRATION

Zugriff ist eine Entscheidung

*Grundlagen einer Identity Governance,
die diesen Namen verdient*

Philipp Beermann

Legitimation · Ownership
Lifecycle · Evidenz

IMPRESSUM

Zugriff ist eine Entscheidung

Grundlagen einer Identity Governance, die diesen Namen verdient
Fachlektüre, 1. Ausgabe, September 2026

Philipp Beermann
Mühlenstr. 17, 45770 Marl
ig@pbeermann.de

© 2026 Philipp Beermann. Dieses Werk ist lizenziert unter der Creative Commons Lizenz
Namensnennung – Keine Bearbeitungen 4.0 International (CC BY-ND 4.0). Lizenztext:
creativecommons.org/licenses/by-nd/4.0

Das Unternehmen Nordfeld Logistik und alle in den Praxisbeispielen genannten Personen sind frei
erfunden. Ähnlichkeiten mit realen Unternehmen oder Personen sind zufällig.

Für Lektorat, Satz und Gestaltung wurden KI-gestützte Werkzeuge eingesetzt; die fachliche
Verantwortung liegt beim Autor. Die Inhalte stellen keine Rechts- oder Prüfungsberatung dar.

*Wie bleibt Zugriff fachlich legitimiert, zeitlich gültig
und nachweisbar steuerbar – auch dann, wenn sich
Organisation, Aufgaben und Risiken verändern?*

Inhalt

— **Vorwort: Warum dieser Text**

Was dieser Text nicht ist · Was dieser Text ist · Die Leitfrage

1 **Orientierung und Grundlogik**

Zugriff ist eine Entscheidung, kein Ticket · IGA in einem Satz · Das Operating Model: Entscheidung, Umsetzung, Kontrolle, Korrektur · Approval ist nicht Accountability · Minimum Viable Evidence · Governance-Objekte und ihre Abgrenzung · Decision Governance als Steuerungsprinzip

2 **Decision Governance**

Welche Entscheidungen IGA wirklich steuert · Decision Records als Governance-Artefakt · Policy-to-Decision Trace · Entscheidungskriterien statt Bedarfsaussagen · Gültigkeit, Timeboxing und Review-Trigger · Entscheidungsqualität messbar machen · Anti-Pattern: Genehmigung ersetzt Entscheidung

3 **Ownership und Verantwortung**

Fachliche vs. operative Verantwortung · Application Owner, Role Owner, Data Owner · Control Owner und Kontrollverantwortung · RACI-Logiken für IGA · Delegation und Stellvertretung · Grenzen von Verantwortung · Anti-Pattern: Verantwortung ohne Ownership

4 **Lifecycle und Drift**

Lifecycle als Zustandslogik · Joiner als Governance-Startpunkt · Mover als Risikoereignis · Leaver und strukturierter Entzug · Reorganisationen als Zugriffstrigger · Projekt- und Mandatsende · Drift: Desired vs. Actual State · Anti-Pattern: Zugriff als Dauerzustand

5 **Ausnahmen und Sonderfälle**

Ausnahme als Governance-Objekt · Exception Records: Das dokumentierte Ausnahme-Objekt · Kompensationskontrollen: Der Sicherheitsmechanismus für Ausnahmen · Timeboxing von Ausnahmen: Das Ablaufdatum als Pflichtfeld · Exception Debt als Steuerungsproblem · Rückbau von Sonderfällen: Der Weg zurück zum Standard

6 **Zugriffsmodelle**

Rollenmodelle und ihre Grenzen · RBAC fachlich eingeordnet · ABAC und Attributqualität · ReBAC und die Kraft der Beziehungen · Scope- und Task-basierte Modelle · Modellkombinationen in der Praxis · Anti-Pattern: Modellwechsel ohne Governance

7 **Privileged Access**

Privileged Access als eigener Kontrollraum · Standard- vs. privilegierter Zugriff · Eligibility vs. Activation · Just-in-Time und Just-Enough fachlich erklärt · Schutz vor Privilegien-Eskalation über den Änderungspfad · Break-Glass als Ausnahme · Session Accountability · Reviews für privilegierten Zugriff

8 **Non-Person Identities**

Non-Person Identities als Governance-Domäne · NPI-Taxonomie · Ownership für technische Identitäten · NPI-Lifecycle · Least Privilege für NPIs · Rotation und Änderung · Anti-Pattern: Dauerhafte technische Admin-Zugriffe

9 **Evidenz & Audit**

Kontrollkette von Entscheidung bis Entzug · Präventive, detektive und korrektive Kontrollen · Evidenz vs. Logs · Wirksame Reviews und Attestierungen · Umgang mit Findings · Governance-KPIs

10 **Reifegrad & Skalierung**

IGA-Reifegradmodelle · Governance vor dem Antrag · Operating Rhythm und Risikoorientierung · Schnittstellen zu Risk, Audit und Compliance · IGA unter Veränderungsdruck · Verdichtung: Die zentralen Governance-Merksätze

VORWORT

Warum dieser Text

In der Praxis wiederholt sich ein Muster, unabhängig von Branche, Unternehmensgröße oder eingesetzter Technologie: Zugriffe werden bereitgestellt – aber nicht konsequent begründet, fachlich verantwortet, wirksam kontrolliert oder sauber nachweisbar gemacht.

Zwei Entwicklungen verstärken dieses Muster zusätzlich. Zum einen wachsen IAM- und IGA-Systeme schrittweise zu zentralen Datenhalden heran, nicht weil es fachlich sinnvoll wäre, sondern weil die Systeme ohnehin angebunden sind. Attribute, Strukturen und Sonderinformationen sammeln sich dort an, ohne klare Ownership oder belastbare Kontrolllogik. Zum anderen verlagert sich fachliche Verantwortung aus den Fachbereichen heraus: Zugriffe, Rollen oder Ausnahmen werden formal genehmigt, die eigentliche Verantwortung dafür jedoch implizit auf den IAM/IGA-Bereich abgeschoben. Technik kompensiert damit Organisationslücken, die eigentlich fachlich zu schließen wären.

Ein dritter Punkt kommt hinzu, der ebenso häufig zu beobachten ist: Fachbereiche definieren Regeln, Rollen und Berechtigungen, kontrollieren deren Wirksamkeit danach aber nicht ganzheitlich. Verantwortung wird als einmaliger Definitionsakt verstanden, nicht als fortlaufende Steuerungsaufgabe über den gesamten Lifecycle hinweg.

MERKSATZ

Governance beginnt dort, wo Verantwortung nicht mehr delegiert wird, sondern getragen werden muss.

Dieser Text setzt an dieser Lücke an. Es geht dabei nicht um die Frage, wer Zugriff technisch erhalten kann, sondern um vier andere Fragen: Wer entscheidet fachlich? Auf welcher Grundlage? Mit welcher Gültigkeit? Und wie bleibt das über den gesamten Lifecycle hinweg prüfbar?

Was dieser Text nicht ist

Es ist keine Tool- oder Produktbetrachtung und enthält keine Implementierungsanleitung. Wer eine Einführung in konkrete IGA-Software sucht, wird hier nicht fündig.

Was dieser Text ist

Es behandelt Identity Governance & Administration (IGA) als eigenständige Security- und Governance-Disziplin – mit Prinzipien, Verantwortungsmodellen, Lifecycle-Logik und Evidenzanforderungen, die für menschliche Identitäten, Non-Person Identities und privilegierte Zugriffe gleichermaßen gelten. IGA bedeutet in diesem Verständnis nicht zentrale Datensammlung ohne Ownership, nicht die Auslagerung fachlicher Verantwortung in die Technik und nicht Regeldefinition ohne Kontrolle. IGA bedeutet stattdessen: bewusste Entscheidungen über Zugriffe, mit Begründung, Verantwortung, Kontrolle und Nachweis.

MERKSATZ

Governance ist kein technisches Nebenprodukt, sondern die bewusste Steuerung von Entscheidungsrechten.

Die Leitfrage

Daraus ergibt sich die Frage, die diesen Text von der ersten bis zur letzten Seite trägt:

Wie bleibt Zugriff fachlich legitimiert, zeitlich gültig und nachweisbar steuerbar – auch dann, wenn sich Organisation, Aufgaben und Risiken verändern?

KAPITEL 1

1

Orientierung und Grundlogik

Zugriff ist eine Entscheidung, kein Ticket

In vielen Organisationen wird IGA auf ein simples Muster reduziert: Antrag rein, Berechtigung raus. Das funktioniert operativ – es beantwortet aber nicht die Fragen, die in Security und Audit tatsächlich zählen. Der Kern von Identity Governance & Administration liegt nicht in der Bereitstellung, sondern in der Steuerbarkeit. Zugriffe sollen nicht nur existieren, sondern begründet, verantwortet, kontrolliert und nachweisbar sein.

MERKSATZ

Ein Ticket dokumentiert eine Ausführung – erst die Begründung macht daraus eine steuerbare Entscheidung.

Fachlich lässt sich das an fünf Dimensionen festmachen: der Begründung (welche Policy oder Regel legitimiert den Zugriff), der Ownership (wer ist fachlich accountable für die Entscheidung), dem Lifecycle (was passiert bei Rollenwechsel, Reorganisation, Projektende oder Offboarding), Kontrolle und Evidenz (welche minimale Evidenz zeigt Wer, Was, Wann, Warum – und was folgt aus Findings) sowie dem Scope, der für menschliche Identitäten ebenso gilt wie für Non-Person Identities und privilegierte Zugriffe.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Ein neuer Mitarbeiter bei Nordfeld Logistik beantragt Zugriff auf LogiCore. Wird nur das Ticket „Zugriff gewährt“ abgehakt, ist das reines Provisioning. Governance beginnt erst, wenn dokumentiert ist: Wer hat entschieden (Sabine Krüger), auf welcher Grundlage (Rolle „Sachbearbeiter Vertrieb“), für wie lange (unbefristet, aber jährlich zu prüfen) und warum (Vertragsbeginn).

IGA in einem Satz

Auf einen Satz reduziert: IGA sorgt dafür, dass Zugriffe begründet, verantwortet, kontrolliert und nachweisbar sind. Diese vier Begriffe sind zugleich die Messlatte, an der sich jede IGA-Praxis prüfen lässt – fehlt einer davon, betreiben wir eher operatives Provisioning als echte Governance.

Begründet heißt: Ein Zugriff existiert nicht, „weil jemand ihn braucht“, sondern weil er durch ein Regelwerk legitimiert ist – etwa durch Policy, Rolle, Scope oder Aufgabe, stets inklusive Kontext und definierter Gültigkeit.

Verantwortet heißt: Für jede Entscheidung gibt es fachliche Accountability. Genehmigungen dürfen Verantwortung nicht ersetzen; ohne klare Ownership wird Zugriff schnell zum ungesteuerten Selbstläufer.

Kontrolliert heißt: Zugriffe sind kein einmaliger Akt, sondern müssen über den gesamten Lifecycle steuerbar bleiben – Joiner/Mover/Leaver, Reorganisationen, Projektende, inklusive aktivem Umgang mit Drift.

Nachweisbar heißt: Minimal ausreichende Evidenz beantwortet Wer, Was, Wann und Warum und zeigt die durchgängige Kette von der Entscheidung über die Umsetzung bis zur Kontrolle und zum Entzug.

MERKSATZ

Governance ist nicht der Besitz von Berechtigungen, sondern die Beherrschung ihrer Gültigkeit.

Das Operating Model: Entscheidung, Umsetzung, Kontrolle, Korrektur

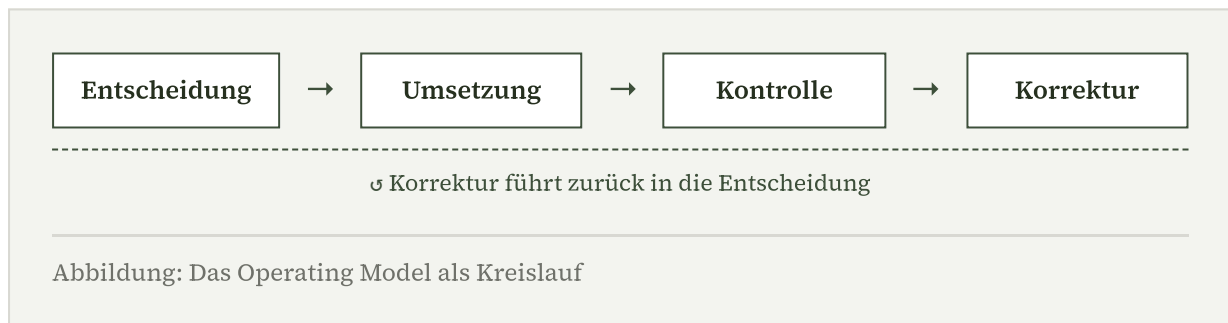
KURZ ERKLÄRT · OPERATING MODEL

Ein einfaches Denkraster mit vier Stationen, das sich auf fast jede Zugriffsfrage anwenden lässt: Wer entscheidet (Entscheidung)? Wie wird es technisch umgesetzt (Umsetzung)? Wie wird geprüft, ob es noch stimmt (Kontrolle)? Und was passiert, wenn nicht (Korrektur)?

Wird IGA nur als Provisioning verstanden, endet Verantwortung oft mit der Bereitstellung. Governance beginnt dort, wo Zugriffe als steuerbarer Zustand begriffen werden – inklusive Kontrolle, Nachweis und Korrektur. Dafür lässt sich ein Operating Model nutzen, das sich auf fast jede IGA-Fragestellung anwenden lässt:

Bei der **Entscheidung** stellt sich die Frage, wer die fachliche Entscheidung trifft, wer dafür die Verantwortung trägt und auf Basis welcher Kriterien – Policy, Aufgabe oder Risiko. Die **Umsetzung** muss operativ wirksam werden, ohne die fachliche Entscheidung zu ersetzen. Die **Kontrolle** klärt, woran erkennbar ist, dass ein Zugriff weiterhin korrekt ist; Abweichungen, Monitoring und belastbare Evidenz sind hier entscheidend. Die **Korrektur** schließlich regelt, was bei Abweichungen passiert: Entzug, Anpassung, befristete Ausnahme oder Ursachenbehebung.

Entscheidend ist: Dieses Modell ist kein Prozessdiagramm, sondern eine Verantwortungs- und Nachweislogik. Es zwingt dazu, Ownership und Evidenz entlang einer durchgängigen Kette zu denken – nicht nur am Genehmigungsknoten.



Governance ist erst dann real, wenn Korrektur vorgesehen ist.

Approval ist nicht Accountability

In vielen Organisationen gilt eine Freigabe bereits als ausreichender Nachweis – ein Klick ersetzt die Verantwortung. Genau hier beginnt eines der häufigsten Anti-Patterns der Disziplin. Ein Approval beantwortet nur eine operative Frage: Hat jemand zugestimmt? Accountability beantwortet die eigentlichen Governance-Fragen: Wer trägt fachlich die Verantwortung für diesen Zugriff, auf welcher Grundlage wurde entschieden, wie lange gilt diese Entscheidung, und was passiert, wenn sich der Kontext ändert?

Das typische Anti-Pattern sieht so aus: Ein Antrag wird genehmigt, der Zugriff bereitgestellt – und er bleibt bestehen, bis Jahre später jemand zufällig darüber stolpert. Der Governance-Ansatz dagegen verlangt, dass jede Entscheidung einen fachlich verantwortlichen Owner hat, an Kriterien und Gültigkeit gebunden ist, Kontrolle und Korrektur nicht optional vorsieht und Evidenz nicht nur zeigt, wer genehmigt hat, sondern wer dafür geradesteht.

Genehmigung ist ein Ereignis. Accountability ist ein Zustand.

Minimum Viable Evidence

Ein verbreitetes Missverständnis lautet: „Wir loggen alles, also sind wir audit-ready.“ In der Realität helfen tausende Zeilen technischer System-Logs bei einer Governance-Prüfung kaum weiter. Governance braucht keine Datenflut, sondern Evidenz – dafür eignet sich das Konzept der Minimum Viable Evidence (MVE): der kleinstmögliche Datensatz, der eine Entscheidung auch Jahre später noch fachlich legitimiert und nachweisbar macht.



Ein wirksamer Governance-Nachweis beantwortet vier Fragen. **Wer** hat die fachliche Entscheidung getroffen und wer ist der accountable Owner – nicht nur, wer den Zugriff erhalten hat. **Was** wurde konkret gewährt, in einer Sprache, die Fachbereich und Revision verstehen, nicht nur in technischen Gruppen-IDs. **Wann** besteht der Zugriff, und vor allem: bis wann ist er gültig, und wann wurde die Entscheidung zuletzt validiert? **Warum** legitimiert welche Policy, welche Rolle oder welche konkrete Aufgabe diesen Zugriff – „Bedarf angemeldet“ ist dabei keine Begründung, sondern eine bloße Behauptung.

Ohne diese vier Anker ist ein Audit-Trail wertlos: Er zeigt, dass etwas passiert ist, aber nicht, ob es rechtmäßig war. Ein technisches Log dokumentiert eine Aktivität. Erst die Beantwortung von Wer, Was, Wann und Warum schafft Evidenz.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Bei einem Audit fragt der Prüfer nach dem Zugriff eines Mitarbeiters auf das Kundenportal NordConnect. Statt nur ein System-Log mit Zeitstempel vorzulegen, kann Nordfeld zeigen: Sabine Krüger (Wer) hat den Zugriff auf Kundendaten der Region West (Was) befristet bis zum jährlichen Rezertifizierungstermin (Wann) aufgrund der Zuordnung des Mitarbeiters zum Vertriebsteam Region West (Warum) genehmigt.

Governance-Objekte und ihre Abgrenzung

Wer IGA nur als Verwaltung von Benutzerkonten versteht, greift zu kurz. Tatsächlich verwalten wir Governance-Objekte – Objekte, die zwei zentrale Merkmale teilen: Sie brauchen einen fachlich verantwortlichen Owner, und sie unterliegen einer regelmäßigen Prüfungspflicht.



Drei Kategorien lassen sich unterscheiden. **Identitäten** sind das Subjekt – Mitarbeitende, Externe oder Non-Person Identities. Die zentrale Frage lautet, wer fachlich dafür bürgt, dass diese Identität überhaupt existieren darf; ohne aktiven Identitäts-Owner fehlt die Grundlage für jeden Zugriff. **Berechtigungsstrukturen** definieren das Was – fachliche Rollen, Profile oder Gruppen. Ein Role Owner muss regelmäßig bestätigen, dass der Inhalt einer Rolle noch korrekt und für die aktuelle Aufgabe angemessen ist. **Steuerungselemente** schließlich betreffen das Wieso, Wo und Wie: Mandate und Strukturen (ist das Projekt noch aktiv, besteht der Vertrag noch, wer verantwortet den strukturellen Rahmen), Policies und Workflows (ist das Regelwerk noch zeitgemäß, entspricht der automatisierte Genehmigungsweg dem aktuellen Risiko) sowie Ausnahmen (ist die Abweichung vom Standard noch begründet und zeitlich befristet).

Governance scheitert häufig an lückenhafter Prüfung: Es reicht nicht, isoliert Berechtigungen zu bestätigen, während Mandate, Policies oder Ausnahmen ungeprüft bleiben. Ein verbreitetes Problem ist, dass fachlich Verantwortliche ihre Steuerungspflicht aus dem Blick verlieren, sobald diese in einem fremden IAM- oder IGA-System stattfindet – doch Technik entbindet nicht von Verantwortung. Identity Governance bedeutet, jedes Objekt, vom Mandat über die Policy bis zur Rolle, einem Verantwortlichen und einem Prüfzyklus zu unterwerfen.

Decision Governance als Steuerungsprinzip

Werden Identitäten, Rollen, Mandate und Policies als Governance-Objekte begriffen, stellt sich die zentrale Frage, wie sich das Ganze wirksam steuern lässt. Die Antwort: Decision Governance, getrieben durch den Business-Lifecycle. Es geht darum, von der Ticket-Logik des reaktiven Abarbeitens zu einer Entscheidungs-Logik des proaktiven Steuerns zu wechseln – jeder Zugriff soll das Ergebnis einer bewussten, regelbasierten Entscheidung sein, abgeleitet aus den Businessprozessen.

Vier Elemente machen Decision Governance im Kern aus. **Lifecycle-First** bedeutet, dass jede Entscheidung an ein Lebensphasen-Ereignis gekoppelt ist: Ein Zugriff entsteht, verändert sich oder endet, weil ein Businessprozess – Eintritt, Projektwechsel, Austritt – dies auslöst; der Lifecycle ist der Taktgeber. **Policy-Bindung** heißt, dass keine Entscheidung ohne Grundlage erfolgt; jede Gewährung referenziert ein Regelwerk, das die fachlichen Anforderungen widerspiegelt, sonst bleibt eine Entscheidung reine Willkür. **Zustandssteuerung statt Punktlandung** bedeutet, dass Decision Governance nicht mit dem Genehmigungsklick endet, sondern den Zustand überwacht – ändert sich der Business-Kontext, etwa Mandat, Standort oder Aufgabe, muss die Entscheidung neu bewertet werden. Die **Entscheidungskette** schließlich ist erst vollständig, wenn sie den gesamten Kreislauf durchläuft: Begründung, Umsetzung, Kontrolle, Korrektur.

Der eigentliche Hebel liegt darin, dass IGA damit unabhängig von manuellem Aufwand wird. Entscheidend ist nicht, wer ein Ticket bearbeitet, sondern wie Businessprozesse sicherstellen, dass fachlich Verantwortliche ihre Steuerungspflicht kontinuierlich wahrnehmen.

MERKSATZ

Governance ist die Summe aller qualifizierten Entscheidungen entlang des Business-Lifecycles.

Grenze des Modells: Die in diesem Kapitel beschriebene Rollentrennung – wer entscheidet, wer verantwortet, wer setzt um – setzt eine gewisse organisatorische Mindestgröße voraus. In sehr kleinen Organisationen oder frühen Start-up-Phasen trägt häufig eine einzelne Person mehrere dieser Rollen gleichzeitig. Das Denkmodell bleibt dabei gültig, aber die strikte Trennung wird durch Transparenz ersetzt: Wer

eine Entscheidung trifft, dokumentiert sie nachvollziehbar, auch wenn dieselbe Person sie später technisch umsetzt. Governance verschwindet nicht, sie verändert nur ihre Form.

2

Decision Governance

Welche Entscheidungen IGA wirklich steuert

Wird Decision Governance auf die Genehmigung eines Antrags reduziert, sehen wir nur die Spitze des Eisbergs. Eine fachlich geführte IGA steuert eine ganze Kaskade von Entscheidungen, die tief in die Businessprozesse hineinreicht – und diese Entscheidungstypen unterscheiden sich grundlegend voneinander.

Die **Existenzentscheidung** klärt, ob eine Identität in einem bestimmten Kontext überhaupt existieren darf – Onboarding auf Basis eines gültigen Vertrags, Mandats oder Projektkontexts. Die **Zuweisungsentscheidung** legt fest, welchen Handlungsspielraum eine Identität für ihre konkrete Aufgabe benötigt, immer im Abgleich mit dem Least-Privilege-Prinzip. Die **Ausnahmeentscheidung** ist die kritischste: Hier wird bewusst ein Risiko akzeptiert, geknüpft an Bedingungen, Kompensationsmaßnahmen und eine Befristung. Die **Fortsetzungsentscheidung** schließlich bestätigt, dass eine ursprüngliche Entscheidung noch valide ist – keine Formsache, sondern die bewusste Prüfung, ob Aufgabe, Risiko und Berechtigung weiterhin im Einklang stehen.

Wer all diese Typen als ein und dasselbe Ticket behandelt, verliert Steuerungsfähigkeit: Eine Ausnahmeentscheidung braucht andere Qualität, andere Verantwortung und andere Evidenz als eine standardisierte Zuweisung im Mover-Prozess.

IGA steuert nicht nur Zugriff, sondern die gesamte Kette der fachlichen Legitimation – vom Eintritt bis zum endgültigen Entzug.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Bei Nordfeld Logistik zeigen sich alle vier Typen im Alltag: Ein neuer Mitarbeiter wird angelegt (Existenzentscheidung), erhält die Rolle „Sachbearbeiter Vertrieb" (Zuweisungsentscheidung), Markus Feldt darf für sein Projekt ausnahmsweise auf ein zusätzliches Reporting-Tool zugreifen (Ausnahmeentscheidung), und einmal jährlich bestätigt Sabine Krüger, dass ihr Team die bestehenden Zugriffe noch braucht (Fortsetzungsentscheidung).

Decision Records als Governance-Artefakt

KURZ ERKLÄRT · DECISION RECORD

Ein Decision Record ist wie ein kurzer Aktenvermerk zu einer Zugriffsentscheidung: Wer hat entschieden, auf welcher Grundlage, für wie lange? Anders als ein Ticket, das nur zeigt, dass etwas gemacht wurde, zeigt der Decision Record, warum es erlaubt war.

In der Praxis herrscht oft der Glaube, ein Eintrag im ITSM-Tool sei bereits Dokumentation genug. Ein Ticket-Kommentar wie „Genehmigt per Mail von Müller" ist jedoch kein Governance-Nachweis, sondern lediglich eine Notiz. Für echte Decision Governance braucht es Decision Records – eigenständige Governance-Artefakte, die die fachliche Entscheidung vom technischen Bereitstellungsprozess trennen. Während ein Ticket dokumentiert, dass etwas getan wurde, dokumentiert der Decision Record, warum es fachlich legitim ist.

Ein vollwertiger Decision Record enthält vier Elemente: die **Entscheidungsgrundlage** (welche Policy, welches Rollenmodell wurde referenziert – keine Entscheidung ohne Regelbezug), den **Entscheidungskontext** (welches Mandat, Projekt oder welcher Business-Kontext war Auslöser), die **Verantwortlichkeit** (wer ist der fachlich accountable Owner) und die **zeitliche Komponente** (wie lange gilt die Entscheidung, welches Lifecycle-Ereignis triggert die nächste Prüfung).

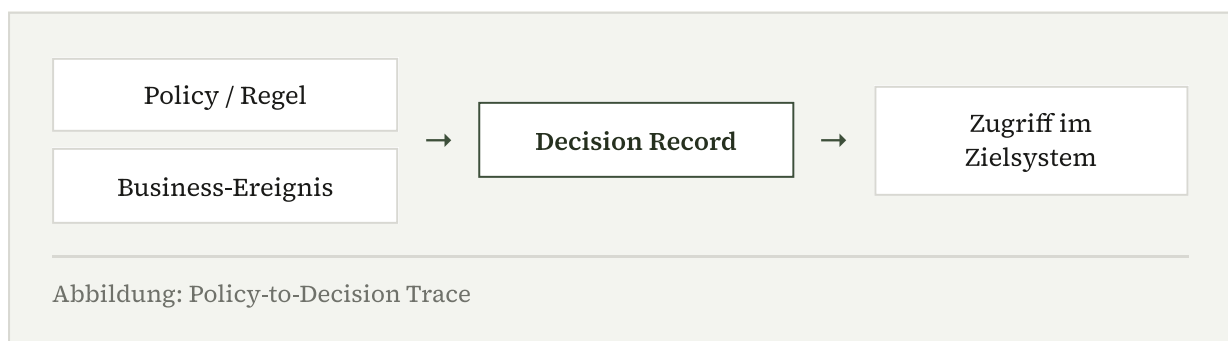
Tickets sind flüchtig und oft technisch überladen; ein Decision Record bleibt über den gesamten Lifecycle eines Zugriffs bestehen – er ist die Urkunde für die Berechtigung. Im Audit muss dann nicht durch tausende Tickets gesucht werden, sondern für jedes Governance-Objekt lässt sich der zugehörige Decision Record vorlegen.

MERKSATZ

Ein Ticket dokumentiert den Prozess. Ein Decision Record dokumentiert die Legitimation.

Policy-to-Decision Trace

Eine Policy darf kein totes Dokument im Intranet sein – in wirksamer Identity Governance ist sie der genetische Code jeder einzelnen Entscheidung. Die zentrale Frage lautet: Wie weisen wir nach, dass eine Entscheidung tatsächlich auf Basis der Policy getroffen wurde? Die Antwort ist der Policy-to-Decision Trace, die lückenlose Rückverfolgbarkeit von der technischen Berechtigung über die fachliche Entscheidung bis zur konkreten Regel in der Policy. Erst wenn diese Kette geschlossen ist, verlassen wir den Bereich der Willkür.



Ein belastbarer Trace braucht eine **explizite Referenz** – jeder Decision Record muss klar benennen, auf welche Policy, inklusive Abschnitt oder Regel, er sich stützt. Er braucht **Business-Logik als Bindeglied**: Sagt die Policy „Projektleiter erhalten Zugriff X“, muss der Trace belegen, dass die Person tatsächlich Projektleiter im aktiven Projekt Y ist. Er muss **Policy-Evolution und Versionierung** abbilden – jede Entscheidung ist ein Kind ihrer Zeit, und bei einer Regeländerung muss sichtbar sein, ob ein Zugriff noch unter der alten Version mit expliziter Bestätigung weiterläuft oder zwingend neu bewertet werden muss. Und moderne IGA-Systeme leisten **automatisierte Validierung**, die eine Abweichung vom aktuell gültigen Regelwerk sofort markiert.

MERKSATZ

Governance ist erst dann wirksam, wenn jede Entscheidung ihre Wurzeln in einer spezifischen Policy-Version nachweisen kann.

Ohne diesen Trace verlieren wir bei jeder Policy-Änderung die Kontrolle über den Bestand.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Die Policy von Nordfeld sagt: „Projektleiter erhalten Zugriff auf das Projekt-Budget-Tool.“ Markus Feldt ist Projektleiter für „Lagerausbau Nord“ – der Trace verknüpft seine Zugriffsentscheidung explizit mit dieser Policy-Regel. Wird die Policy später geändert, etwa weil künftig eine zusätzliche Freigabe nötig ist, zeigt der Trace sofort, welche bestehenden Zugriffe – wie der von Markus Feldt – nun erneut geprüft werden müssen.

Entscheidungskriterien statt Bedarfsaussagen

In vielen Organisationen läuft es so: Ein Fachbereich fordert Zugriff auf System X, ein Owner genehmigt das. Das ist eine Bedarfsaussage – subjektiv und punktuell. Für echte Decision Governance ist das keine tragfähige Grundlage. Wenn IGA skalierbar und auditfähig sein soll, müssen Fachbereiche lernen, in Entscheidungskriterien zu denken: objektiv, wiederverwendbar, direkt in eine Policy übersetzbar.

Der Wechsel führt weg von der Einzelfallprüfung: Definiert ein Owner einmal die Kriterien, unter denen ein Zugriff gewährt wird, muss er nicht mehr jeden Einzelfall neu bewerten – das System kann diese Kriterien konsistent anwenden. Solange Bedarfsaussagen akzeptiert werden, bleibt IGA ein manueller Ticket-Prozess. Erst Kriterien machen IGA zu einem echten Steuerungsinstrument für konsistente Zugriffszustände.

Governance bedeutet nicht, Einzelfälle zu genehmigen, sondern die Kriterien festzulegen, nach denen das System entscheidet.

Gültigkeit, Timeboxing und Review-Trigger

In der klassischen Administration galt oft: einmal berechtigt, immer berechtigt – bis zum Austritt. In moderner Decision Governance ist das ein erhebliches Sicherheitsrisiko, denn jede Entscheidung ist nur so lange valide, wie der Kontext existiert, in dem sie getroffen wurde. Um schleichenden Rechtezuwachs, sogenannten Privilege Creep, zu verhindern, brauchen Entscheidungen eine Haltbarkeit.

Drei Steuerungsinstrumente greifen dabei ineinander. Die **Gültigkeitsdauer** stellt sicher, dass keine Entscheidung unbefristet bleibt – bereits bei der Erteilung muss klar sein, für welchen Zeitraum die Legitimation gilt. **Timeboxing** setzt insbesondere bei Ausnahmen oder hochprivilegierten Zugriffen ein fixes Ablaufdatum, nach dessen Erreichen der Zugriff automatisch entzogen wird, sofern keine neu begründete Entscheidung vorliegt. **Review-Trigger** lösen eine sofortige Neubewertung aus, unabhängig vom Kalender: Business-Trigger wie ein vorzeitig endendes Projekt, Lifecycle-Trigger wie ein Abteilungs- oder Rollenwechsel, und Risk-Trigger wie eine Policy-Änderung, die den Bestand betrifft.

Ohne Befristung oder klare Trigger verwalten wir irgendwann nur noch historische Altlasten – der Entzug beim Austritt allein reicht nicht aus.

Eine Entscheidung ist kein statisches Denkmal, sondern ein temporärer Zustand, der proaktiv entlang des Business-Lifecycles gesteuert werden muss.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Markus Feldts Zugriff auf das Budget-Tool für „Lagerausbau Nord“ ist von Beginn an bis zum geplanten Projektende befristet (Timeboxing). Endet das Projekt vorzeitig, weil es gestoppt wird, löst das einen sofortigen Review-Trigger aus – unabhängig vom ursprünglich geplanten Ablaufdatum.

Entscheidungsqualität messbar machen

Wer IGA steuert, verfällt schnell in die KPI-Falle: gemessen werden geschlossene Tickets, Durchlaufzeiten oder die Anzahl von Rezertifizierungen. Diese Zahlen sagen jedoch nichts über Sicherheit oder Governance-Wirkung aus. In einer Decision Governance muss stattdessen die Qualität der Entscheidungen gemessen werden – wie gut eine Entscheidung dem tatsächlichen Business-Need und den definierten Policies entspricht.

Vier Metriken machen das sichtbar. Die **Policy-Adherence-Rate** zeigt, wie hoch der Anteil der Entscheidungen ist, die vollständig auf einer existierenden Policy basieren; eine niedrige Quote weist auf Bauchgefühl-Entscheidungen oder lückenhafte Regelwerke hin. Der **kriterienbasierte Automatisierungsgrad** zeigt, wie viele Entscheidungen auf Basis klar definierter Business-Attribute getroffen wurden. **Exception Ratio und Exception Debt** zeigen, wie viele Entscheidungen bewusste Abweichungen vom Standard sind und wie viele davon überfällig sind, weil ihr Timeboxing abgelaufen ist – steigende Exception Debt ist ein Frühwarnsignal für erodierende Governance. Und die **Evidenz-Vollständigkeit** zeigt, wie viele Entscheidungen über ein vollständiges Minimum Viable Evidence verfügen; eine Entscheidung ohne belastbaren Decision Record ist ein latentes Audit-Risiko.

Nur wer die Qualität seiner Entscheidungen kennt, kann belegen, dass Zugriffsrisiken wirklich unter Kontrolle sind.

MERKSATZ

Gute Governance misst nicht, wie viel gearbeitet wurde, sondern wie sicher und regelkonform die Zustände im System sind.

Anti-Pattern: Genehmigung ersetzt Entscheidung

Zum Abschluss dieses Kapitels lohnt der Blick auf eines der gefährlichsten Missverständnisse der Disziplin: die Verwechslung von technischer Genehmigung und fachlicher Entscheidung. In vielen Organisationen gilt ein Zugriff bereits als „governed“, sobald ein Workflow durchlaufen wurde und jemand auf „Genehmigen“ geklickt hat. Eine Genehmigung im Tool ist jedoch oft nur eine prozessuale Geste; eine Entscheidung ist ein qualifizierter fachlicher Akt.

Vier Mechanismen machen die bloße Genehmigung zum Risiko. Der **„Gefällt-mir“-Effekt**: Ein Workflow ohne Kontext verleitet zum Durchwinken – steht dort nur „Müller möchte Zugriff auf Ordner X“, fehlt die Entscheidungsgrundlage; es wird zugestimmt, nicht entschieden. **Technik ersetzt keine Verantwortung**: Governance findet im Kopf des Verantwortlichen statt, nicht im Code eines Workflows. **Fehlende Herleitung**: Eine Genehmigung ist binär, eine Entscheidung muss begründbar sein – „Dieser Zugriff ist erlaubt, weil Kriterium A erfüllt ist und Policy B ihn für Zeitraum C vorsieht“; fehlt diese Herleitung, bricht der Policy-to-Decision Trace. Und der **Blindflug im Audit**: Ein Zeitstempel einer Genehmigung hilft wenig, wenn niemand erklären kann, warum zugestimmt wurde.

Der Weg aus dem Anti-Pattern verlangt, Verantwortliche zu befähigen und zu verpflichten, echte Entscheidungen zu treffen. Das Tool darf nicht nur fragen „Darf er das?“, sondern muss zeigen: „Er darf das laut Policy X, wenn Bedingung Y erfüllt ist – bestätigst du das als accountable Owner?“

Ein Workflow ohne Entscheidungsgrundlage ist keine Steuerung, sondern nur eine automatisierte Form der Willkür.

Grenze des Modells: Decision Governance setzt voraus, dass sich Entscheidungskriterien tatsächlich vorab formulieren lassen. Bei hochgradig individuellen oder erstmaligen Bedarfen – etwa einem völlig neuen Aufgabenzuschnitt ohne Vorbild im Rollenmodell – gibt es noch keine Kriterien, auf die sich eine Entscheidung stützen könnte.

In solchen Fällen bleibt die Einzelfallprüfung durch einen Menschen unvermeidlich; das Ziel ist dann nicht, sie zu automatisieren, sondern sie sauber zu dokumentieren, damit aus dem Einzelfall bei Wiederholung ein Kriterium werden kann.

3

Ownership und Verantwortung

Fachliche vs. operative Verantwortung

Nachdem geklärt ist, wie Entscheidungen aussehen müssen, stellt sich die zentrale Frage: Wer steht am Ende dafür gerade? Ein verbreitetes Missverständnis in der IGA ist die Vermischung von fachlicher und operativer Verantwortung – oft hört man, das IAM-Team habe den Zugriff eingerichtet, sei also auch verantwortlich, wenn etwas falsch ist. Genau hier beginnt das Verantwortungs-Vakuum.

Zwei Rollen müssen klar getrennt werden. Die **fachliche Verantwortung (Accountability)** liegt meist im Business: Der Verantwortliche entscheidet über das Ob und das Was, kennt Geschäftsprozess, Risiko und Bedarf und bürgt dafür, dass der Zugriff legitim ist – im Audit muss er erklären können, warum diese Entscheidung richtig war. Die **operative Verantwortung (Responsibility)** liegt meist bei IT oder IAM: Sie kümmert sich um das Wie, stellt sicher, dass Workflows funktionieren und der Decision Record technisch korrekt erzeugt wird. Sie ermöglicht die Entscheidung, trifft sie aber nicht.

Wenn IT fachliche Risiken bewertet, fehlt ihr der Business-Kontext. Wenn das Business versucht, technische Details zu steuern, verliert es den Blick auf seine eigentliche Aufgabe: die kontinuierliche Steuerung des Zugriffs-Lifecycles.

MERKSATZ

Die IT verantwortet das Werkzeug, das Business verantwortet die Entscheidung. Wer das vermischt, schafft Verantwortungs-Vakuum.

KURZ ERKLÄRT · ACCOUNTABILITY VS. RESPONSIBILITY

Accountability heißt: Jemand steht am Ende dafür gerade, ob eine Entscheidung richtig war – auch wenn er sie nicht selbst technisch umgesetzt hat. Responsibility heißt: Jemand hat die Aufgabe erledigt. Ein Pilot ist accountable für einen sicheren Flug, auch wenn der Copilot gerade steuert.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Sabine Krüger, Bereichsleiterin Vertrieb, entscheidet, dass ein neuer Mitarbeiter Zugriff auf das Kundenportal NordConnect erhält – sie kennt den Bedarf und trägt die Verantwortung dafür. Tom Brenner aus dem IAM-Team richtet den Zugriff technisch ein. Geht später etwas schief, weil der Zugriff fachlich nicht gerechtfertigt war, steht Sabine Krüger dafür gerade – nicht Tom Brenner.

Application Owner, Role Owner, Data Owner

In vielen Organisationen ist „Owner“ nur ein Pflichtfeld in einer Datenbank. Damit Ownership tatsächlich Risiken reduziert, muss verstanden werden: Verantwortung findet auf unterschiedlichen Ebenen statt. Ein einzelner Zugriff wird idealerweise durch drei Instanzen legitimiert.

Der **Application Owner** trägt Verantwortung für die Anwendung als Ganzes: Er definiert, welche technischen Funktionen und Zugriffsmöglichkeiten existieren, und stellt die sichere Betriebsfähigkeit sicher – bei Systemstabilität oder technischen Schwachstellen ist er der Ansprechpartner. Der **Role Owner** bündelt Berechtigungen zu fachlich sinnvollen Rollen und entscheidet einmal für viele, welches Set an Rechten eine Rolle umfasst; er stellt sicher, dass die Kombination der Rechte logisch ist und keine internen Kontrollregeln verletzt. Der **Data Owner** verantwortet nicht die Software, sondern die Inhalte: Er legt fest, wie sensibel Daten sind und wer sie sehen oder verarbeiten darf, und muss im Ernstfall bewerten können, welche Auswirkungen ein unberechtigter Zugriff hätte.

MERKSATZ

Wer alles verantwortet, verantwortet am Ende nichts. Ownership braucht eine klare Domäne: System, Logik oder Inhalt.

Verschwimmen diese Rollen, entstehen blinde Flecken: Der Application Owner kennt nicht zwangsläufig die Sensitivität einzelner Datensätze, der Data Owner nicht jede technische Berechtigungsstruktur im Hintergrund. Erst im Zusammenspiel entsteht echte Steuerbarkeit – der Application Owner verantwortet das System, der Role Owner die Zugriffslogik, der Data Owner den Schutz der Inhalte.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Beim ERP-System LogiCore trägt ein IT-Kollege als Application Owner die Verantwortung dafür, dass das System stabil läuft. Yildiz Aksoy ist Role Owner der Rolle „Sachbearbeiter Vertrieb“ und legt fest, welche Rechte diese Rolle in LogiCore umfasst. Für die Kundendaten selbst ist wiederum ein Data Owner verantwortlich, der entscheidet, wer welche Kundendatensätze einsehen darf. Drei Personen, drei unterschiedliche Fragen – System, Logik, Inhalt.

Control Owner und Kontrollverantwortung

Application Owner, Role Owner und Data Owner steuern den Zugriff – doch wer stellt sicher, dass diese Rollen ihre Verantwortung auch tatsächlich wahrnehmen, dass Rezertifizierungen nicht zum bloßen Durchklicken werden? Hier kommt der Control Owner ins Spiel. Während andere Owner den Zugriff steuern, verantwortet er die Wirksamkeit der Kontrolle selbst – er prüft nicht einzelne User, sondern den Prüfprozess.

Vier Kernaufgaben gehören dazu: **Kontrolldesign**, also die Definition, wie eine Kontrolle fachlich und organisatorisch ausgestaltet sein muss, um ein konkretes Risiko zu reduzieren; die **Überwachung der Ausführung**, damit Kontrollen rechtzeitig und vollständig durchgeführt werden – bleibt eine Rezertifizierung aus, eskaliert er unabhängig davon, ob Business oder IT zuständig ist; die **Validierung der Kontrollgüte**, also die Prüfung, ob die Kontrolle ihren Zweck tatsächlich erfüllt und Abweichungen korrekt bewertet werden; und das **Remediation-Management**, das identifizierte Lücken bis zur Wiederherstellung des Soll-Zustands verfolgt.

Ohne einen klar benannten Control Owner werden Kontrollen schnell verwaist – IT verweist auf das Business, das Business auf die Revision.

Ein Owner verantwortet den Zustand. Ein Control Owner verantwortet die Sicherheit, dass dieser Zustand den Regeln entspricht.

RACI-Logiken für IGA

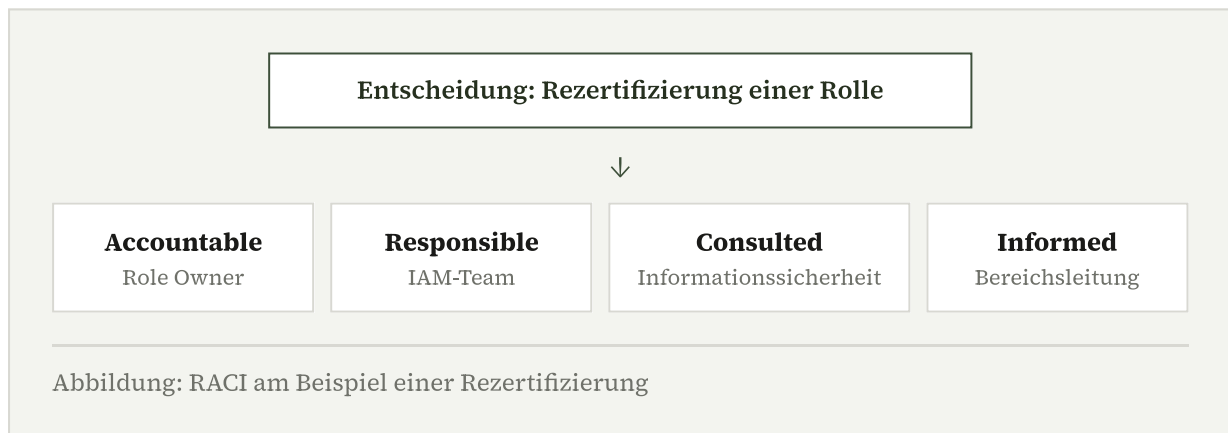
KURZ ERKLÄRT · RACI

RACI ist ein einfaches Ordnungsschema für Verantwortung in einem Prozess: Responsible (macht es), Accountable (steht dafür gerade), Consulted (wird gefragt), Informed (wird informiert). Es beantwortet die Frage „Wer macht was, und wer haftet dafür?“ für jeden einzelnen Schritt.

Ownership klingt klar, scheitert in der Praxis aber oft an einer simplen Frage: Wer macht es jetzt wirklich? Glauben mehrere Personen, sie seien nur „beratend“ beteiligt, bleibt Verantwortung liegen. Hier hilft eine klare RACI-Matrix – Responsible, Accountable, Consulted, Informed –, die zwischen Business und IT ein Verantwortungs-Vakuum verhindert.

Pro Entscheidung darf es genau eine Person geben, die **Accountable** ist, häufig der Role oder Application Owner; sie trägt die fachliche Verantwortung für das Ergebnis und muss es im Audit erklären können. Das **Responsible** ist für die operative Umsetzung zuständig, meist IAM oder IT, und kann Aufgaben delegieren – das Accountable delegiert niemals die Verantwortung. **Consulted** sind Fachexperten, deren Einschätzung vor einer Entscheidung eingeholt werden muss, etwa Informationssicherheit oder Datenschutz; sie tragen zur Qualität bei, treffen aber nicht die finale Entscheidung. **Informed** sind Personen, die über das Ergebnis informiert werden, etwa der betroffene User oder dessen Führungskraft – hier geht es um Transparenz, nicht um Mitwirkung.

Häufig wird die IT als Accountable eingetragen, weil sie das System bedient – für die Systemverfügbarkeit mag das stimmen, für die fachliche Legitimation eines Zugriffs jedoch nicht. Erst wenn klar ist, dass Business Accountable und IT Responsible ist, wird Governance belastbar.



Ohne RACI bleibt Ownership eine Absichtserklärung. Erst die Matrix klärt, wer handelt und wer Rechenschaft trägt.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Bei der jährlichen Rezertifizierung der Rolle „Sachbearbeiter Vertrieb“ ist Yildiz Aksoy als Role Owner Accountable – sie muss im Audit erklären können, warum die Rolle so aussieht. Tom Brenner aus dem IAM-Team ist Responsible und führt den technischen Rezertifizierungs-Workflow durch. Die Informationssicherheit wird als Consulted einbezogen, weil die Rolle sensible Kundendaten betrifft. Sabine Krüger als Bereichsleiterin wird am Ende nur noch Informed – sie muss nichts tun, aber über das Ergebnis Bescheid wissen.

Delegation und Stellvertretung

„Ich habe keine Zeit für 500 Rezertifizierungen im Monat“ ist ein berechtigter Einwand aus dem Business – die naheliegende Lösung ist Delegation. Doch in der Identity Governance ist Delegation ein zweischneidiges Schwert: Wer Aufgaben überträgt, ohne klare Leitplanken zu definieren, riskiert Kontrollverlust. Zu unterscheiden sind operative Delegation und formale Stellvertretung.

Vier Spielregeln sichern das ab. **Accountability bleibt beim Owner**: Man kann die Durchführung delegieren, niemals aber die Rechenschaftspflicht – trifft ein Delegierter eine Fehlentscheidung, steht weiterhin der Owner im Audit-Bericht. Die **formale Stellvertretung** muss klar geregelt und im System dokumentiert sein; der Stellvertreter braucht die fachliche Kompetenz, um Entscheidungen qualifiziert zu treffen. **Transparenz im Decision Record** heißt, dass eine delegiert getroffene Entscheidung dies auch abbildet: „Entscheidung durch Person X im Auftrag von Owner Y“ – ohne

diese Transparenz verliert die Entscheidung im Audit an Belastbarkeit. Und die **Überwachungspflicht bei Delegation** verlangt, dass ein Owner regelmäßig prüft, ob im Sinne der Policy entschieden wird; blindes Vertrauen ersetzt keine Governance.

Delegation ist notwendig, um IGA skalierbar zu machen – sie darf aber nicht dazu führen, dass Verantwortung verdünnt wird. Ein gutes Delegationsmodell verteilt Last auf qualifizierte Schultern, ein schlechtes verteilt nur Verantwortungslosigkeit.

Delegiere die Aufgabe, aber niemals das Risiko. Wer delegiert, muss die Leitplanken für die Entscheidung mitliefern.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Sabine Krüger ist im Urlaub, als eine dringende Rezertifizierung ansteht. Sie benennt einen Kollegen als formalen Stellvertreter, dokumentiert im Decision Record „Entscheidung durch Stellvertreter im Auftrag von Sabine Krüger“ und prüft nach ihrer Rückkehr stichprobenartig, ob im Sinne der Policy entschieden wurde. Trifft der Stellvertreter eine Fehlentscheidung, bleibt Sabine Krüger dennoch Accountable.

Grenzen von Verantwortung

Wer Verantwortung trägt, muss auch wissen, wo sie endet. Ohne klare Demarkationslinien wird ein Owner schnell für systemische Fehler verantwortlich gemacht, die außerhalb seines Einflussbereichs liegen. Governance bedeutet nicht unbegrenzte Verantwortung, sondern klar definierte Zuständigkeiten.

Vier Grenzen sind dabei typisch. Die **Informationsgrenze**: Ein Role Owner verantwortet die inhaltliche Stimmigkeit einer Rolle, nicht aber, wenn ein User sie aufgrund falsch gepflegter HR-Attribute im Quellsystem erhält – wo die Qualität der Zulieferdaten endet, endet auch sein Einfluss. Die **systemische Grenze**: Rechte, die über globale Gruppen wie „Alle Mitarbeitenden Standort X“ vererbt werden, kann ein lokaler Application Owner nicht steuern; hier liegt die Verantwortung bei der zentralen Governance-Instanz. Die **Kontrollgrenze (Segregation of Duties)**: Ein Owner darf nicht allein für einen Zustand verantwortlich sein, der organisationsübergreifende Kontrollregeln verletzt – betreffen Risiken mehrere Domänen, muss Accountability geteilt oder eskaliert werden. Und die **zeitliche Grenze**: Verantwortung gilt für

den Zeitraum der aktiven Entscheidung; ist ein Zugriff sauber befristet, endet die fachliche Verantwortung mit Ablauf dieser Frist – wird der Entzug technisch nicht umgesetzt, liegt das Problem in der operativen Schicht, nicht beim fachlichen Owner.

Klare Grenzen erhöhen Akzeptanz und schaffen Fairness im System. Ein Owner muss für das geradestehen, was er steuern kann – nicht für das, was außerhalb seines Einflusses liegt.

Verantwortung ohne Einflussmöglichkeit ist keine Governance, sondern strukturelles Risiko. Klare Grenzen schützen den Owner und das System.

Anti-Pattern: Verantwortung ohne Ownership

Zum Abschluss dieses Kapitels lohnt der Blick auf eine organisatorische Falle, die jede Governance-Initiative untergräbt: Verantwortung ohne die dazugehörige Entscheidungsgewalt. In vielen Unternehmen werden Personen für die Sicherheit von Systemen oder Daten verantwortlich gemacht, während die tatsächliche Steuerungsmacht über Rollen, Kriterien oder Konfigurationen an anderer Stelle liegt. Das ist nicht nur unfair – es ist ein strukturelles Risiko.

Vier Ausprägungen zeigen, wie gefährlich das ist. Der **IT-Sündenbock**: Häufig wird die IT-Administration für unberechtigte Zugriffe verantwortlich gemacht, obwohl sie das fachliche Risiko nicht bewerten kann – Verantwortung ist dann von der Entscheidung entkoppelt. Das „**Blind-Signer**“-**Syndrom**: Manager unterschreiben Rezertifizierungen, ohne die Logik hinter Rollen oder Kriterien beeinflussen zu können; sie tragen formale Verantwortung ohne Ownership über die Entscheidungsgrundlage – das Ergebnis ist formaler Klickbetrieb statt echter Steuerung. Die **Erosion der Motivation**: Niemand übernimmt gerne Verantwortung für eine Black Box; wer Risiken verwalten, aber nicht aktiv steuern darf, verliert die Bereitschaft zur Sorgfalt. Und das **Audit-Vakuum**: Bricht die Frage „Warum wurde das so entschieden?“ auf die Antwort „Ich musste unterschreiben, hatte aber keinen Einfluss auf die Kriterien“ herunter, ist die Governance-Logik nicht belastbar.

Die Lösung ist klar: Accountability und Decision Power müssen zusammengehören. Wer für ein Risiko Rechenschaft trägt, muss auch die Kriterien und die Policy-Referenz steuern können.

Verantwortung ohne Ownership ist keine Governance, sondern organisierte Schuldzuweisung.

MERKSATZ

Governance funktioniert nur, wenn die Person, die Rechenschaft trägt, auch die Person ist, die steuert.

Grenze des Modells: Die hier beschriebene Ownership-Logik setzt stabile, benennbare Rollen voraus – einen Application Owner, einen Role Owner, jemanden, der über längere Zeit für ein Objekt geradesteht. In agil organisierten Umgebungen mit rotierenden Teams oder häufig wechselnden Produktverantwortlichkeiten ist genau diese Stabilität nicht immer gegeben. Das Modell bricht dadurch nicht zusammen, aber es verlangt zusätzlichen Aufwand: Ownership muss dann explizit bei jedem Rollenwechsel neu zugewiesen und dokumentiert werden, statt implizit an eine feste Position gekoppelt zu sein.

Eine strukturelle Lösung dafür koppelt Owner-Rollen nicht an eine Person, sondern automatisiert an das Struktur-Objekt selbst (Team, Produkt, Organisationseinheit): Wird die Struktur angelegt, entstehen Owner- und Genehmiger-Rolle automatisch mit; ändert sich die Leitung, aktualisiert sich die Mitgliedschaft automatisch; löst sich die Struktur auf, verschwinden die Rollen oder wandern zur Nachfolgestruktur. Die leitende Person einer Struktur ist dabei zwingendes Pflichtmitglied der zugehörigen Owner-Rolle – eine Owner-Rolle kann so nie leer werden, solange die Führungsposition besetzt ist. Das verschiebt Ownership-Stabilität von einer organisatorischen Hoffnung zu einer technisch erzwungenen Eigenschaft.

4

Lifecycle und Drift

Lifecycle als Zustandslogik

Oft wird der Identity Lifecycle als Abfolge von Ereignissen betrachtet: Joiner, Mover, Leaver. Dieses Event-Denken führt in der Praxis zu Problemen, weil darauf vertraut wird, dass ein Impuls eine Kette von Aktionen korrekt auslöst. Echte Governance denkt anders: Lifecycle ist keine Ereigniskette, sondern eine Zustandslogik. Die entscheidende Frage lautet nicht „Was müssen wir tun, weil sich etwas geändert hat?“, sondern „Welchen Status hat die Identität aktuell – und passen die Berechtigungen zu diesem Status?“

Eine zustandsbasierte IGA behandelt jede Identität als Objekt in einem definierten Status – aktiv, in Elternzeit, externer Dienstleister, ausgetreten –, an den Berechtigungen gekoppelt sind; ändert sich der Status, ergibt sich unmittelbar der neue Soll-Zustand. Sie verabschiedet sich vom „Fire and Forget“: Statt einen Zugriff zu vergeben und nicht mehr zu hinterfragen, wird kontinuierlich geprüft, ob die Bedingung dafür noch erfüllt ist. Sie verhindert Berechtigungs-Drift, weil es in der Zustandslogik kein „Mitnehmen“ alter Berechtigungen gibt – der aktuelle Zustand bestimmt vollständig den gültigen Zugriff. Und sie steuert über Attribute wie Jobcode, Standort oder Kostenstelle als zentrale Steuerungsgrößen, sodass manuelle Nachpflege zur Ausnahme wird.

Wer in Ereignissen denkt, reagiert auf Fehler. Wer in Zuständen denkt, schafft konsistente Zugriffszustände als Normalfall.



MERKSATZ

Identitätsmanagement ist kein Domino-Effekt, sondern eine dauerhafte Synchronisation. Der Zugriff folgt dem Status, nicht dem Ticket.

KURZ ERKLÄRT · DESIRED STATE VS. ACTUAL STATE

Der Desired State ist das, was laut Regelwerk gelten sollte – etwa „nur aktive Mitarbeiter haben Zugriff“. Der Actual State ist das, was tatsächlich im System steht. Governance sorgt dafür, dass beide möglichst deckungsgleich bleiben.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Wechselt eine Mitarbeiterin von Nordfeld Logistik in Elternzeit, ändert sich ihr Status im HR-System. Die Zustandslogik erkennt das automatisch und setzt ihre LogiCore-Zugriffe auf „ruhend“ – ohne dass jemand manuell ein Ticket stellen muss.

Joiner als Governance-Startpunkt

Der erste Tag eines neuen Mitarbeiters wird oft als rein administrativer Akt gesehen: Hardware ausgeben, Account anlegen, fertig. Aus Sicht der Identity Governance ist der Joiner-Prozess jedoch der Startpunkt aller Zugriffsentscheidungen – was hier falsch aufgesetzt wird, zieht sich durch den gesamten Lifecycle.

Der Joiner-Prozess ist die **Geburtsstunde der Legitimation**: Bevor ein Zugriff technisch umgesetzt wird, muss die fachliche Entscheidung stehen, warum diese Person Zugriff erhält – der Prozess übersetzt den Arbeitsvertrag in eine steuerbare Identität. Er folgt dem **„Golden Record“-Prinzip**: Governance beginnt bei der Datenquelle, meist HR – wer Daten in der IT manuell korrigiert, verliert sofort die Steuerbarkeit. Er arbeitet mit **Birthrights statt Copy-Paste**: „Gib ihm einfach die Rechte von Kollege Müller“ kopiert nicht nur notwendige Zugriffe, sondern auch Altlasten und Fehlberechtigungen – ein sauberer Prozess definiert stattdessen den minimal notwendigen Zugriff für eine Rolle. Und er etabliert vom ersten Tag an **Accountability**, indem die Identität mit einem Owner oder Vorgesetzten verknüpft wird.

Ein Joiner-Prozess, der nur auf Geschwindigkeit optimiert ist, produziert langfristig Risiken. Der Joiner-Prozess ist die Qualitätskontrolle am Eingang. Was hier ungeprüft durchrutscht, wird später zum teuren Audit-Finding.

Fein granular betrachtet ist es dabei nicht die Identität selbst, sondern der einzelne **Vertrag**, der Rollen und Zugriffe legitimiert – ein arbeits- oder dienstleistungsbezogenes Verhältnis zwischen Identität und Geschäftseinheit. Diese Unterscheidung wird wichtig, sobald eine Identität mehr als einen Status kennt: Ein binäres aktiv/inaktiv auf Identitätsebene bildet Zwischenzustände wie Elternzeit oder Sabbatical nur unsauber ab – entweder ganz ohne Sonderbehandlung, oder faktisch wie ein vollständiger Leaver. Modelliert man stattdessen den Vertrag selbst mit einem eigenen Zustand wie „pausiert“, lassen sich dynamische Rollen befristet entziehen, während richtlinienkonforme Basiszugänge erhalten bleiben. Hat eine Identität mehrere gleichzeitige Verträge zu unterschiedlichen Einheiten, braucht es zusätzlich eine klare Priorisierung, etwa über einen als führend markierten Primärvertrag – sonst konkurrieren mehrere Legitimationsgrundlagen unklar miteinander.

Mover als Risikoereignis

Ein interner Wechsel ist oft ein Grund zum Feiern – aus Sicht der Identity Governance beginnt hier jedoch die kritischste Phase im gesamten Lifecycle. Während beim Joiner auf der grünen Wiese gestartet wird, bringt ein Mover eine digitale Vergangenheit mit. Liegt der Fokus nur auf Geschwindigkeit, entsteht fast zwangsläufig Berechtigungs-Wildwuchs (Entitlement Creep) – kein technisches, sondern ein Thema der Verantwortungssteuerung.

Vier Bruchstellen sind kritisch: die **Entlastungspflicht des bisherigen Verantwortlichen**, der aktiv sicherstellen muss, dass Zugriffe auf seine Domäne entzogen werden – „nicht mehr mein Mitarbeiter“ beendet nicht automatisch die Verantwortung; die **Sorgfaltspflicht des neuen Verantwortlichen**, der nicht ungeprüft übernehmen darf, was der Mitarbeiter mitbringt, sondern nur die für die neue Rolle notwendigen Zugriffe legitimiert; die **Rolle der Role Owner**, die das Zielbild definieren, welche Berechtigungen zur neuen Funktion gehören; und das **Risiko von SoD-Konflikten**, wenn alte und neue Rechte in Kombination kritische Konflikte erzeugen.

Ein sauberer Mover-Prozess erzwingt einen klaren Schnitt: Alte, funktionsgebundene Rechte werden entzogen, neue Zugriffe ergeben sich aus der Zielrolle, alles darüber hinaus erfordert eine neue, begründete Entscheidung.

Dieser klare Schnitt bleibt der Standardweg – er ist aber nicht immer sofort umsetzbar, ohne laufende Aufgaben abzureißen. Wo fachliche Kontinuität nötig ist, lässt sich dafür ein kontrollierter Abweichungspfad vorsehen, keine gleichwertige Alternative: eine befristete Nachlaufrist, in der alte und neue Führungskraft gemeinsam vereinbaren, dass einzelne Altzugriffe zeitlich begrenzt bestehen bleiben, während parallel die neuen Zugriffe beantragt werden. Wie jede Ausnahme aus Kapitel 5 braucht das einen benannten Owner, ein festes Ablaufdatum und – weil während der Nachlaufrist alte und neue Rechte gleichzeitig aktiv sind – eine verschärfte SoD-Prüfung, die genau diese Kombination explizit bewertet, nicht nur die neuen Rechte isoliert.

MERKSATZ

Ein interner Wechsel ist kein Domino-Effekt, sondern ein Neustart der Legitimationskette.

Ein Mover braucht den Entzug von Altlasten, bevor neue Privilegien entstehen.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Ein Mitarbeiter wechselt vom Vertrieb in die Logistik-Abteilung. Sabine Krüger als bisherige Vorgesetzte muss aktiv seine Vertriebs-Zugriffe in LogiCore entziehen lassen – „ist ja nicht mehr mein Mitarbeiter“ reicht nicht. Der neue Vorgesetzte in der Logistik legitimiert nur die Zugriffe, die für die neue Rolle wirklich gebraucht werden, statt die alten Vertriebsrechte einfach mitlaufen zu lassen.

Leaver und strukturierter Entzug

Während beim Mover zu viele Rechte das Risiko sind, gilt für den Leaver das Gegenteil: Er darf nichts mehr dürfen. In dem Moment, in dem ein Mitarbeiter das Unternehmen verlässt, wird seine Identität zum kritischsten Risiko im gesamten Lifecycle – ein aktiver Account ohne zugehörige Person ist eine offene Angriffsfläche. Der Leaver-Prozess ist damit der finale Härtetest jeder Governance.

HR als Auslöser: Sobald das Austrittsdatum feststeht, muss der Status im führenden System aktualisiert und unmittelbar in die IGA überführt werden – jede Verzögerung zwischen physischem Austritt und technischer Sperrung schafft unnötiges Risiko. **Verantwortung des Managers:** Der Vorgesetzte muss den Übergang aktiv steuern – welche Daten gesichert, wer Aufgaben und Zugriffe übernimmt, Delegationen vor dem Austritt geklärt sein müssen. Der Umgang mit „Grace Periods“ verlangt besondere Vorsicht: „Lass den Account noch zwei Wochen offen“ ist eine häufige Praxis und ein erhebliches Risiko – jede Ausnahme von der sofortigen Sperrung benötigt klares Timeboxing und eine dokumentierte Entscheidung. Und der **Lizenz- und Ressourcenrückbau** betrifft nicht nur Sicherheit, sondern auch Effizienz, da nicht entfernte Zugriffe weiterhin Lizenzen und Ressourcen binden.

Sobald der Status auf „ausgetreten“ wechselt, greift die Zustandslogik: Der Soll-Zustand ist eindeutig – kein Zugriff. Es gibt kein „Vergessen“, weil der Zustand kontinuierlich überprüft wird.

Ein sauberer Austritt ist kein administrativer Vorgang, sondern der finale Nachweis funktionierender Kontrolle. Wer den Entzug nicht beherrscht, hat den Lifecycle nie wirklich gesteuert.

Reorganisationen als Zugriffstrigger

Nach dem individuellen Mover stellt sich dieselbe Herausforderung in größerem Maßstab: Reorganisationen. Ob Zusammenlegung von Abteilungen, Ausgliederung von Bereichen oder neue Hierarchien – für Identity Governance ist das kein IT-Event, sondern ein fachlicher Massen-Trigger. Was beim einzelnen Mover noch manuell gesteuert werden kann, bricht bei hunderten oder tausenden Betroffenen zusammen.

Vier Anforderungen sind zentral: **Abhängigkeit vom HR-Trigger** – Strukturänderungen müssen im führenden System sauber abgebildet sein, manuelle Listen durchbrechen die Kette der Legitimation. **Massenhafte Zustandsänderung ohne Altlasten** – die Zustandslogik muss sicherstellen, dass alte Zugriffe entfallen und neue automatisch entstehen, statt „mitzuwandern“. **Verantwortung der Role Owner**, die vor der Umstellung definieren müssen, welche Berechtigungen zur neuen Struktur gehören – eine Reorganisation zeigt unmittelbar, wie belastbar das Rollenmodell wirklich ist. Und **zentrale Legitimation statt Einzelentscheidungen**: Ein zentraler Decision Record für die Reorganisation dokumentiert die fachliche Entscheidung, aus der die Anpassungen systemisch umgesetzt werden.

Reorganisationen sind keine IT-Migrationen, sondern fachliche Strukturentscheidungen. Wer die Legitimationskette systemisch mitführt, behält die Kontrolle. Wer sie manuell anpasst, verliert sie.

Projekt- und Mandatsende

Neben dem Leaver gibt es eine leisere, aber ebenso kritische Situation: das Ende von Projekten, Sonderaufgaben oder befristeten Mandaten. Der Mitarbeiter bleibt – aber die fachliche Legitimation für einen Teil seiner Zugriffe entfällt. Diese Zugriffe werden häufig vergessen; zurück bleiben digitale Altlasten, die das Least-Privilege-Prinzip untergraben und Berechtigungs-Drift fördern.

Vier Herausforderungen prägen diese Situation. Im Projektkontext entsteht neben der Linienverantwortung eine **temporäre fachliche Ownership**: Der Projektleiter legitimiert den Zugriff und ist damit auch für dessen Beendigung verantwortlich. Projektzugriffe brauchen eine **Trennung von Mandat und Linie**: Sie dürfen kein Teil der regulären Rollen sein, sondern müssen als zusätzliche, befristete Berechtigungen geführt werden. **Timeboxing als primärer Mechanismus** ist unerlässlich, weil manuelle Meldungen über Projektenden unzuverlässig sind – jeder Zugriff im Projektkontext

braucht ein klares Ablaufdatum. Und **Rezertifizierung als Absicherung** greift bei längeren Laufzeiten, wenn der Projektverantwortliche die fortbestehende Erforderlichkeit bestätigt.

Ein Projekt ist ein eigener Lifecycle innerhalb des Mitarbeiter-Lifecycles. Sobald es endet, muss die Zustandslogik greifen und die zugehörigen Berechtigungen entfallen.

Ein Projektende ist ein Teil-Offboarding. Wer nur den Austritt aus dem Unternehmen kontrolliert, lässt zu, dass sich über Jahre unnötige Projektrechte ansammeln.

Drift: Desired vs. Actual State

Nach Joiner, Mover und Projektende folgt das operative Kernproblem der Governance: Berechtigungs-Drift. In einer idealen Welt entspricht der Ist-Zustand exakt dem Soll-Zustand. In der Realität driften beide auseinander – Drift ist die Lücke zwischen dem, was sein sollte, und dem, was tatsächlich ist.

Zwei Zustände treffen aufeinander: der **Desired State (Soll)**, bestehend aus Birth-rights, Rollenlogik und begründeten Entscheidungen, und der **Actual State (Ist)**, die tatsächlich vorhandenen Berechtigungen in den Zielsystemen – das, was effektiv wirksam ist, unabhängig davon, wie es entstanden ist. Drift entsteht selten bewusst, sondern im Alltag: durch direkte Änderungen außerhalb des Governance-Prozesses, nicht entfernte Zugriffe nach Rollenwechseln oder Projektenden, oder Fehler in Provisionierungsprozessen. Das Risiko dabei: Drift untergräbt jede Policy – sagt der Soll-Zustand „Lesen“ und der Ist-Zustand erlaubt „Admin“, entsteht ein Risiko, das in keinem Regelwerk sichtbar ist.

Die Governance-Lösung liegt im kontinuierlichen Abgleich: eine **detektive Kontrolle**, die Abweichungen erkennt, und eine **korrektive Kontrolle**, die den Zugriff entweder automatisch anpasst oder bewusst neu legitimiert. Abweichungen dürfen nicht dauerhaft bestehen bleiben – Drift ist kein Ausnahmefall, sondern ein Signal, dass Steuerung nicht durchgängig greift. IGA ist kein einmaliger Aufbau, sondern ein kontinuierlicher Abgleich gegen die natürliche Entropie von Systemen.

Der Standardfall dieses Abgleichs ist Top-down: Das Governance-Regelwerk ist die führende Quelle, der Actual State wird an sie angeglichen. Für einzelne Zielsysteme ist das nicht immer sofort möglich – etwa wenn kein verlässliches Regelwerk für das System vorliegt oder die Schnittstelle instabil ist. In diesen Fällen kann für dieses

eine System bewusst Bottom-up abgeglichen werden, also die im Zielsystem tatsächlich vergebenen Rechte zunächst ins Governance-System übernommen werden. Das ist keine gleichwertige Designentscheidung, sondern eine dokumentierte Governance-Lücke: Sie muss als solche gekennzeichnet sein und mit dem Ziel geführt werden, zurück auf Top-down zu wechseln, sobald das Zielsystem dafür bereit ist.

Wer seinen Drift nicht misst, hat keine Kontrolle.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Laut Rollenmodell (Desired State) darf „Sachbearbeiter Vertrieb“ nur Kundendaten lesen. Ein Administrator hat aber vor Monaten für einen Sonderfall Schreibrechte direkt im System vergeben und vergessen, sie wieder zu entfernen – der Actual State weicht damit vom Soll ab. Ohne regelmäßigen Abgleich bleibt diese Abweichung unentdeckt, bis ein Audit sie aufdeckt.

Anti-Pattern: Zugriff als Dauerzustand

Über den kompletten Lifecycle hinweg – vom sauberen Start beim Joiner über die Gefahren beim Mover und Projektende bis zum Umgang mit Drift – zeigt sich ein zentrales Missverständnis im Identity Management: Zugriff wird oft wie ein dauerhaftes Eigentum behandelt, nach dem impliziten Modell „Einmal genehmigt, immer gültig“. Dieses Denken untergräbt jede Form von Governance.

Drei Probleme zeigen, warum: Die **Illusion der statischen Welt** ignoriert, dass sich Organisationen, Rollen und Risiken permanent weiterentwickeln – unbefristete Zugriffe entkoppeln Berechtigungen vom tatsächlichen Bedarf, der Ist-Zustand entfernt sich zwangsläufig vom Soll-Zustand. Die **Anhäufung von Berechtigungen** wächst über Jahre an, wenn Zugriffe nicht entzogen werden, und unterläuft systematisch das Least-Privilege-Prinzip. Und der **Verlust von Verantwortung** zeigt sich, wenn niemand mehr erklären kann, wer einen vor Jahren vergebenen Zugriff verantwortet, weil Personen Rollen gewechselt oder das Unternehmen verlassen haben.

Das Gegenmodell versteht Zugriff als temporäre Delegation: **Timeboxing als Standard**, mit definiertem Ablaufdatum; **bewusste Erneuerung**, bei der nach Ablauf der Entzug erfolgt, sofern die Notwendigkeit nicht neu begründet und dokumentiert wird; und **zustandsbasierte Steuerung**, bei der sich ändernder Kontext automatisch zur Anpassung oder zum Entzug führt.

MERKSATZ

Zugriff ist kein dauerhaftes Eigentum, sondern eine temporäre, zweckgebundene Delegation.

Wer Zugriff nicht befristet, überlässt die Kontrolle der Zeit statt der Governance.

Grenze des Modells: Die gesamte Zustandslogik dieses Kapitels setzt ein verlässliches, zeitnah gepflegtes „Golden Record“-System voraus – meist HR. In Übergangsphasen wie Fusionen, Carve-outs oder bei kleineren Organisationen ohne dediziertes HR-System ist genau diese Datenquelle oft lückenhaft oder inkonsistent. Die Zustandslogik funktioniert dann nur so gut wie ihre schwächste Datenquelle: Governance-Tooling kann fehlende oder falsche Quelldaten nicht kompensieren, es macht ihre Folgen nur schneller sichtbar.

5

Ausnahmen und Sonderfälle

Ausnahme als Governance-Objekt

Nach dem regulären Lifecycle und dem Soll-Ist-Abgleich stellt sich die Realität: Abweichungen vom Standard sind unvermeidlich. Es wird immer zeitkritische Projekte, Sonderaufgaben oder technische Notwendigkeiten geben, die nicht in das Standard-Rollenmodell passen. Das Problem ist nicht die Ausnahme selbst – sondern ihr informeller Umgang. Wird eine Ausnahme „auf Zuruf“ oder direkt im System umgesetzt, entsteht Intransparenz und Kontrollverlust.

Die Governance-Lösung macht die Ausnahme selbst zum Governance-Objekt: Sie darf nicht informell existieren, sondern muss als eigenständiges Objekt mit klarer Zuordnung und Dokumentation geführt werden. Jede Ausnahme braucht einen fachlichen Owner, eine nachvollziehbare Begründung und eine klare zeitliche Begrenzung. Und sie darf nicht zum Dauerzustand werden – sie muss regelmäßig überprüft und aktiv gesteuert werden, genau wie jede andere Berechtigung.

Ziel ist es, Ausnahmen aus der Grauzone der manuellen Administration zu holen und sie als kontrollierten, auditierbaren Sonderpfad in die Governance zu integrieren.

MERKSATZ

Eine Ausnahme ist kein Governance-Versagen, sondern ein bewusst zu steuerndes Ereignis.

KURZ ERKLÄRT · EXCEPTION RECORD

Eine formale Notiz dazu, warum jemand ausnahmsweise mehr oder etwas anderes darf als der Standard vorsieht – mit Owner, Begründung und Ablaufdatum. Ohne diese Notiz ist die Ausnahme unsichtbar für die Governance.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Für das Projekt „Lagerausbau Nord“ braucht Markus Feldt kurzfristig Zugriff auf ein Reporting-Tool, das eigentlich nur der Geschäftsleitung vorbehalten ist. Statt den Zugriff informell einzurichten, wird eine Ausnahme mit Owner (Markus Feldt), Begründung (Projektsteuerung) und Ablaufdatum (Projektende) dokumentiert.

Exception Records: Das dokumentierte Ausnahme-Objekt

Damit eine Ausnahme steuerbar wird, braucht sie eine saubere Form: den Exception Record. Er ist mehr als eine Genehmigung – er ist das Governance-Objekt, das eine Abweichung nachvollziehbar und überprüfbar macht.

Fünf Elemente muss ein Exception Record enthalten: **Kontext und Begründung**, keine pauschalen Aussagen wie „Projektbedarf“, sondern klare, fachlich begründete Kriterien; den **konkreten Zugriff**, eindeutig beschrieben, welche Berechtigungen vom Standard abweichen; **klare Gültigkeit** mit festem Ablaufdatum, denn ohne Zeitbezug wird aus einer Ausnahme ein Dauerzustand; **fachliche Verantwortung**, getragen von einem klar benannten Application oder Data Owner; und die **Einbindung in die Kontrolllogik**, da Abweichungen vom Standard zusätzliche Aufmerksamkeit erfordern.

Ohne Exception Records bleiben Ausnahmen unsichtbar – sie existieren dann außerhalb der Governance und entziehen sich jeder Kontrolle.

Jede Ausnahme braucht einen Owner und ein Enddatum. Wer Ausnahmen nicht formalisiert, überlässt ihre Wirkung dem Zufall.

Kompensationskontrollen: Der Sicherheitsmechanismus für Ausnahmen

Ein Exception Record dokumentiert die Abweichung vom Standard – doch Dokumentation allein reduziert kein Risiko. Jede Ausnahme erzeugt eine Kontrolllücke, die aktiv geschlossen werden muss. Hier kommen Kompensationskontrollen ins Spiel: temporäre Maßnahmen, die sicherstellen, dass trotz Abweichung ein vergleichbares Sicherheitsniveau erreicht wird.

Eine wirksame Kompensationskontrolle braucht eine **risikobasierte Ausrichtung**, die gezielt das durch die Ausnahme entstehende Risiko adressiert – wird etwa eine Aufgabentrennung aufgehoben, muss die Nutzung besonders überwacht werden. Sie erfordert **höhere Prüfintensität**, da Ausnahmen engmaschigere Kontrollen brauchen als im Standardfall. Sie verlangt **aktives Monitoring**, damit die Nutzung von Ausnahmezugriffen nachvollziehbar und beobachtbar ist und relevante Ereignisse an verantwortliche Stellen wie den Control Owner weitergeleitet werden. Und sie ist untrennbar mit dem Lebenszyklus der Ausnahme verbunden: Greift die Kontrolle nicht mehr oder wird sie nicht bestätigt, muss auch die Ausnahme entzogen werden.

Erst durch Kompensationskontrollen wird eine Ausnahme nachvollziehbar und für Audit und Revision bewertbar.

Eine Ausnahme ohne Kompensationskontrolle ist kein gesteuerter Sonderfall, sondern ein bewusst akzeptiertes Risiko.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Weil Markus Feldts Ausnahmezugriff auf das Geschäftsleitungs-Reporting ein erhöhtes Risiko darstellt, wird zusätzlich vereinbart: Seine Nutzung wird monatlich statt jährlich überprüft, und jede Anmeldung wird protokolliert und an den Control Owner gemeldet.

Timeboxing von Ausnahmen: Das Ablaufdatum als Pflichtfeld

In der Identity Governance ist eine Ausnahme per Definition ein temporärer Zustand. Wird sie unbefristet gewährt, entwickelt sie sich faktisch zu einer neuen Regel – Timeboxing stellt sicher, dass genau das nicht passiert.

Vier Gründe machen ein zwingendes Ablaufdatum notwendig: die **Vermeidung von Exception Debt**, da sich ohne definiertes Ende Ausnahmen über die Zeit ansammeln und eine zunehmend unübersichtliche Berechtigungslandschaft entsteht; die **erzwungene Neubewertung**, denn ein Ablaufdatum ist kein reiner Abschaltpunkt, sondern ein bewusster Prüfimpuls – besteht der Bedarf weiterhin, muss die Ausnahme aktiv verlängert werden, inklusive neuer Begründung; die **Grundlage für automatisierten Entzug**, damit eine nicht verlängerte Ausnahme systemisch endet, ohne manuelles Eingreifen; und die **Begrenzung des Risikofensters**, da jede Ausnahme das Risiko erhöht und ein klares Zeitfenster die mögliche Exposition reduziert.

Ein Ausnahmeantrag ohne Enddatum darf nicht existieren. Timeboxing ist eine der einfachsten und wirkungsvollsten präventiven Kontrollen.

Eine Ausnahme ohne Ablaufdatum ist keine Ausnahme, sondern eine dauerhaft etablierte Abweichung vom Regelwerk.

Exception Debt als Steuerungsproblem

KURZ ERKLÄRT · EXCEPTION DEBT

Ähnlich wie „technische Schulden“ in der Softwareentwicklung: Jede Ausnahme, die länger lebt als nötig, ist eine kleine Schuld gegenüber der eigenen Governance. Viele solcher Schulden zusammen machen das System unübersichtlich und riskant.

Formalisieren, dokumentieren, absichern und befristen – all das hat ein Ziel: Ausnahmen kontrollierbar zu halten. Doch was passiert, wenn Ausnahmen immer wieder verlängert werden, ohne echte Neubewertung? Dann entsteht Exception Debt – die Summe aller Ausnahmen, die ihren ursprünglichen Zweck verloren haben, aber weiterhin bestehen.

Vier Effekte machen das zum zentralen Governance-Problem: der **Verlust der Transparenz**, da mit zunehmender Anzahl der Überblick schwindet und der Soll-Zustand zur Fiktion wird, während der Ist-Zustand dominiert; das **erhöhte Sicherheitsrisiko**, da nicht mehr benötigte Ausnahmen aktiv bleiben, die Angriffsfläche erweitern und Szenarien wie Lateral Movement begünstigen; das **Compliance-Risiko**,

weil unbegründete oder veraltete Zugriffe im Audit schwer zu rechtfertigen sind; und die **operative Belastung**, da mit steigender Anzahl der Prüf- und Verlängerungsaufwand wächst.

Gesteuert wird Exception Debt über drei Hebel: **Ablauf bedeutet Entzug** als Standardfall, nicht die Fortführung; **KPI-basierte Steuerung**, die sichtbar macht, wie viele Ausnahmen überfällig oder mehrfach verlängert wurden – nur messbare Schulden lassen sich aktiv steuern; und **geplanter Rückbau**, da jede Ausnahme von Beginn an eine Exit-Strategie braucht.

MERKSATZ

Exception Debt ist keine technische, sondern eine Governance-Schuld.

Wer Ausnahmen anhäuft, verliert schrittweise die Kontrolle über seine Zugriffslandschaft.

Rückbau von Sonderfällen: Der Weg zurück zum Standard

Um die Schulden aus Exception Debt abzubauen, reicht ein Ablaufdatum allein nicht aus – jede Ausnahme braucht einen aktiven Rückbaupfad. Drei Strategien greifen dabei ineinander. Die **Standardisierung** prüft, ob eine regelmäßig verlängerte Ausnahme möglicherweise kein Sonderfall mehr ist und ins Rollenmodell oder die Policy überführt werden sollte – aus dem Sonderfall wird ein sauber definierter Standard. Der **automatisierte Entzug** stellt sicher, dass eine Ausnahme, die ihr Enddatum ohne Verlängerung erreicht, tatsächlich entzogen wird, statt bestehen zu bleiben, weil sie vergessen wurde. Und die **Verbesserung der Standardprozesse** erkennt in gehäuften Ausnahmen oft ein Signal, den Standard selbst zu überarbeiten – Ziel ist, den Bedarf direkt im Regelwerk abzubilden statt über Umwege.

Fehlt dieser Pfad, entsteht das, was sich als „Unmanaged Process“ bezeichnen lässt: eine Ausnahme ohne definierten Lebenszyklus, deren ursprünglicher Bedarf nie erneut hinterfragt wird. Solche Fälle summieren sich über die Zeit – aus einer einzelnen, gedachten Ausnahme wird eine unübersichtliche Struktur, in der der Soll-Zustand an Bedeutung verliert und der Ist-Zustand faktisch übernimmt. Anders als bei

einer aktiv gesteuerten Ausnahme trägt hier niemand die Verantwortung für den Abschluss – das ist der eigentliche Unterschied zum kontrollierten Sonderfall, nicht die Ausnahme selbst.

Ein Sonderfall ohne Rückbauplan wird früher oder später zu einem strukturellen Problem für Compliance und Steuerbarkeit.

MERKSATZ

Ein Sonderfall ohne Abbaupfad ist kein temporäres Problem, sondern eine dauerhafte Risikoerhöhung.

Grenze des Modells: Die formale Exception-Record-Logik dieses Kapitels setzt voraus, dass genug Zeit bleibt, eine Ausnahme vor ihrer Nutzung zu dokumentieren. Bei echten Notfällen – ein System muss sofort repariert werden, ein Sicherheitsvorfall erfordert unmittelbares Handeln – ist dieser Vorlauf nicht gegeben. Für solche Fälle ist die reguläre Ausnahme-Logik nicht das richtige Werkzeug; dafür braucht es einen eigenen, nachgelagerten Pfad, den Kapitel 7 unter dem Stichwort Break-Glass behandelt.

6

Zugriffsmodelle

Rollenmodelle und ihre Grenzen

Bevor einzelne Modelle wie RBAC oder ABAC betrachtet werden, lohnt der Blick auf die eigentliche Aufgabe eines Rollenmodells: Es übersetzt organisatorische Komplexität in eine steuerbare Struktur – die Brücke zwischen „Wer bist du?“ und „Was darfst du tun?“

Rollenmodelle existieren aus drei Gründen. Sie bündeln statt einzeln zu steuern: Statt einzelner Berechtigungen pro Person werden Zugriffe in fachlich sinnvolle Pakete zusammengefasst. Sie schaffen fachliche Verständlichkeit: Ein Verantwortlicher versteht die Rolle „Einkauf“, technische Einzelberechtigungen dagegen selten. Und sie sind Grundlage für Automatisierung, da nur standardisierte Strukturen konsistente Joiner-, Mover- und Leaver-Prozesse ermöglichen.

Genau hier beginnen aber auch die Grenzen. **Rollenexplosion** entsteht, wenn für jede Ausnahme eine eigene Rolle gebaut wird – die Struktur wird dann so komplex wie die Realität selbst. **Fehlender Kontext** zeigt sich, weil klassische Rollenmodelle statisch sind und nicht berücksichtigen, wann, wo oder unter welchen Bedingungen ein Zugriff genutzt wird. Und **Wartungsschuld** entsteht, weil Rollenmodelle altern – ohne kontinuierliche Pflege entstehen über Jahre unklare oder überladene Rollenstrukturen.

Ein Rollenmodell ist kein Selbstzweck. Es soll Komplexität reduzieren – nicht reproduzieren.

MERKSATZ

Ein Modell ist die Landkarte, nicht das Gelände.

Es muss präzise genug sein, um zu führen – und flexibel genug, um der Realität standzuhalten.

KURZ ERKLÄRT · ROLLENMODELL

Ein Rollenmodell bündelt einzelne Zugriffsrechte zu einem fachlich benannten Paket – statt zehn Einzelrechte zu vergeben, wird einmal die Rolle „Vertriebsmitarbeiter“ zugewiesen, die alle notwendigen Rechte automatisch mitbringt.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Statt jedem neuen Vertriebsmitarbeiter einzeln Zugriff auf Kundendaten, Angebots-Tool und Reporting zu geben, existiert bei Nordfeld Logistik die Rolle „Sachbearbeiter Vertrieb“ in LogiCore. Wird jemand neu eingestellt, erhält er diese eine Rolle – und damit automatisch alle passenden Rechte. Bekäme jede Sonderaufgabe eine eigene Zusatzrolle, würde die Rollenliste jedoch schnell unübersichtlich – genau das ist die Rollenexplosion.

RBAC fachlich eingeordnet

Role-Based Access Control (RBAC) ist in der Identity Governance weit mehr als das Zuweisen von Gruppen – es ist das strategische Werkzeug, um die Kluft zwischen fachlicher Organisation und technischer Realität zu überbrücken. Eine professionelle RBAC-Implementierung trennt zwischen fachlicher und technischer Ebene, wobei je nach Organisationsstruktur und Zielarchitektur auch deutlich mehr Ebenen möglich sind. Typisch sind **Business Roles**, die die funktionale Identität eines Nutzers abbilden (etwa „Projektleiter“ oder „Personalreferent“), für Fachverantwortliche lesbar sind und die Basis für Governance und Rezertifizierung bilden, sowie **IT-Roles**, die die konkreten technischen Berechtigungen in den Zielsystemen bündeln und die technische Komplexität hinter einer fachlich verständlichen Struktur kapseln.

RBAC bleibt aus drei Gründen das Fundament vieler IGA-Modelle: **Standardisierung**, weil gleiche Aufgaben zu gleichen Zugriffen führen und individuelle Sonderlösungen reduziert werden; **steuerbare Funktionstrennung (SoD)**, weil sich kritische Kombinationen auf Rollenebene deutlich besser kontrollieren lassen als auf Ebene einzelner Berechtigungen; und **Auditierbarkeit**, weil fachliche Rollen nachvollziehbare Legitimationen schaffen – im Fokus steht die fachliche Funktion, nicht das einzelne technische Recht.

MERKSATZ

RBAC ist die gemeinsame Sprache zwischen Business und IT – solange nicht jede Ausnahme zu einem eigenen Dialekt wird.

Strategisch eingeordnet ist RBAC die Lösung für das 80/20-Prinzip: Es deckt die stabilen, wiederkehrenden Standardfälle hervorragend ab. Die Grenzen entstehen dort, wo hochdynamische oder kontextabhängige Anforderungen beginnen – wird versucht, auch die letzten Sonderfälle vollständig in starre Rollen zu pressen, entsteht die bekannte Rollenexplosion. RBAC sollte deshalb nicht als vollständige Lösung verstanden werden, sondern als stabiles Fundament, das durch andere Modelle ergänzt werden kann.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Die Business Role „Sachbearbeiter Vertrieb" ist für Sabine Krüger als Bereichsleiterin verständlich – sie weiß, was diese Rolle bedeutet. Dahinter steckt eine IT-Rolle, die technisch definiert, dass diese Rolle Lesezugriff auf bestimmte LogiCore-Tabellen und Schreibrechte im Angebots-Modul umfasst. Sabine muss diese technischen Details nie sehen – sie entscheidet auf Ebene der Business Role.

ABAC und Attributqualität

Während RBAC vor allem beantwortet, welche Rolle eine Person hat, stellt Attribute-Based Access Control (ABAC) eine andere Frage: Welche Eigenschaften liegen aktuell vor – und unter welchem Kontext erfolgt der Zugriff? ABAC ergänzt klassische Rollenmodelle um dynamische Entscheidungslogik: Entscheidungen basieren nicht nur auf einer Rolle, sondern auf Attributen – Eigenschaften von Personen, Objekten oder der Umgebung. Typische Beispiele sind Subjekt-Attribute wie Abteilung, Standort oder Sicherheitsfreigabe, Objekt-Attribute wie Vertraulichkeitsstufe oder Eigentümer eines Dokuments, und Umwelt-Attribute wie Uhrzeit, Netzwerkzone oder aktuelles Risiko-Level.

Der strategische Vorteil liegt darin, dass sich hochdynamische Anforderungen besser steuern lassen: Statt für jede Sonderkonstellation eine eigene Rolle zu bauen – etwa „Eigene Rolle für Land A" –, lässt sich eine Regel definieren wie „Zugriff erlaubt, wenn User.Country == Object.Country". Genau hier entsteht aber auch die größte Herausforderung: ABAC ist nur so gut wie die Qualität seiner Attribute. Werden Daten veraltet, widersprüchlich oder uneinheitlich gepflegt, entstehen falsche Entscheidungen – „Finance" und „Finanzwesen" dürfen fachlich nicht dasselbe bedeuten, technisch aber unterschiedlich behandelt werden.

Dabei lohnt eine Unterscheidung, die in der Praxis oft übersehen wird: die Pflege eines Attributwerts (etwa die korrekte Region eines Mitarbeiters im HR-System) ist etwas anderes als die Verantwortung für die Attributdefinition selbst – wer legt fest, welche Werte für "Region" überhaupt gültig sind, und wer entscheidet, wenn ein neues Attribut wie "Sicherheitsfreigabe" eingeführt wird? Diese Verantwortung lässt sich nicht automatisch einer der drei Owner-Rollen aus Kapitel 3 zuordnen – sie kann je nach Attribut beim Data Owner liegen (bei inhaltlich sensiblen Attributen) oder eine eigene, vierte Owner-Rolle erfordern, wenn ein Attribut mehrere Systeme und An-

wendungsfälle gleichzeitig betrifft. Ohne einen klar benannten Attribute Owner für diese strukturelle Ebene bleibt ABAC formal steuerbar, aber inhaltlich beliebig – jedes Team pflegt seine Werte nach eigenem Verständnis.

ABAC verschiebt den Schwerpunkt von statischen Rollen hin zur Qualität und Steuerbarkeit von Daten. Damit wird Attribut-Governance zu einem zentralen Bestandteil moderner Identity Governance.

MERKSATZ

ABAC bietet enorme Flexibilität – aber nur, wenn die zugrunde liegenden Daten konsistent, aktuell und kontrolliert sind.

KURZ ERKLÄRT · ABAC

Statt starrer Rollen prüft ABAC bei jedem Zugriff automatisch Eigenschaften: Wer bist du, was willst du sehen, unter welchen Umständen? Das funktioniert wie eine Regel „Wenn A und B zutreffen, dann ist Zugriff erlaubt“ – flexibel, aber nur so gut wie die zugrunde liegenden Daten.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Im Kundenportal NordConnect soll ein Vertriebsmitarbeiter nur Kunden aus seiner eigenen Region sehen. Statt für jede Region eine eigene Rolle anzulegen, definiert Nordfeld eine Regel: „Zugriff erlaubt, wenn Region des Mitarbeiters gleich Region des Kunden ist.“ Trägt jedoch jemand die Region im HR-System als „Nord“ und im CRM als „Norden“ ein, funktioniert die Regel nicht mehr zuverlässig – ein typisches Attributqualitäts-Problem.

ReBAC und die Kraft der Beziehungen

Während RBAC auf Rollen und ABAC auf Eigenschaften basiert, stellt Relationship-Based Access Control (ReBAC) die Beziehung zwischen Identitäten und Ressourcen in den Mittelpunkt – die zentrale Frage lautet: In welcher Beziehung stehst du zu diesem Objekt? Berechtigungen entstehen damit nicht primär über starre Rollen oder einzelne Attribute, sondern über Verknüpfungen innerhalb eines Netzwerks.

Drei Formen sind typisch. Die **hierarchische Vererbung**: Wer Eigentümer eines Bereichs ist, erhält automatisch Zugriff auf die darunterliegenden Objekte, ohne dass Rechte für jedes einzelne Element separat vergeben werden müssen. **Transitive Beziehungen**: Besitzt ein Mitarbeiter Zugriff auf ein Projekt, kann sein verantwortlicher Manager bestimmte Rechte automatisch mit erhalten. Und **Kollaboration & Ownership**: Zugriffe entstehen direkt aus der Zusammenarbeit heraus – wer Inhalte erstellt, wird „Owner“, wer beteiligt wird, erhält definierte Bearbeitungsrechte.

ReBAC eignet sich besonders für moderne Kollaborationsplattformen und komplexe Organisationsstrukturen, da sich Zugriffe stärker an realen Arbeitsbeziehungen orientieren und sich dynamisch mit Teams, Projekten und Datenstrukturen verändern. Dadurch reduziert sich der Bedarf an manueller Einzelsteuerung erheblich.

Diese Flexibilität hat aber eine Grenze, die mit der Größe der Organisation wächst: Je mehr Beziehungsebenen und transitive Vererbungen ein Berechtigungsmodell zulässt, desto schwerer wird die einfache Frage "Wer hat eigentlich Zugriff auf X?" zu beantworten. Was als direkte Rolle in einer Liste steht, muss in einem Beziehungsgraphen erst über mehrere Kanten nachverfolgt werden – für einen Reviewer oder Auditor ist das ungleich aufwendiger als eine Rollenliste. ReBAC verlagert Komplexität von der Verwaltung hin zur Nachvollziehbarkeit, sie verschwindet nicht.

Nicht nur Rollen oder Attribute bestimmen den Zugriff, sondern die fachliche Beziehung zwischen Identität und Ressource.

KURZ ERKLÄRT · REBAC

ReBAC fragt nicht „Welche Rolle hast du?“, sondern „In welcher Beziehung stehst du zu dieser Sache?“ Wer ein Dokument erstellt hat, darf es bearbeiten. Wer zu einem Team gehört, sieht die Team-Dateien. Der Zugriff ergibt sich aus der Verbindung, nicht aus einem vorab definierten Paket.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Markus Feldt legt für das Projekt „Lagerausbau Nord“ einen gemeinsamen Projektordner an. Jeder, der dem Projektteam hinzugefügt wird, erhält automatisch Zugriff auf diesen Ordner – nicht, weil er eine bestimmte Rolle hat, sondern weil er Teil des Teams ist. Verlässt jemand das Projekt, entfällt der Zugriff automatisch mit der Teammitgliedschaft.

Scope- und Task-basierte Modelle

Neben Rollen, Attributen und Beziehungen gibt es Zugriffsszenarien, in denen nicht die Identität selbst im Mittelpunkt steht, sondern die konkrete Aufgabe oder der fachliche Geltungsbereich. Hier setzen Scope- und Task-basierte Modelle an: Der Fokus verschiebt sich von „Wer bist du dauerhaft?“ hin zu „Was musst du aktuell tun – und in welchem Kontext?“

Task-Based Access Control (TBAC) vergibt Berechtigungen für die Dauer einer konkreten Aufgabe; ist der Prozessschritt abgeschlossen, wird auch der Zugriff wieder entzogen – der Zugriff folgt damit direkt dem Geschäftsprozess. **Scope-basierte Zugriffsteuerung** begrenzt den Zugriff auf einen definierten Geltungsbereich, etwa eine Region, ein Projekt oder ein Mandat – eine Person darf beispielsweise auf Kundendaten zugreifen, aber nur innerhalb des Scopes „Region West“.

Diese Modelle unterstützen das Least-Privilege-Prinzip besonders konsequent: Statt dauerhafte Berechtigungen bereitzustellen, werden Zugriffe an konkrete Aufgaben und klar definierte Kontexte gekoppelt. Dadurch entstehen weniger stehende Berechtigungen und die Angriffsfläche wird reduziert – Berechtigungen entstehen aus einem fachlichen Bedarf heraus und sollten enden, sobald dieser Bedarf entfällt.

Zugriff sollte kein Dauerzustand sein, sondern eine zeitlich und fachlich begrenzte Notwendigkeit.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Für das befristete Projekt „Lagerausbau Nord“ erhält Markus Feldt Zugriff auf ein Budget-Tool – aber nur mit dem Scope „Standort Nord“ und nur für die Projektlaufzeit. Nach Projektende entfällt der Zugriff automatisch, unabhängig davon, ob Markus Feldt danach ein neues Projekt übernimmt.

Modellkombinationen in der Praxis

Rollen, Attribute, Beziehungen und kontextbezogene Zugriffe wurden bislang einzeln betrachtet – in der Praxis moderner Organisationen reicht jedoch selten ein einzelnes Modell aus. Governance entsteht häufig erst durch die Kombination mehrerer Ansätze, wobei jedes Modell die Aufgaben übernimmt, für die es am besten geeignet ist.



Ein typisches Hybridmodell nutzt **RBAC als Fundament** für Standardzugriffe, was Struktur schafft und administrativen Aufwand reduziert; **ABAC als Feinsteuerung**, das die Rolle um dynamische Regeln ergänzt, etwa Zugriff nur bei gültiger Zertifizierung oder bestimmtem Standort; **ReBAC für Kollaboration**, das projekt- oder teambezogene Zugriffe steuert – wer Teil eines Projektteams ist, erhält Zugriff auf die gemeinsame Arbeitsumgebung; und **Task- und Scope-Modelle für kritische Aktionen**, die besonders sensible Aktivitäten zusätzlich an konkrete Aufgaben oder Geltungsbereiche koppeln.

Durch die Kombination lassen sich die Schwächen einzelner Modelle ausgleichen: RBAC reduziert Standardaufwand, ABAC vermeidet unnötige Sonderrollen, ReBAC unterstützt dynamische Zusammenarbeit, Task- und Scope-Modelle begrenzen hochkritische Zugriffe. So entsteht ein Modell, das stabil genug für Governance und gleichzeitig flexibel genug für moderne Arbeitsweisen ist.

Zugriffsmodelle sollten nicht isoliert betrachtet werden. Entscheidend ist, wie gut sie gemeinsam dabei helfen, Zugriffe nachvollziehbar, steuerbar und kontrollierbar zu machen.

Wie bei Decision Governance (Kapitel 2) lässt sich auch die Qualität eines Zugriffsmodells messen, statt sie nur zu behaupten. Eine einfache, aber aussagekräftige Kennzahl ist die durchschnittliche Anzahl an Rollen pro Nutzer: Steigt sie über die Zeit kontinuierlich, ohne dass sich die Aufgabenvielfalt der Organisation entsprechend verändert hat, deutet das auf Rollenexplosion oder mangelnde Pflege hin. Ergänzend zeigt der Anteil ungenutzter Rollen oder Attribute, wie viel vom Modell tatsächlich lebt und wie viel nur historisch mitgeschleppt wird.

Rollen geben die Grundstruktur vor. Attribute, Beziehungen und Kontext definieren die Grenzen und Bedingungen des Zugriffs.

Anti-Pattern: Modellwechsel ohne Governance

Die Reise durch Rollen, Attribute, Beziehungen und kontextbezogene Zugriffe zeigt, wie flexibel moderne Zugriffsmodelle geworden sind. Genau hier entsteht aber ein häufiges Missverständnis: Ein neues Modell löst keine alten Governance-Probleme – Technik heilt keine Prozessfehler. Oft wird ein Modellwechsel als schnelle Lösung betrachtet, um starre Rollenstrukturen oder operative Probleme zu umgehen. Ohne ein belastbares Governance-Fundament entsteht jedoch lediglich neue Komplexität.

Drei Gründe zeigen, warum Governance vor dem Modell kommen muss. Die **Verlässlichkeit der Datenquellen**: Steuern Attribute oder Beziehungen den Zugriff, muss klar geregelt sein, wer für Qualität und Aktualität dieser Daten verantwortlich ist – falsche Daten führen unmittelbar zu falschen Entscheidungen. **Klare Verantwortlichkeiten**: Ein flexibles Modell ohne definierte Ownership-Strukturen ist nicht steuerbar – wer definiert Regeln, wer überprüft Ausnahmen, wer trägt die Accountability? Und

Nachvollziehbarkeit und Auditierbarkeit: Je dynamischer ein Modell wird, desto wichtiger wird die Fähigkeit, Entscheidungen nachvollziehbar zu machen – ohne klare Vorgaben für den Policy-to-Decision Trace wird jede Prüfung schwierig.

MERKSATZ

Technik folgt der Governance. Ein Modellwechsel ohne klare Prozesse verschiebt das Chaos lediglich an eine andere Stelle.

Ein Modellwechsel ist kein reines IT-Projekt. Er verändert Entscheidungslogiken, Verantwortlichkeiten und Kontrollmechanismen. Die Wahl eines Zugriffsmodells sollte deshalb immer Ergebnis einer Governance-Strategie sein – nicht der Versuch, fehlende Governance technisch zu kompensieren.

7

Privileged Access

Privileged Access als eigener Kontrollraum

Muss man für administrative Rechte andere Fragen stellen als für den Zugriff auf ein Fachsystem? Absolut. Privilegierter Zugriff wird in der Praxis oft technisch betrachtet – Passwort-Tresore, Admin-Konten, Root-Rechte. Aus Governance-Sicht greift das zu kurz: Ein privilegierter Zugriff ist nicht einfach ein normaler Account mit mehr Berechtigungen, er verändert das Risikoprofil einer Organisation grundlegend und erfordert einen eigenen Kontrollraum mit deutlich strengeren Anforderungen.

Dazu gehört eine **höhere Begründungstiefe**: Ein einfaches „wird benötigt“ reicht nicht aus, jede Aktivierung sollte an einen konkreten Zweck, ein Ereignis oder eine definierte Aufgabe gekoppelt sein. Es gehört eine **höhere Nachweisdichte** dazu: Während bei Standardzugriffen oft die Frage im Vordergrund steht, wer einen Zugriff besitzt, kommt bei privilegierten Zugriffen zwingend hinzu, was mit diesem Zugriff tatsächlich getan wurde. Und es braucht **Verfall statt Standing Access**: Dauerhaft aktive privilegierte Berechtigungen gehören zu den größten Governance-Risiken – sie sollten nur so lange bestehen, wie sie tatsächlich benötigt werden.

MERKSATZ

Privilegierter Zugriff ist kein technisches Privileg, sondern ein kritisches Governance-Objekt mit besonders hohen Anforderungen an Steuerung und Nachweis.

Standardzugriffe und privilegierte Zugriffe folgen denselben Grundprinzipien – Legitimation, Verantwortung, Kontrolle, Nachweis –, im privilegierten Umfeld werden diese Anforderungen jedoch deutlich verschärft. Wenn Standard-IGA das Sichern der

Haustür ist, dann ist Privileged Access Governance die Kontrolle des Tresorraums.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Jonas Vogel aus dem IT-Team hat administrative Rechte auf LogiCore. Anders als ein normaler Sachbearbeiter-Zugriff reicht bei ihm nicht die Frage „hat er die Rolle?“ – es muss auch protokolliert werden, was er mit diesen Rechten konkret getan hat, und die Rechte sollten nicht dauerhaft aktiv sein, sondern nur bei tatsächlichem Bedarf.

Standard- vs. privilegierter Zugriff

Macht es für die Governance einen Unterschied, ob ein Mitarbeiter Stammdaten pflegt oder die zugrundeliegende Datenbank konfiguriert? Ja, einen fundamentalen. In der Praxis wird die Grenze oft technisch gezogen – Root-Rechte gelten als privilegiert, Arbeit im Fachverfahren als Standard. Aus Governance-Sicht greift diese rein technische Trennung zu kurz: Die Unterscheidung basiert nicht auf Systemgrenzen, sondern auf dem inhärenten Risiko und den potenziellen Auswirkungen einer Fehlbedienung oder Kompromittierung.

Daraus ergeben sich zwei Kontrollwelten. **Standard-Zugriff (Business Access)** dient der Durchführung regulärer Geschäftsprozesse innerhalb definierter Leitplanken, mit Fokus auf dauerhafter, rollenbasierter Bereitstellung für den täglichen Bedarf. **Privilegierter Zugriff (Privileged Access)** ermöglicht die Veränderung, Erweiterung oder Umgehung bestehender Kontrollmechanismen – das kann technische Infrastruktur betreffen, ebenso aber fachliche Funktionen mit besonders hohem Schadenspotenzial. Entscheidend ist nicht die Technologie, sondern die Wirkungsmacht des Zugriffs.

Das größte Risiko entsteht in der Grauzone: Werden hochkritische Berechtigungen wie normale Standardzugriffe behandelt, fehlen die erforderlichen Kontrollen, Nachweise und Überwachungsmechanismen. Umgekehrt dürfen privilegierte Zugriffe niemals außerhalb fachlicher Verantwortlichkeiten existieren. Standard-Zugriffe bewegen sich innerhalb definierter Leitplanken, privilegierte Zugriffe besitzen die Möglichkeit, diese Leitplanken zu verändern, zu erweitern oder außer Kraft zu setzen.

Nicht die Technologie macht einen Zugriff privilegiert, sondern sein Potenzial, Kontrolle auszuüben oder Kontrollen zu verändern.

Eligibility vs. Activation

Reicht es aus zu wissen, wer administrative Rechte besitzen darf, um ein System abzusichern? Nein – das ist erst die halbe Wahrheit. In der klassischen IGA ist ein zugewiesenes Recht meist sofort und dauerhaft aktiv. Im privilegierten Umfeld ist genau dieses Modell eines der größten Governance- und Sicherheitsrisiken, denn zwischen der grundsätzlichen Berechtigung und der tatsächlichen Nutzung besteht ein entscheidender Unterschied.

Moderne Privileged Access Governance trennt deshalb zwei Zustände. **Eligibility (Berechtigung)** definiert, welche Identität unter bestimmten Bedingungen überhaupt privilegierte Aktionen ausführen darf – fachliche Legitimation, erforderliche Freigaben und Verantwortlichkeiten werden im Vorfeld geprüft; die Identität ist berechtigt, besitzt aber im Normalzustand keine aktiven Privilegien. **Activation (Aktivierung)** ist der bewusste und nachvollziehbare Übergang von einer potenziellen Berechtigung zu einem tatsächlich nutzbaren privilegierten Zugriff – jede Aktivierung erfolgt für einen konkreten Zweck, innerhalb eines definierten Zeitfensters und mit vollständiger Nachvollziehbarkeit.

Die entscheidende Frage lautet nicht „Wer darf privilegierte Zugriffe erhalten?“, sondern „Wer hat wann, warum und für welchen Zweck privilegierte Zugriffe tatsächlich genutzt?“ Eligibility schafft den sicheren Rahmen, Activation liefert den konkreten Kontext für Kontrolle, Auditierbarkeit und Accountability. Ein Administrator sollte seine Privilegien nicht wie einen Schlüsselbund permanent in der Tasche tragen, sondern sie nur dann gezielt aktivieren, wenn ein Schloss geöffnet werden muss.



MERKSATZ

Eligibility beschreibt das Potenzial. Activation dokumentiert die tatsächliche Nutzung.

Erst die Trennung beider Zustände ermöglicht wirksame Governance im privilegierten Raum.

KURZ ERKLÄRT · ELIGIBILITY VS. ACTIVATION

Eligibility ist wie ein Autoschlüssel im Schrank – man darf fahren, tut es aber gerade nicht. Activation ist der Moment, in dem der Schlüssel tatsächlich ins Zündschloss gesteckt wird. Governance interessiert sich vor allem für den zweiten Moment.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Jonas Vogel ist grundsätzlich berechtigt (Eligibility), Admin-Rechte auf LogiCore zu nutzen. Im Alltag hat er diese Rechte aber nicht aktiv – erst wenn er tatsächlich eine Systemwartung durchführen muss, aktiviert er sie für ein festes Zeitfenster (Activation). Danach verfallen sie automatisch wieder.

Just-in-Time und Just-Enough fachlich erklärt

Wie viel Berechtigung und wie viel Zeit benötigt man wirklich, um eine administrative Aufgabe zu erledigen? Die Antwort lautet meistens: deutlich weniger, als standardmäßig vergeben wird. In vielen Organisationen bedeutet administrativer Zugriff noch immer, dass umfangreiche Rechte dauerhaft bereitgestellt werden, selbst wenn sie nur gelegentlich benötigt werden – aus Governance-Sicht eine unnötige Risiko-Exposition.

Just-in-Time (JIT) steuert die zeitliche Dimension: Berechtigungen werden nicht dauerhaft zugewiesen, sondern erst dann bereitgestellt, wenn sie tatsächlich benötigt werden, und nach Abschluss der Aufgabe oder Ablauf eines Zeitfensters automatisch wieder entzogen. **Just-Enough Access (JEA)** steuert die inhaltliche Dimension: Eine Identität erhält genau die Berechtigungen, die für eine konkrete Aufgabe erforderlich sind – und nicht mehr, statt pauschaler Administratorrechte werden gezielt die benötigten Funktionen bereitgestellt.

JIT und JEA sind keine reinen Produktfunktionen moderner PAM-Lösungen, sondern Kontrollprinzipien, die das Risiko- und Audit-Profil einer Organisation grundlegend verändern: Privilegierte Berechtigungen werden vom Dauerzustand zum kontrollierten Ereignis. Die Kombination beider Prinzipien stellt sicher, dass die Kette aus Begründung, Verantwortung, Aktivierung und Kontrolle für jeden einzelnen Zugriffsschritt geschlossen bleibt.

Ein Zugriff sollte so kurz wie möglich und so restriktiv wie nötig sein.

Schutz vor Privilegien-Eskalation über den Änderungspfad

JIT und JEA steuern, wie privilegierter Zugriff im Moment der Aktivierung dosiert wird. Eine andere Gefahr entsteht schleichender, über einen ganz anderen Weg: die nachträgliche Aufwertung eines bereits bestehenden Rechts oder einer bestehenden Rolle. Klassische Kontrollen wirken meist nur bei der Zuweisung eines Zugriffs – nicht, wenn eine bereits zugewiesene Rolle im Nachhinein leicht verändert wird und dabei kritischer wird, als sie ursprünglich genehmigt war. Über viele kleine, für sich genommen unauffällige Änderungsschritte kann eine Berechtigung so „schleichend mächtiger“ werden, ohne dass jemals der strengere Prüfpfad durchlaufen wird, der für privilegierte Neuzugänge eigentlich vorgesehen ist.

Der wirksame Schutz liegt in einer strukturellen Regel: Eine Hochstufung Richtung privilegiert ist im leichteren Änderungsprozess eines bestehenden Objekts grundsätzlich nicht zulässig. Wer eine bestehende Rolle oder ein bestehendes Berechtigungsobjekt auf ein höheres Kritikalitätsniveau anheben will, muss stattdessen den vollständigen Neuanlage-Prozess mit der dafür vorgesehenen, strengeren Prüfung durchlaufen – denselben Weg, den auch ein komplett neues privilegiertes Recht nehmen müsste. Dasselbe Prinzip gilt sinngemäß für schrittweise Scope-Erweiterung: Eine Rolle, die durch viele kleine „Anpassung der Aufgaben“-Schritte am Ende deutlich mehr Systeme umfasst als ursprünglich vorgesehen, hat in der Summe denselben Effekt wie eine einzelne, große Kritikalitätserhöhung – auch wenn kein Einzelschritt für sich genommen auffällig wirkt.

Jede Kontrolle, die am Erstellungsprozess ansetzt, muss auch an jedem Pfad gelten, über den sich ein bestehendes Objekt nachträglich in denselben kritischen Zustand versetzen lässt.

Break-Glass als Ausnahme

Was passiert, wenn reguläre Kontrollsysteme ausfallen oder ein kritischer Vorfall sofortiges Handeln erfordert? Dann schlägt die Stunde des Notfallzugriffs – oft als „Break-Glass“ bezeichnet. Break-Glass ist die bewusste Ausnahme von Prinzipien wie Just-in-Time und Just-Enough Access, und gerade deshalb benötigt dieser Sonderweg besonders strenge Kontrollen. In vielen Organisationen sind Notfallzugriffe ein kritischer Graubereich: Technische Notfallkonten existieren, ihre Nutzung umgeht jedoch häufig reguläre Freigabeprozesse und schafft Lücken in der Nachweiskette.

Ein wirksames Notfallkonzept verschiebt den Schwerpunkt von präventiven auf detektive Kontrollen. **Detektion statt Prävention:** Im Notfall sind klassische Freigabeprozesse oft zu langsam, jede Aktivierung eines Break-Glass-Kontos muss deshalb sofort erkannt, protokolliert und automatisiert alarmiert werden. **Vollständige Isolation:** Notfallzugriffe müssen strikt vom regulären Betrieb getrennt sein und dürfen niemals zur operativen Abkürzung werden. **Rückwirkende Legitimation (Post-Event Review):** Da eine reguläre Vorabprüfung häufig nicht möglich ist, muss unmittelbar nach dem Einsatz eine vollständige Aufarbeitung erfolgen – was wurde getan, warum war der Zugriff erforderlich, wann wurde der Normalzustand wiederhergestellt?

Das größte Risiko von Break-Glass-Prozessen ist nicht der Ausnahmefall selbst, sondern die schleichende Normalisierung – sobald Notfallpfade genutzt werden, weil reguläre Prozesse als unbequem empfunden werden, beginnt die Governance zu erodieren. Governance im Krisenfall bedeutet nicht, Handlungsfähigkeit zu verhindern, sondern sicherzustellen, dass auch unter Zeitdruck Nachvollziehbarkeit, Verantwortung und Kontrolle erhalten bleiben.

MERKSATZ

Die Aktivierung eines Notfallzugriffs ist kein Ausstieg aus der Governance, sondern der Beginn eines eigenständigen Hochrisiko-Prozesses.

Je weniger präventive Kontrolle im Moment der Aktivierung möglich ist, desto lückenloser muss die Evidenz im Nachgang sein.

Break-Glass regelt die eine Richtung eines Notfalls: zusätzlichen Zugriff schnell und kontrolliert bereitzustellen. Die Kehrseite braucht einen eigenen, ebenso formalisierten Weg: eine Identität, bei der Missbrauch oder Kompromittierung vermutet wird,

muss ad hoc vollständig gesperrt werden können – unabhängig vom regulären Leaver-Prozess und ohne auf dessen üblichen Vorlauf zu warten. In vielen Organisationen existiert diese Fähigkeit nur informell über administrative Sonderrechte, nicht als eigener, protokollierter Lifecycle-Zustand mit klar definiertem Rückweg. Ein sauber modellierter Zustand „Sicherheitsvorfall“ schafft genau das: Auf Anweisung einer autorisierten Instanz – etwa Information Security oder Personalleitung – wird die Identität gesperrt, bis der Vorfall geklärt ist; danach folgt entweder die automatische Reaktivierung oder der Übergang in den regulären Leaver-Prozess. Wichtig ist die Trennung von Leaver: Ein Sicherheitsvorfall ist zunächst ein Verdacht, kein endgültiger Zustand, und sollte nicht dieselbe Unumkehrbarkeit signalisieren.

Break-Glass öffnet kontrolliert eine Tür, die sonst zu bleibt. Der Sicherheitsvorfall-Zustand schließt kontrolliert eine Tür, die sonst offensteht – beides braucht denselben Anspruch an Nachvollziehbarkeit.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Fällt LogiCore an einem Wochenende komplett aus und Jonas Vogel ist nicht erreichbar, kann ein anderer IT-Mitarbeiter das Notfallkonto „LogiCore-Admin“ nutzen. Die Aktivierung löst sofort einen Alarm aus, und am nächsten Werktag muss lückenlos dokumentiert werden, was genau getan wurde und warum der reguläre Weg nicht möglich war.

Session Accountability

Reicht es aus zu wissen, wer sich an einem kritischen System angemeldet hat, wenn nicht nachweisbar ist, was diese Person dort getan hat? Nein – hier entsteht aus Sicht von Governance, Revision und Audit eine kritische Kontrolllücke. Ein privilegierter Zugriff kann ordnungsgemäß beantragt, genehmigt und aktiviert worden sein; sind die tatsächlichen Aktivitäten innerhalb der Sitzung jedoch nicht nachvollziehbar, geht die Verbindung zwischen Berechtigung und Handlung verloren. Besonders kritisch wird dies bei generischen oder geteilten Konten, wo sich oft nicht mehr eindeutig feststellen lässt, welche natürliche Person tatsächlich gehandelt hat.

Genau hier setzt Session Accountability an: die lückenlose und belastbare Zuordnung jeder privilegierten Aktion zu einer verantwortlichen Person. Drei Säulen tragen das Prinzip. **Eindeutige Zuordnung:** Auch bei technisch generischen Konten darf die Ver-

bindung zur handelnden Person niemals verloren gehen – Anonymität oder geteilte Passwörter sind mit wirksamer Governance nicht vereinbar. **Nachweis der Wirkungsmacht:** Es geht nicht um allgemeine Mitarbeiterüberwachung, sondern um die Nachvollziehbarkeit sicherheitsrelevanter Handlungen – welche Konfiguration geändert, welche Berechtigung vergeben, welche Sicherheitsfunktion deaktiviert wurde. **Unveränderlichkeit des Audit-Trails:** Nachweise privilegierter Sitzungen müssen manipulationssicher gespeichert werden – wer Systeme administrieren kann, darf niemals gleichzeitig die eigenen Spuren beseitigen können.

Session Accountability schützt nicht nur die Organisation, sondern auch die handelnden Personen – sie schafft den objektiven Nachweis, dass privilegierte Tätigkeiten ordnungsgemäß und im Rahmen eines legitimen Auftrags durchgeführt wurden.

Verantwortung endet nicht mit dem Login. Sie muss über die gesamte Dauer der privilegierten Sitzung nachvollziehbar bleiben.

Reviews für privilegierten Zugriff

Reicht es aus, einmal im Jahr formal zu bestätigen, dass ein Administrator seine Rechte noch benötigt? Nein. Bei privilegierten Zugriffen ist die klassische, rein periodische Rezertifizierung oft wenig wirksam: Lange Listen werden bestätigt, ohne die tatsächliche Notwendigkeit kritisch zu hinterfragen – während ein überflüssiger Standardzugriff meist ein begrenztes Risiko darstellt, kann ein nicht mehr benötigter privilegierter Zugriff erhebliche Auswirkungen auf die gesamte Organisation haben. Das Review wird so zur Pflichtübung ohne echten Beitrag zur Risikoreduzierung.

Effektive Reviews für privilegierte Berechtigungen basieren auf drei Säulen. **Fokus auf die Berechtigung (Eligibility):** Geprüft wird nicht ein dauerhaft aktiver Zugriff, sondern die grundsätzliche Legitimation, privilegierte Rechte bei Bedarf aktivieren zu dürfen. **Einbezug von Aktivierungsdaten:** Ein Review ohne Nutzungskontext bleibt unvollständig – Aktivierungsdaten liefern wichtige Hinweise darauf, ob eine Berechtigung tatsächlich benötigt wird, und ergänzen die fachliche Bewertung um objektive Nutzungsevidenz. **Ereignisgesteuerte Frequenz:** Ein fester jährlicher Zyklus reicht häufig nicht aus – Änderungen von Aufgaben, Verantwortlichkeiten, Projekten oder Systemlandschaften sollten unmittelbar eine Neubewertung auslösen.

Die Verantwortung für privilegierte Reviews darf nicht allein bei der IT oder den Administratoren liegen – Application Owner und fachliche Verantwortliche müssen aktiv eingebunden werden, denn sie tragen die Verantwortung dafür, wer Systeme verändern, Kontrollen umgehen oder hochkritische Funktionen ausführen kann. Jedes nicht erforderliche administrative Recht erhöht die Angriffsfläche und das Risiko.

Verantwortung im privilegierten Raum zeigt sich darin, Berechtigungen konsequent zu hinterfragen und bei fehlender Notwendigkeit kontrolliert zurückzuführen.

Grenze des Modells: Just-in-Time und Just-Enough Access setzen voraus, dass Zielsysteme granulare, kurzfristig aktivierbare Berechtigungen technisch überhaupt unterstützen. Bei älteren Anwendungen oder Legacy-Systemen ohne feingranulare Rechteverwaltung – oft mit nur einer einzigen "Admin"-Rolle statt abgestufter Berechtigungen – lässt sich dieses Prinzip technisch nicht umsetzen, unabhängig davon, wie gut der Governance-Prozess dahinter ist. In solchen Fällen verschiebt sich der Schwerpunkt zwangsläufig von Prävention auf Kompensation: engmaschigeres Monitoring, kürzere Reviewzyklen und eine explizit dokumentierte Risikoakzeptanz, bis das System selbst modernisiert oder abgelöst wird.

8

Non-Person Identities

Non-Person Identities als Governance-Domäne

Wer kontrolliert eigentlich die Zugriffe, wenn kein Mensch am Keyboard sitzt? In vielen IGA-Initiativen stehen menschliche Identitäten im Mittelpunkt. In modernen IT-Landschaften machen Non-Person Identities (NPIs) – Service Accounts, API-Schlüssel, Zertifikate, Bots oder Workload-Identitäten – jedoch einen erheblichen Teil aller Identitäten aus. In der Praxis werden sie häufig als technische Konfigurationen behandelt: Sie entstehen im Rahmen von Projekten oder Integrationen und bleiben anschließend oft dauerhaft bestehen – ohne klaren Lifecycle, ohne definierte Verantwortlichkeiten, ohne ausreichende Kontrolle. Damit werden NPIs zu einem der attraktivsten Angriffsziele moderner IT-Umgebungen.

Drei Aspekte begründen, warum NPIs eine eigenständige Governance-Domäne brauchen. Die **Identitäten-Inflation**: Die Anzahl von NPIs wächst in vielen Organisationen deutlich schneller als die Zahl menschlicher Identitäten, ohne automatisierte Erfassung entsteht ein kaum überschaubarer Bestand. Die **verdeckte Wirkungsmacht**: NPIs besitzen häufig weitreichende Berechtigungen und agieren automatisiert und permanent – ein kompromittiertes NPI-Konto kann erheblichen Schaden verursachen. Und der **fehlende natürliche Lifecycle**: Hinter einer NPI steht keine Person, sie wechselt weder Abteilung noch verlässt sie das Unternehmen – ohne explizite Lifecycle-Steuerung bleiben Berechtigungen oft aktiv, obwohl der ursprüngliche Zweck längst entfallen ist.

Eine NPI darf niemals im organisatorischen Vakuum existieren. Jede NPI benötigt einen klaren fachlichen Zweck, einen verantwortlichen Owner und definierte Regeln für Erstellung, Nutzung, Überwachung und Entzug – dieselben Fragen wie bei men-

schlichen Identitäten: Wer verantwortet diesen Zugriff, wofür wurde er eingerichtet, welche minimalen Berechtigungen benötigt er tatsächlich?

MERKSATZ

Eine Non-Person Identity ist technisch autonom, darf aber organisatorisch niemals eigentümerlos sein.

KURZ ERKLÄRT · NON-PERSON IDENTITY (NPI)

Eine NPI ist eine „Identität“ ohne Mensch dahinter – ein technisches Konto, das automatisiert Aufgaben erledigt, etwa Daten zwischen zwei Systemen austauscht. Sie braucht trotzdem einen menschlichen Verantwortlichen, so wie ein Firmenwagen einen Halter braucht, auch wenn er von verschiedenen Personen gefahren wird.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Der LogiCore-Sync-Service ist ein technisches Konto, das jede Nacht automatisch Bestandsdaten zwischen LogiCore und dem Kundenportal NordConnect abgleicht. Kein Mensch meldet sich mit diesem Konto an – trotzdem braucht es einen Owner, der verantwortet, welche Rechte der Service tatsächlich hat.

NPI-Taxonomie

Kann man einen statischen Datenbank-Service-Account nach denselben Regeln steuern wie einen dynamischen API-Schlüssel oder einen RPA-Bot? Nein. In vielen Organisationen existiert für alle nicht-menschlichen Zugriffe dennoch nur eine einzige Schublade: „Technische Benutzer“. Wer alles gleich behandelt, kann jedoch keine risikogerechten Kontrollen etablieren – eine weitreichende Cloud-Workload braucht andere Steuerungsmechanismen als ein lokaler Systemdienst.

Eine praxistaugliche Taxonomie unterscheidet nach Funktion, Risiko und Verhalten. **Infrastruktur- und System-Accounts** sind meist langlebig und fest mit Servern, Betriebssystemen oder Datenbanken verbunden – der Governance-Fokus liegt auf Überwachung, Secret-Management und regelmäßiger Rotation. **Anwendungs- und Workload-Identitäten** umfassen API-Keys, Zertifikate, Tokens sowie Cloud- und Container-Workloads – hochdynamisch und in großer Zahl vorhanden, hier bedeutet Go-

vernance vor allem kurze Laufzeiten, kontrollierte Ausstellung und nachvollziehbare Herkunft. **RPA-Identitäten und Bots** führen Prozesse automatisiert auf Anwendungsebene aus, interagieren oft mit mehreren Systemen gleichzeitig und verfügen deshalb nicht selten über weitreichende Berechtigungen – Governance muss hier sicherstellen, dass jede dieser Identitäten einer verantwortlichen menschlichen Instanz zugeordnet bleibt.

Die Taxonomie ist kein theoretisches Ordnungssystem, sondern die Grundlage für Risikoanalysen, Kontrollanforderungen und Ownership-Modelle. Erst wenn bekannt ist, welcher Kategorie eine NPI angehört, lassen sich die richtigen Fragen stellen: Wie hoch ist das Risiko, welche Kontrollen sind erforderlich, wie häufig muss überprüft werden, wer trägt die Verantwortung?

Die Schublade „Technischer Benutzer“ kaschiert Unterschiede und Risiken. Erst eine klare NPI-Taxonomie ermöglicht eine wirksame und risikogerechte Steuerung.

Ownership für technische Identitäten

Wer übernimmt die Verantwortung, wenn ein technisches Konto Daten exfiltriert oder ein kritisches System lahmlegt? In vielen Organisationen lautet die Antwort: Niemand weiß es genau, oder „das ist ein Systemkonto, das gehört der IT.“ Aus Governance-Sicht ist das ein fundamentaler Trugschluss – Verantwortung lässt sich nicht an Quellcode, Server oder Abteilungen delegieren. Die gleichen Prinzipien, die für menschliche Identitäten gelten, gelten auch für technische: Ownership und Accountability enden nicht dort, wo kein Mensch mehr direkt am System arbeitet. NPIs werden von Entwicklern angelegt, von Administratoren betrieben und von Systemen genutzt – die fachliche Verantwortung bleibt dabei oft ungeklärt, mit der Folge, dass NPIs verwaisen: übersehen bei Reviews, Berechtigungen nicht hinterfragt, Konten aktiv, obwohl die zugehörige Anwendung längst abgeschaltet wurde.

Ein klares Ownership-Modell verlangt die **Koppelung an die fachliche Verantwortung**: Eine NPI dient immer einer Anwendung, Schnittstelle oder einem Geschäftsprozess, die Ownership muss dort liegen, wo Nutzen und Risiko entstehen, typischerweise beim Application oder Process Owner. Es verlangt die **Trennung von fachlicher und technischer Verantwortung**: Der fachliche Owner verantwortet Zweck, Notwendigkeit und Berechtigungsumfang, IT oder DevOps die technische Umsetzung

und Absicherung. Und es verlangt: **keine NPI ohne Owner** – eine technische Identität darf erst bereitgestellt werden, wenn ein namentlich bekannter Owner dokumentiert ist, und ein Verantwortungswechsel braucht einen strukturierten Übergabeprozess.

MERKSATZ

Systeme handeln automatisiert. Verantwortung bleibt immer menschlich.

Ownership für NPIs ist kein Asset-Management, sondern Risikosteuerung. Bei einem Audit oder Sicherheitsvorfall muss sofort klar sein, welche fachliche Instanz rechnungspflichtig ist.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Als der LogiCore-Sync-Service eingerichtet wird, ist Tom Brenner aus dem IAM-Team zunächst technisch beteiligt – aber fachlicher Owner ist der Application Owner von LogiCore, weil dort Nutzen und Risiko des Service entstehen. Verlässt Tom Brenner das Unternehmen, bleibt der Service trotzdem eindeutig zugeordnet.

NPI-Lifecycle

Wie stellt man sicher, dass eine technische Identität gelöscht wird, wenn das zugehörige System gar nicht mehr existiert? In der Praxis meistens gar nicht – technische Identitäten überleben Anwendungen oft um Jahre. Während der Identitäts-Lifecycle für menschliche Mitarbeiter etabliert ist, existiert für NPIs häufig kein strukturierter Lebenszyklus; einmal eingerichtet, verbleiben sie oft dauerhaft im System. Die Folge ist ein kontinuierlich wachsender Bestand verwaister, aber häufig hochprivilegierter Identitäten.

Ein definierter NPI-Lifecycle folgt denselben Governance-Prinzipien wie bei menschlichen Identitäten. **Kontrollierte Erstellung (Provisioning)**: Eine NPI darf nicht ungesteuert entstehen, die Bereitstellung erfordert einen fachlichen Owner, einen dokumentierten Zweck und idealerweise eine definierte Laufzeit. **Kontinuierliche Pflege (Maintenance & Review)**: Während ihrer Lebensdauer muss die NPI regelmäßig überprüft werden – Bestätigung der Ownership, Überprüfung der Berechtigungen, Rotati-

on von Secrets, Zertifikaten oder Tokens. **Konsequente Deaktivierung (Deprovisioning):** Wird eine Anwendung abgeschaltet oder ein Prozess eingestellt, müssen die zugehörigen technischen Identitäten identifiziert, deaktiviert und entfernt werden.

Der Lifecycle einer NPI darf nicht isoliert betrachtet werden – er muss direkt an den Lebenszyklus der Anwendung, Plattform oder des Geschäftsprozesses gekoppelt sein. Endet die Lebensdauer des übergeordneten Objekts, muss auch die zugehörige technische Identität konsequent entzogen werden.

Systeme räumen sich nicht selbst auf. Genau dafür existiert Governance.

Least Privilege für NPIs

Warum läuft eine Applikationsschnittstelle mit Administratorrechten, obwohl sie lediglich Daten lesen muss? Die Antwort lautet in der Praxis oft: „Damit es beim Deployment keine Probleme gibt.“ Während bei menschlichen Identitäten zumindest versucht wird, Least Privilege durchzusetzen, gilt im Bereich technischer Identitäten häufig das Gegenteil: maximale Rechte, um technische Reibung zu vermeiden. Aus Governance-Sicht ist das fatal – NPIs agieren automatisiert, permanent und mit hoher Geschwindigkeit, wird eine überprivilegierte technische Identität kompromittiert, ist das Schadensausmaß oft deutlich höher als bei einem kompromittierten Standard-Benutzer.

Drei Hebel setzen Least Privilege für NPIs konsequent um. **Granularität statt Pauschalrechte:** Technische Identitäten benötigen keine generischen „Super-User“-Rollen, Berechtigungen müssen präzise auf die tatsächlich benötigten API-Endpunkte, Datenbanktabellen oder Cloud-Ressourcen zugeschnitten werden. **Kontextbasierte Einschränkungen (Guardrails):** Da das Verhalten von NPIs weitgehend vorhersehbar und deterministisch ist, lässt sich ihr Aktionsradius gezielt begrenzen, etwa auf definierte IP-Bereiche, Cluster oder Kommunikationspfade. **Nutzungsbasiertes Right-Sizing:** Berechtigungen dürfen keine statische Einbahnstraße sein – durch kontinuierliche Analyse von Zugriffs- und Systemprotokollen wird überprüft, welche Berechtigungen tatsächlich genutzt werden, nicht genutzte Rechte werden konsequent entzogen.

Dabei gilt die Umkehr der Beweislast: Entwickler und Systemverantwortliche müssen begründen, warum eine NPI ein bestimmtes Recht benötigt – nicht die Governance, warum es entzogen werden sollte.

Rechte auf Vorrat haben im automatisierten Raum keinen Platz.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Der LogiCore-Sync-Service braucht eigentlich nur Lesezugriff auf Bestandsdaten und Schreibzugriff auf ein einzelnes Datenfeld in NordConnect. Beim ersten Aufsetzen hatte er versehentlich vollen Schreibzugriff auf die gesamte Datenbank erhalten, „damit nichts schiefgeht“. Eine nachträgliche Prüfung deckt das auf und reduziert die Rechte auf das tatsächlich benötigte Minimum.

Rotation und Änderung

Ein Secret, das seit Jahren unverändert im Einsatz ist, gilt in vielen IT-Landschaften nicht als Risiko, sondern als „läuft ja stabil“ – genau das ist das Problem. Während Passwörter menschlicher Identitäten meist durch Richtlinien und Multi-Faktor-Authentifizierung geschützt sind, hängen NPIs fast ausschließlich an statischen Secrets. Manuelle Rotation gilt als aufwendig, ein möglicher Ausfall als reales Risiko – „Never touch a running system“ wird so zur impliziten Governance-Strategie. Je länger ein Secret unverändert bleibt, desto größer das Zeitfenster für einen dauerhaften, unmerkten Zugriff durch Dritte.

Drei Prinzipien steuern den Lebenszyklus technischer Geheimnisse. **Automatisierung statt Handarbeit:** Manuelle Prozesse skalieren bei tausenden technischen Identitäten nicht, die regelmäßige Rotation von Passwörtern, Schlüsseln und Zertifikaten sollte über zentrale Secrets-Management-Systeme automatisiert erfolgen. **Kryptographische Vergänglichkeit:** Je kürzer die Gültigkeit eines Secrets, desto kleiner das Angriffsfenster – moderne Architekturen setzen auf dynamische Secrets, die erst bei Bedarf erzeugt werden und nach kurzer Zeit automatisch verfallen. **Governance von Konfigurationsänderungen:** Jeder Secret-Wechsel ist ein sicherheitskritisches Ereignis und muss lückenlos dokumentiert werden – unautorisierte Änderungen oder Konfigurations-Drift müssen frühzeitig erkannt werden.

Die Fähigkeit, technische Zugangsdaten jederzeit schnell und kontrolliert rotieren zu können, ist ein entscheidender Gradmesser für die Governance-Reife einer Organisation. Wer Angst davor hat, ein technisches Passwort zu ändern, weil unklar ist, welche Anwendungen danach ausfallen, hat die Governance über seine IT-Landschaft faktisch bereits verloren.

Das Ziel von Governance ist es, den Secret-Wechsel vom risikobehafteten Ausnahmefall zu einem kontrollierten Standardprozess zu machen.

Anti-Pattern: Dauerhafte technische Admin-Zugriffe

Ein Service Account läuft seit drei Jahren mit Administratorrechten. Niemand hat sie beantragt, niemand reviewt sie, niemand kann sagen, wer eigentlich noch dafür verantwortlich ist. Trotzdem stört das kaum jemanden – das System läuft ja. Genau das ist das Problem. Für menschliche Administratoren haben sich Just-in-Time-Zugriffe, getrennte Adminkonten und regelmäßige Reviews längst etabliert; bei Service Accounts, API-Identitäten und Automatisierungsprozessen sieht die Realität oft anders aus – dauerhaft weitreichende Berechtigungen, rund um die Uhr, oft ohne aktiven Owner. Die Begründung klingt fast immer gleich: „Sonst läuft das Tool nicht fehlerfrei.“

Drei Folgen zeigen, warum das keine technische Notwendigkeit, sondern eine unnötige Risiko-Exposition ist: das **permanente Angriffsfenster**, weil ein kompromittiertes technisches Konto mit Administratorrechten dauerhaft privilegierten Zugriff ermöglicht und lange unentdeckt bleiben kann; der **fehlende Nutzungskontext**, weil permanent privilegierte NPIs die Erkennung von Anomalien erschweren – missbräuchliche Aktivitäten gehen leichter im normalen Systembetrieb unter; und die **Umgehung organisatorischer Kontrollen**, weil dauerhafte Admin-Rechte die fachliche Entscheidungskette umgehen – Automatisierung ersetzt dann Kontrolle, nicht Verantwortung.

„Das System benötigt diese Rechte dauerhaft“ darf niemals ungeprüft akzeptiert werden. Für NPIs gelten dieselben Grundprinzipien wie für privilegierte menschliche Zugriffe: Least Privilege, klare Ownership, regelmäßige Reviews und, wo technisch möglich, zeitlich begrenzte Berechtigungen. Lässt sich ein dauerhafter privilegierter Zugriff wirklich nicht vermeiden, braucht es risikoadäquate Kompensationskontrollen – nicht Stillschweigen.

MERKSATZ

Dauerhafte Administratorrechte für technische Identitäten sind selten ein Zeichen stabiler Systeme. Sie sind meist ein Hinweis auf fehlende Governance.

Grenze des Modells: Das Prinzip "ein Owner pro NPI" trägt gut bei hunderten oder wenigen tausend technischen Identitäten. In sehr großskaligen, Cloud-nativen Umgebungen mit zehntausenden kurzlebigen Workload-Identitäten, die sich automatisiert selbst erzeugen und wieder auflösen, wird die individuelle Owner-Zuordnung pro Einzelidentität praktisch unhandhabbar. Governance verschiebt sich dort auf eine höhere Abstraktionsebene: Nicht mehr jede einzelne Instanz braucht einen Owner, sondern die Vorlage oder Pipeline, die diese Identitäten erzeugt – Ownership wandert vom Objekt zum Bauplan.

9

Evidenz & Audit

Kontrollkette von Entscheidung bis Entzug

Eine Berechtigung wurde vor 14 Monaten korrekt genehmigt – dokumentiert, autorisiert, sauber begründet. Heute besteht sie immer noch, obwohl sich die Rolle der Person seither zweimal geändert hat. Genau an dieser Stelle zeigt sich, ob ein IGA-Programm nur gute Absichten verwaltet oder tatsächlich Kontrolle ausübt. Das Problem liegt selten am Startpunkt – es entsteht später, wenn der Zusammenhang zwischen ursprünglicher Entscheidung und aktuellem Zustand im Laufe des Lifecycles verloren geht und aus dem einmal Legitimierten stillschweigend ein Dauerzustand wird.

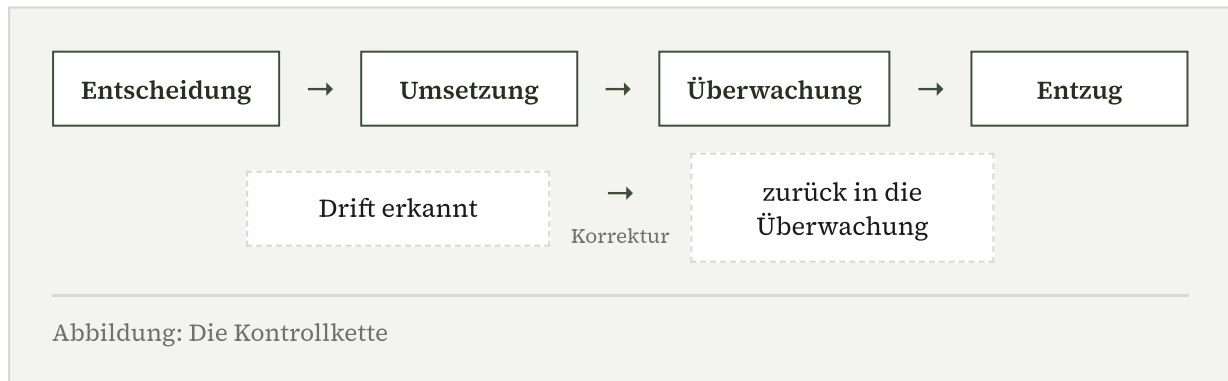
Wirksame Identity Governance braucht deshalb eine durchgängige Kontrollkette aus vier Gliedern, die nur im Zusammenspiel funktionieren. Die **Entscheidung**: Jede Berechtigung benötigt eine fachliche Legitimation – wer erhält welchen Zugriff, warum und für welchen Zeitraum, beantwortet und dokumentiert vor der technischen Umsetzung. Die **Umsetzung**: Die Entscheidung muss zuverlässig in technische Berechtigungen überführt werden – manuelle Änderungen ohne Bezug zur ursprünglichen Entscheidung sind eine der häufigsten Ursachen für Berechtigungs-Drift. Die **Überwachung**: Governance endet nicht mit der Bereitstellung, der Ist-Zustand muss kontinuierlich mit dem legitimierten Soll-Zustand abgeglichen werden. Und der **Entzug**: Sobald die fachliche Grundlage entfällt, muss auch die Berechtigung entzogen werden.

Eine Kontrollkette ist nur so stark wie ihr schwächstes Glied – eine dokumentierte Entscheidung nützt nichts, wenn sie fehlerhaft umgesetzt wird, eine korrekte Umsetzung verliert ihren Wert, wenn Berechtigungen nach Wegfall der Legitimation beste-

hen bleiben. Identity Governance verwaltet deshalb keine Benutzerkonten, sondern stellt sicher, dass diese Kontrollkette jederzeit geschlossen bleibt.

MERKSATZ

Wer die Kontrollkette beherrscht, beherrscht das Risiko. Wer sie unterbricht, schafft Berechtigungs-Drift.

**PRAXISBEISPIEL · NORDFELD LOGISTIK**

Ein Zugriff auf LogiCore wird korrekt entschieden und umgesetzt. Ein Jahr später wechselt die Person die Abteilung, doch niemand überwacht, ob der alte Zugriff noch gebraucht wird – die Überwachung fehlt, die Kette reißt. Der Zugriff bleibt bestehen, obwohl die fachliche Grundlage längst entfallen ist.

Präventive, detektive und korrektive Kontrollen

Ein Unternehmen mit lückenloser Rezertifizierungshistorie gilt vielen als vorbildlich kontrolliert – und kann trotzdem seit Monaten überprivilegierte Konten führen, unbemerkt, weil niemand danach gesucht hat. Der Grund liegt selten in fehlender Sorgfalt, sondern in einer einseitigen Kontrollarchitektur: Liegt der Fokus fast ausschließlich auf detektiven Kontrollen wie Audits oder Rezertifizierungen, ist das, als würde man erst nach einem Einbruch prüfen, ob die Haustür abgeschlossen war.



Ein wirksames IGA-System kombiniert drei Kontrolltypen. **Präventive Kontrollen** greifen, bevor ein Risiko entsteht, und verhindern unsichere Zustände von Anfang an – etwa Least Privilege, Genehmigungs-Workflows, Separation of Duties, keine Berechtigung ohne fachlichen Owner. **Detektive Kontrollen** identifizieren Abweichungen zwischen Soll- und Ist-Zustand, wenn präventive Maßnahmen nicht ausgereicht haben – etwa Rezertifizierungen, Drift-Monitoring, Session-Logging, Analyse ungewöhnlicher Nutzungsmuster. **Korrektive Kontrollen** stellen den gewünschten Soll-Zustand wieder her, nachdem eine Abweichung erkannt wurde – etwa automatischer Berechtigungsentzug, Sperrung verwaister Konten oder Rückbau befristeter Ausnahmen.

Keine dieser Kontrollarten kann die anderen ersetzen: Präventive Kontrollen vermeiden Risiken, detektive schaffen Transparenz, korrektive stellen Compliance wieder her. Wer ausschließlich auf Detektion setzt, akzeptiert Berechtigungs-Drift als Normalzustand. Wer ausschließlich präventiv arbeitet, riskiert unflexible Prozesse.

Prävention verhindert Risiken. Detektion macht sie sichtbar. Korrektur beseitigt sie. Erst ihr Zusammenspiel macht Identity Governance wirksam.

Evidenz vs. Logs

KURZ ERKLÄRT · EVIDENZ VS. LOG

Ein Log ist wie eine Kamera-Aufnahme: Sie zeigt, dass jemand durch eine Tür gegangen ist. Evidenz ist der Hausausweis, der zeigt, warum die Person berechtigt war, durch diese Tür zu gehen. Für ein Audit reicht die Kamera-Aufnahme allein nicht.

Ein Auditor fragt, wer einen Administratorzugriff freigegeben hat. Die Antwort scheint vorzuliegen: ein Log, das die Vergabe protokolliert, mit Zeitstempel, Benutzer und Aktion. Reicht das als Nachweis? In der Praxis werden technische Logs regelmä-

ßig mit Governance-Evidenz verwechselt, dabei erfüllen sie grundlegend unterschiedliche Aufgaben. Ein Log zeigt, dass etwas passiert ist. Evidenz belegt, warum es passieren durfte. Ein Log dokumentiert die Ausführung, Evidenz dokumentiert die Legitimation.

Damit eine Governance-Kette belastbar ist, muss Evidenz drei Anforderungen erfüllen. Den **Ursprung der Entscheidung**: Ein System-Log beweist keine Berechtigung, die eigentliche Evidenz ist der Decision Record oder die dokumentierte Genehmigung des fachlichen Owners. **Kontext und Begründung**: Logs enthalten meist keine Informationen darüber, warum ein Zugriff vergeben wurde – Governance-Evidenz dokumentiert den fachlichen Kontext, etwa Projektbezug, Timeboxing oder besondere Auflagen. **Integrität und Verknüpfung**: Logs können rotiert oder gelöscht werden, Governance-Evidenz muss unveränderbar sein und eindeutig mit der jeweiligen Berechtigung verknüpft werden.

Erst die Kombination aus fachlicher Entscheidung und technischer Ausführung ergibt einen lückenlosen Policy-to-Decision Trace – und damit einen belastbaren Audit-Trail. Wer sich im Audit ausschließlich auf Logs stützt, versucht nachträglich zu rekonstruieren, ob eine Entscheidung legitim war – das ist Forensik, keine Governance. Ein reifes IGA-System erzeugt die Evidenz bereits im Moment der Entscheidung; das technische Log bestätigt anschließend lediglich deren Umsetzung.

MERKSATZ

Logs zeigen, was passiert ist. Evidenz belegt, warum es passieren durfte. Erst beides zusammen macht Governance auditfähig.

Wirksame Reviews und Attestierungen

Ein Reviewer bekommt eine Liste mit 340 Berechtigungen, 20 Minuten Zeit. Am Ende: alle bestätigt, keine einzige entzogen. Kein Einzelfall – dieses Muster wiederholt sich Quartal für Quartal, bei völlig unterschiedlichen Teams und Branchen. Der Grund liegt selten bei den Reviewern selbst, sondern darin, wie Reviews aufgesetzt sind: Steht nicht die Qualität der Entscheidung im Mittelpunkt, sondern der Abschluss des Prozesses, wird ein Review zum administrativen Akt, der Listen bestätigt, statt zur Governance-Kontrolle, die prüft, ob ein Zugriff weiterhin fachlich gerechtfertigt ist. Sieht ein Reviewer hunderte Berechtigungen ohne Kontext, entsteht fast zwangsläufig „Rubber-Stamping“ – formal wurde geprüft, tatsächlich aber kaum bewertet.

Wirksame Reviews folgen einer qualitativen statt rein quantitativen Logik. **Risikoorientierte Prüfung:** Nicht jede Berechtigung erfordert dieselbe Aufmerksamkeit – privilegierte Zugriffe, sensible Daten oder SoD-Konflikte verdienen deutlich intensivere Prüfung als Standard-Berechtigungen. **Kontext statt Listen:** Eine fundierte Entscheidung braucht Informationen – wann wurde der Zugriff zuletzt genutzt, wer hat ihn genehmigt, gibt es vergleichbare Berechtigungen innerhalb der Peer Group. **Rezertifizierung vs. Attestierung:** Eine Rezertifizierung überprüft bestehende Berechtigungen, eine Attestierung geht einen Schritt weiter – der Reviewer bestätigt ausdrücklich, dass er die Verantwortung für den Zugriff übernimmt, und diese Entscheidung wird als Governance-Evidenz dokumentiert.

Ein gutes Review-System bewertet nicht nur Berechtigungen, sondern auch den Review-Prozess selbst. Werden trotz regelmäßiger Reviews praktisch nie Berechtigungen entzogen, ist das kein Zeichen perfekter Governance, sondern häufig ein Hinweis auf ungeeignete Prozesse oder fehlenden Entscheidungskontext. Die wirksamste Governance reduziert den Aufwand für Reviews durch saubere Lifecycle-Prozesse und automatisierte Entzüge – Reviews bleiben die zweite Verteidigungslinie, nicht die erste.

Ein oft übersehener, aber entscheidender Punkt ist der Umgang mit Nichtreaktion: Reagiert der zuständige Reviewer innerhalb der vorgesehenen Frist nicht – auch nach Erinnerung nicht –, muss der Zugriff automatisch entzogen werden, nicht stillschweigend bestehen bleiben. Viele Implementierungen wählen den bequemeren, aber sicherheitstechnisch falschen Default: keine Reaktion gilt als Genehmigung. Das konterkariert den Zweck der Rezertifizierung vollständig, weil Owner-Passivität so faktisch zur Dauergenehmigung wird. In jedem Freigabeprozess mit Zeitfenster sollte der Default bei Nichtreaktion der konservativere Zustand sein, nicht der bequemere – ergänzt um einen Eskalationsweg, der einen unbearbeiteten Vorgang an eine Vertretung oder die nächste Führungsebene weiterreicht, bevor der automatische Entzug greift.

Ein Review bestätigt keine Liste. Es bestätigt Verantwortung. Erst wenn Entscheidungen nachvollziehbar getroffen werden, entsteht echte Governance.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Statt Sabine Krüger eine Liste mit 80 Zugriffen ihres Teams zur pauschalen Bestätigung vorzulegen, erhält sie nur die drei kritischen Fälle mit Kontext: einen Zugriff, der seit sechs Monaten nicht genutzt wurde, einen mit ungewöhnlich weitreichenden Rechten, und einen, der einer bereits gewechselten Rolle zuzuordnen ist. So kann sie tatsächlich bewerten, statt nur abzunicken.

Umgang mit Findings

Ein Finding taucht im Auditbericht auf. Die erste Reaktion in vielen Organisationen: schnell schließen, relativieren, möglichst aus dem nächsten Audit heraushalten. Diese Reaktion ist menschlich verständlich, erkennt aber, was ein Finding eigentlich ist: kein Makel, sondern ein Signal. Es zeigt an, dass die Governance-Kontrollkette an einer konkreten Stelle nicht wie vorgesehen funktioniert hat. Wird dieses Signal lediglich formal quittiert statt inhaltlich verstanden, bleibt die eigentliche Ursache unberührt – und das Finding kehrt zurück.

Ein professioneller Umgang folgt drei Grundprinzipien. **Ursachenanalyse statt Symptombekämpfung:** Die meisten Findings sind Symptome fehlerhafter Prozesse oder Rollenmodelle – wer lediglich eine Berechtigung entfernt, ohne Lifecycle oder Rol-
lendesign zu korrigieren, produziert dasselbe Finding beim nächsten Audit erneut.

Risikobasierte Priorisierung: Nicht jedes Finding besitzt dieselbe Kritikalität, die Priorisierung orientiert sich am tatsächlichen Risiko – ein unnötiges Administratorrecht wiegt schwerer als eine fehlende Dokumentation. **Nachhaltige Wirksamkeit:** Eine Remediation ist erst abgeschlossen, wenn ihre Wirksamkeit nachgewiesen wurde – technische Korrektur, Prozessanpassung und Nachweis dauerhafter Behebung, etwa durch Monitoring oder Drift-Kontrollen.

Findings sind der Feedback-Mechanismus einer Governance-Architektur. Treten dieselben Findings wiederholt auf, versagt nicht das Audit, sondern der Verbesserungsprozess dahinter. Ein reifes IGA-Programm behandelt Findings deshalb als Input für den kontinuierlichen Verbesserungsprozess – Ziel ist die systematische Beseitigung wiederkehrender Ursachen, nicht die Bereinigung einzelner Symptome.

MERKSATZ

Ein Finding zeigt nicht nur einen Fehler. Es zeigt, wo Governance verbessert werden muss.

Governance-KPIs

In vielen Steering-Committees fällt zum Status der Identity Governance ein Satz, der nie wirklich hinterfragt wird: „Wir sind auf einem guten Weg.“ Das klingt beruhigend – nur, woran genau macht die Organisation das fest? Ohne messbare Kennzahlen bleibt Identity Governance eine Blackbox. Dabei ist nicht jede Kennzahl automatisch ein Governance-KPI: Die Anzahl der Benutzer oder Rollen beschreibt den Bestand, nicht die Qualität der Governance.

Wirksame Steuerung konzentriert sich auf vier Perspektiven. **Identitätsqualität (Hygiene-KPIs):** wie sauber der Bestand ist – verwaiste Konten, Identitäten ohne Owner, Zeit bis zur vollständigen Deprovisionierung nach einem Austritt. **Prozesseffizienz (Velocity-KPIs):** wie schnell und zuverlässig Governance-Prozesse laufen – Durchlaufzeiten von Berechtigungsanträgen, Automatisierungsgrad, Fehlerquoten bei Provisionierungen. **Compliance und Risiko (Risk-KPIs):** wie wirksam die Kontrollen funktionieren – Wiederholungsrate von Audit-Findings, Berechtigungen ohne dokumentierte Entscheidung, tatsächlich entzogene Rechte im Rahmen von Reviews. **Go-**

vernance-Abdeckung (Coverage-KPIs): wie groß der bereits gesteuerte Teil der IT-Landschaft ist – IGA-Abdeckung von Anwendungen, privilegierten Konten oder Non-Person Identities.

Governance-KPIs sind kein Reporting für das Management, sondern das Navigationsinstrument der Governance – besonders aussagekräftig werden sie erst im Zusammenspiel. Eine kurze Bearbeitungszeit für Berechtigungen ist nur dann ein Erfolg, wenn gleichzeitig die Kontrollqualität hoch bleibt. Auch ein steigender Governance-Debt-Index – die Summe offener Findings, Ausnahmen oder überfälliger Reviews – ist ein wichtiges Frühwarnsignal für wachsende Risiken.

Governance beginnt dort, wo Risiken messbar werden. Denn nur was gemessen wird, kann auch wirksam gesteuert werden.

Grenze des Modells: KPIs sind nur so belastbar wie die Datenqualität, aus der sie entstehen. Wenn die zugrundeliegenden Systeme selbst lückenhaft sind – etwa weil nicht alle Anwendungen an die IGA angebunden sind –, suggerieren gute KPI-Werte eine Kontrolle, die für die nicht erfassten Bereiche gar nicht existiert. Eine hohe Policy-Adherence-Rate ist wertlos, wenn sie sich nur auf 60 Prozent der tatsächlichen Systemlandschaft bezieht. Governance-KPIs brauchen deshalb immer eine Coverage-Angabe als Kontext, nicht nur einen isolierten Prozentwert.

Diese Coverage-Lücke lässt sich nicht allein durch mehr Anbindungen schließen – manche Anwendungen sind für eine saubere Governance-Anbindung schlicht noch nicht bereit, etwa ohne API-Zugriff, ohne Trennung von Entwicklungs-, Test- und Produktivumgebung, ohne dokumentiertes Berechtigungskonzept oder ohne benannten Verantwortlichen. Der Reflex, solche Systeme trotzdem mit unsauberen Workarounds anzubinden, verwässert die Governance-Qualität insgesamt und verschlechtert die Aussagekraft gerade der Coverage-KPIs, die eigentlich Klarheit schaffen sollen. Tragfähiger ist ein expliziter Mindeststandard für die Aufnahme: "Diese Anwendung ist noch nicht IGA-fähig" ist eine legitime, dokumentierte Aussage, die auf Nachrüstung hinwirkt, statt stillschweigend eine schwache Anbindung zu akzeptieren.

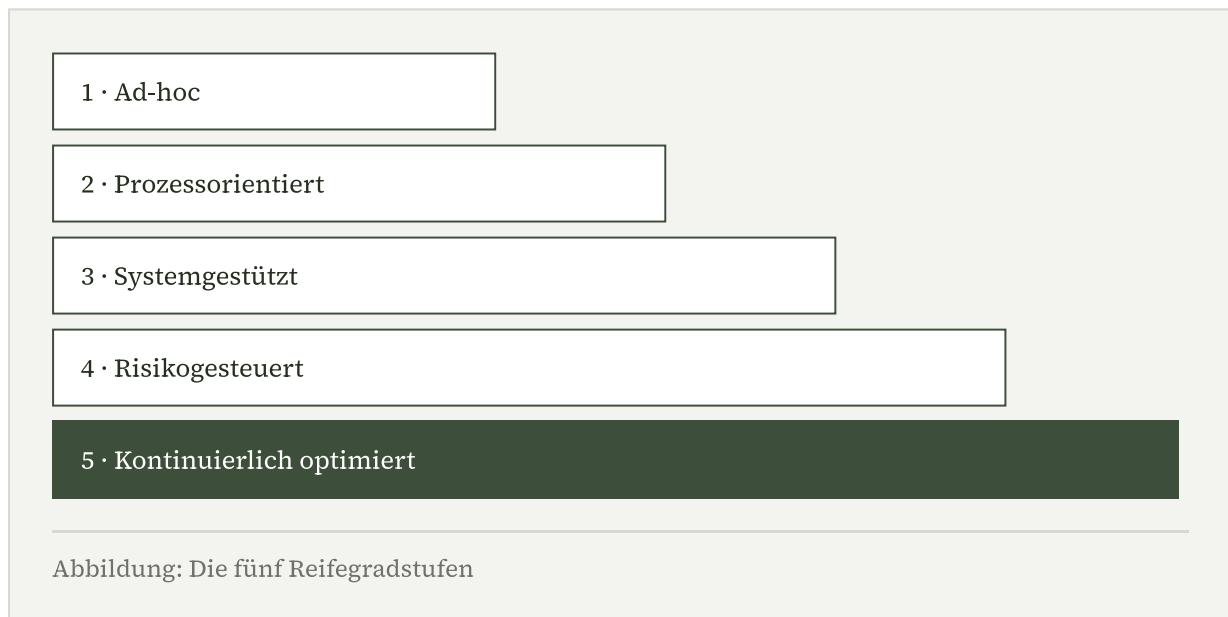
10

Reifegrad & Skalierung

IGA-Reifegradmodelle

Wann ist eine Identity Governance „fertig“? Die Antwort ist einfach: nie. Governance ist kein Endzustand, sondern ein kontinuierlicher Reifeprozess. Viele Unternehmen hoffen, mit der Einführung eines IGA-Tools schlagartig „compliant“ zu werden – in der Praxis entstehen dadurch oft technisch leistungsfähige Systeme mit unausgereiften Governance-Prozessen. Reifegradmodelle helfen, den eigenen Entwicklungsstand realistisch einzuordnen.

Ein praxisnahes Modell umfasst fünf Stufen: **Ad-hoc**, wo Berechtigungen manuell verwaltet werden, Prozesse uneinheitlich und Entscheidungen kaum nachvollziehbar sind; **Prozessorientiert**, wo Joiner-, Mover- und Leaver-Prozesse definiert sind und Standards entstehen, aber überwiegend manuell bleiben; **Systemgestützt**, wo IGA-Werkzeuge Lifecycle, Provisionierung und Kontrollen automatisieren und Governance messbar wird; **Risikogesteuert**, wo sich Kontrollen am tatsächlichen Risiko orientieren und Ausnahmen, privilegierte Zugriffe und Non-Person Identities differenziert gesteuert werden; und **Kontinuierlich optimiert**, wo Governance vollständig in Risikomanagement und IT-Architektur integriert ist und Prozesse datenbasiert weiterentwickelt werden.



MERKSATZ

Reife entsteht nicht durch mehr Technologie, sondern durch bessere Governance.

Der häufigste Fehler besteht darin, den Reifegrad eines Unternehmens mit dem Reifegrad seines Tools zu verwechseln. Ein modernes IGA-System kann Prozesse automatisieren, aber keine fehlende Governance ersetzen – wer unausgereifte Prozesse digitalisiert, erhält lediglich schneller dieselben Fehler. Der eigentliche Reifegrad zeigt sich nicht in der Funktionsvielfalt einer Plattform, sondern in der Qualität der Entscheidungen und der Wirksamkeit der Kontrollen.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Nordfeld Logistik hat vor zwei Jahren ein IGA-Tool eingeführt (Stufe 3, Systemgestützt) – Lifecycle-Prozesse laufen automatisiert. Doch Ausnahmen wie die von Markus Feldt werden noch nicht systematisch nach Risiko unterschieden, und NPIs wie der LogiCore-Sync-Service haben keinen strukturierten Review-Rhythmus. Das zeigt: Das Tool ist reif, die Governance dahinter noch auf dem Weg zu Stufe 4.

Governance vor dem Antrag

„Wir führen jetzt ein IGA-Tool ein, um das Berechtigungs-Chaos zu beenden“ ist ein Satz, der in nahezu jedem IGA-Projekt fällt – und Monate später stellt sich fast immer dieselbe Frage: Warum ist das Chaos immer noch da, nur jetzt digital? Die Antwort liegt selten im Tool, sondern im Zeitpunkt der Entscheidung. Ein Tool kann Anträge digitalisieren, es entscheidet aber nicht, ob ein Antrag fachlich sinnvoll ist – wird jede Anforderung lediglich elektronisch genehmigt, bleibt das Chaos bestehen, nur digital.

Die wichtigste Regel lautet deshalb: Governance beginnt vor dem Antrag. Eine wirk-same IGA-Architektur verlagert die Steuerung nach vorne: **Standardisierung statt Individualisierung**, indem Antragsteller aus definierten Rollen und Berechtigungs-paketen wählen, statt individuelle Berechtigungen zusammenzustellen. **Governance by Design**, indem Genehmigungskriterien wie Qualifikationen oder Sicherheitsfreiga-ben als Regeln definiert werden – werden sie nicht erfüllt, kann der Antrag gar nicht erst gestellt oder automatisch abgelehnt werden. **Policies statt Einzelfallentschei-dungen**, da die Aufgabe des IGA-Teams nicht darin besteht, möglichst viele Anträge zu bearbeiten, sondern die Entscheidungslogik zu gestalten. Und **automatische Vor-abprüfungen**, die SoD-Konflikte oder unzulässige Kombinationen bereits vor dem Genehmigungsprozess erkennen.

Ein Antrag sollte nur noch die Umsetzung einer bereits definierten Governance-Ent-scheidung sein. Je mehr Risiken erst während des Genehmigungsprozesses bewertet werden müssen, desto schwächer ist das Governance-Design.

MERKSATZ

Die beste Governance beschleunigt Anträge, weil sie Entscheidungen bereits im Design getroffen hat.

Operating Rhythm und Risikoorientierung

Die meisten IGA-Programme haben einen beeindruckenden Go-Live-Tag – Rollen definiert, Prozesse abgenommen, das Kickoff-Meeting ein Erfolg. Ein Jahr später weiß in vielen dieser Organisationen niemand mehr genau, wie aktuell die Berechtigungslage wirklich ist. Das liegt selten an der Technologie, sondern daran, dass Governance als Projekt behandelt wurde – mit einem Anfang und einem Ende – statt als kontinuierlicher Steuerungsprozess.

Ein wirksamer Operating Rhythm besteht aus drei Ebenen. Der **operative Rhythmus** (täglich/wöchentlich) umfasst Provisionierung, Deprovisionierung, Anträge und operative Vorfälle. Der **Governance-Rhythmus** (monatlich/quartalsweise) umfasst Reviews, die Bewertung von Governance-KPIs, die Überprüfung von Ausnahmen und die Neubeurteilung von Risiken. Der **strategische Rhythmus** (halbjährlich/jährlich) entwickelt die Governance selbst weiter – Rollenmodelle, Policies und Risikomodelle werden überprüft und angepasst.

Risikoorientierung bestimmt dabei den Takt: Nicht jede Identität benötigt dieselbe Aufmerksamkeit – privilegierte Konten, kritische Non-Person Identities oder besonders sensible Zugriffe sollten deutlich häufiger überprüft werden als Standard-Berechtigungen. Ein risikobasierter Operating Rhythm ersetzt starre „One-Size-Fits-All“-Zyklen durch differenzierte Kontrollintervalle und verhindert, dass Kontrollen nur auf externe Audits reagieren.

Governance ist kein Projektplan, sondern ein dauerhafter Steuerungsrhythmus.

PRAXISBEISPIEL · NORDFELD LOGISTIK

Bei Nordfeld läuft die Zugriffsvergabe in LogiCore täglich (operativer Rhythmus). Einmal im Quartal überprüft Sabine Krüger die Zugriffe ihres Teams (Governance-Rhythmus). Und einmal im Jahr wird das gesamte Rollenmodell mit der Unternehmensleitung überprüft, ob es noch zur Organisation passt (strategischer Rhythmus).

Schnittstellen zu Risk, Audit und Compliance

Ein Audit-Finding zu privilegierten Konten, eine Risikobewertung zu denselben Konten, ein Compliance-Report zur gleichen Anforderung – drei Zahlen, drei Antworten. Risk Management, Compliance und Audit arbeiten in vielen Unternehmen nebeneinander statt miteinander, jede Disziplin stellt eigene Fragen und berichtet in eigene Linien. Die Zugriffsrealität, auf die sich alle drei beziehen, existiert dadurch in drei Versionen statt in einer.

Identity Governance ist die Instanz, die diese Versionen zusammenführt. Für **Risk Management** liefert sie die Fakten, auf denen Bedrohungsbewertungen aufbauen – wie viele privilegierte Konten existieren, wie viele Berechtigungen wurden nicht re-zertifiziert. Für **Compliance** sorgt sie dafür, dass Vorgaben als technische Kontrollen umgesetzt werden, idealerweise als Policy-as-Code statt reiner Dokumentation. Für **Audit** erzeugt sie kontinuierlich belastbare Evidenz, wodurch Audit-Vorbereitung von einer manuellen Sammelaktion zu einem standardisierten Prozess wird.

IGA ist keine isolierte Sicherheitsfunktion, sondern der gemeinsame Daten- und Prozesslieferant für Risk, Compliance und Audit. Je besser diese Schnittstellen funktionieren, desto konsistenter werden Risikoanalysen, Compliance-Nachweise und Auditergebnisse.

Identity Governance verbindet Risiko, Compliance und Audit. Ohne diese Schnittstellen fehlt allen drei die gemeinsame Grundlage.

IGA unter Veränderungsdruck

Die meisten Identity-Governance-Programme scheitern nicht im Rollout – sie scheitern in der ersten großen Veränderung danach. Fusionen, Cloud-Migrationen, neue Geschäftsmodelle, organisatorische Umstrukturierungen gehören heute zum Normalzustand. Eine starre Identity Governance wird in solchen Phasen schnell zum Flaschenhals oder wird durch Ausnahmen und Schattenprozesse einfach umgangen. Der Grund liegt oft in der stillen Annahme, Identity Governance sei eingerichtet und laufe jetzt von selbst.

Vier Prinzipien machen Governance veränderungsfähig. **Fachliche Governance von der Technik entkoppeln:** Business Roles und Governance-Regeln sollten unabhängig von einzelnen Anwendungen oder Plattformen definiert werden, damit Governance auch bei Technologiewechseln stabil bleibt. **Governance by Design:** Veränderungen an Organisation, Rollen oder Prozessen müssen sich kontrolliert auf Berechtigungen auswirken, statt dass Governance nachträglich reagieren muss. **Transformationen bewusst steuern:** Große Veränderungen erzeugen häufig temporäre Berechtigungen und Ausnahmen, die von Beginn an als zeitlich begrenzte Risiken behandelt und konsequent zurückgebaut werden müssen. **Governance früh einbinden:** Identity Governance darf nicht erst nach Architekturentscheidungen ins Spiel kommen, sondern muss bereits bei der Planung neuer Systeme mitgestalten.

Die Reife einer IGA zeigt sich nicht daran, wie gut sie den Normalbetrieb verwaltet, sondern wie sicher sie Veränderungen begleitet. Governance darf den Wandel nicht bremsen – sie muss ihn sicher ermöglichen.

Eine gute Identity Governance schützt nicht vor Veränderung. Sie sorgt dafür, dass Veränderung kontrolliert stattfinden kann.

Grenze des Modells: Reifegradmodelle wie das in diesem Kapitel beschriebene suggerieren einen linearen Fortschritt von Stufe 1 zu Stufe 5. In der Praxis verläuft Reife selten so gleichmäßig – eine Organisation kann bei privilegierten Zugriffen bereits auf Stufe 4 stehen, während NPI-Governance noch bei Stufe 2 verharret, weil sie historisch weniger Aufmerksamkeit bekam. Das Modell bleibt als Orientierung wertvoll, sollte aber nicht als einzelne Gesamt-Kennzahl missverstanden werden, sondern getrennt nach Domänen (menschliche Identitäten, privilegierte Zugriffe, NPIs) betrachtet werden.

Verdichtung: Die zentralen Governance-Merksätze

Nach der gesamten bisherigen Betrachtung bleibt eine unbequeme Frage offen: Wie viele der beschriebenen Prinzipien überleben eigentlich den ersten Kontakt mit der eigenen Systemlandschaft? Viele Governance-Sätze klingen in der Theorie selbstverständlich – Ownership, Legitimation, Least Privilege, kaum jemand widerspricht ihnen offen. Das eigentliche Problem zeigt sich erst im Betrieb: Sätze, die im Deck stehen, aber im Reifegrad-Assessment, im Access Review oder im nächsten Audit nicht mehr auffindbar sind.

Dieser Text war nie als Sammlung von Merksätzen gedacht. Es war der Versuch, ein zusammenhängendes Denkmodell zu entwickeln – zehn Leitsätze, die zusammen mehr ergeben als die Summe ihrer Teile:

Identität ist der neue Perimeter – wer Zugriffe steuert, steuert Risiken. Jede Berechtigung braucht eine Legitimation – Entscheidung vor Provisionierung. Governance geht Automatisierung voraus – schlechte Prozesse werden durch Automatisierung nicht besser. Ownership ist unverzichtbar – jedes Governance-Objekt braucht einen verantwortlichen Owner. Drift ist der Normalzustand – Abweichungen müssen kontinuierlich erkannt und korrigiert werden. Least Privilege ist ein Dauerprinzip – zusätzliche Rechte benötigen Begründung, Begrenzung und Kontrolle. Evidenz schlägt Logfiles – Auditfähigkeit entsteht durch nachvollziehbare Entscheidungen. Governance braucht Rhythmus – regelmäßige Reviews und Verbesserungen halten das System wirksam. Technische Identitäten sind echte Identitäten – NPIs benötigen Lifecycle, Ownership und Kontrolle. Governance ermöglicht Veränderung – gute Governance macht Wandel beherrschbar.

MERKSATZ

Identity Governance ist kein Projekt mit Enddatum. Es ist ein Betriebsmodell, das jeden Tag neu bewiesen werden muss.

Aus Governance-Sicht sind das keine zehn getrennten Empfehlungen, sondern ein einziges Betriebsmodell, zehnmal aus unterschiedlichen Blickwinkeln betrachtet: Entscheidung → Umsetzung → Kontrolle → Korrektur. Jeder Leitsatz beschreibt, was passiert, wenn eine dieser vier Stationen fehlt oder schwach ausgeprägt ist.

Abkürzungsverzeichnis

ABAC	Attribute-Based Access Control	JEA	Just-Enough Access
		JIT	Just-in-Time
API	Application Programming Interface	KPI	Key Performance Indicator
		MVE	Minimum Viable Evidence
CRM	Customer Relationship Management	NPI	Non-Person Identity
		RACI	Responsible, Accountable, Consulted, Informed
ERP	Enterprise Resource Planning		
HR	Human Resources	RBAC	Role-Based Access Control
IAM	Identity and Access Management	ReBAC	Relationship-Based Access Control
IGA	Identity Governance and Administration	RPA	Robotic Process Automation
		SoD	Segregation of Duties

Glossar

Die Kernbegriffe des Textes in alphabetischer Folge. Die Kapitelangabe verweist auf die Stelle, an der der Begriff fachlich entwickelt wird.

ABAC Kap. 6

Attribute-Based Access Control. Zugriffsentscheidungen ergeben sich aus Attributen von Person, Objekt und Kontext, ausgewertet durch eine Regel. Setzt voraus, dass jedes verwendete Attribut einen Owner und eine gesicherte Qualität hat.

Accountability Kap. 1

Die persönliche Einstandspflicht für eine Entscheidung. Sie ist nicht delegierbar und nicht mit einer Genehmigung gleichzusetzen: Wer freigibt, muss nicht zwingend derjenige sein, der für die Entscheidung geradesteht.

Activation Kap. 7

Die tatsächliche Aktivierung eines Zugriffs für einen konkreten Bedarf und ein begrenztes Zeitfenster. Zu unterscheiden von Eligibility, der bloßen Berechtigungsfähigkeit.

Actual State Kap. 4

Der real im Zielsystem vorhandene Zugriff. Die Abweichung zum Desired State ist Drift.

Attestierung Kap. 9

Die formale Bestätigung durch einen Verantwortlichen, dass ein Zugriff weiterhin fachlich begründet ist. Wirksam nur, wenn Nichtreaktion nicht als Bestätigung gilt.

Attribute Owner Kap. 6

Verantwortlicher für Bedeutung, Qualität und Pflege eines Attributs, das in Zugriffsentscheidungen einfließt. Ohne diese Rolle wird ABAC unkontrollierbar.

Break-Glass Kap. 7

Ein vorab definierter Notzugang für Ausnahmesituationen. Kontrolliert nur dann, wenn seine Nutzung protokolliert, alarmiert und nachträglich aufgearbeitet wird.

Control Owner Kap. 3

Verantwortlicher für die Wirksamkeit einer Kontrolle, nicht für den Zugriff selbst. Er beantwortet, ob die Kontrolle funktioniert, nicht ob ein einzelner Zugriff richtig ist.

Decision Record Kap. 2

Das eigenständige Artefakt einer Zugriffsentscheidung: wer entschieden hat, worüber, wann, auf welcher Grundlage und mit welcher Gültigkeit. Grundlage jeder späteren Nachweisführung.

Delegation Kap. 3

Die Weitergabe der operativen Durchführung einer Entscheidung. Die Accountability bleibt beim ursprünglichen Owner.

Desired State Kap. 4

Der fachlich begründete Sollzustand eines Zugriffs, abgeleitet aus Rolle, Aufgabe oder Policy. Referenz für jeden Soll-Ist-Abgleich.

Detektion Kap. 9

Kontrolltyp, der eine bereits eingetretene Abweichung erkennt. Wirkt nur im Zusammenspiel mit Korrektur.

Drift Kap. 4

Das Auseinanderlaufen von Desired State und Actual State im Zeitverlauf. Entsteht nicht durch Fehler, sondern durch fehlende fortlaufende Steuerung.

Eligibility Kap. 7

Die Berechtigungsfähigkeit einer Identität für einen Zugriff, ohne dass dieser aktiv ist. Trennt die fachliche Prüfung von der tatsächlichen Nutzung.

Entitlement Creep Kap. 4

Das schleichende Anwachsen von Berechtigungen über die Zeit, typischerweise weil bei Aufgabenwechseln Rechte hinzukommen, aber keine entzogen werden.

Evidenz Kap. 9

Der Nachweis, dass ein Zugriff begründet, verantwortet und kontrolliert war. Abzugrenzen von Logs: Logs zeigen, was geschah, Evidenz zeigt, warum es zulässig war.

Exception Debt Kap. 5

Der aufgelaufene Bestand an Ausnahmen, die nie zurückgebaut wurden. Wirkt wie technische Schuld und wird zum eigenständigen Steuerungsproblem.

Exception Record Kap. 5

Die dokumentierte Ausnahme als Governance-Objekt, mit Begründung, Kompensationskontrolle, Owner und verbindlicher Befristung.

Fail-safe Default Kap. 9

Das Prinzip, dass bei fehlender Reaktion der sichere Zustand eintritt, also der Entzug und nicht die Verlängerung.

Findings Kap. 9

Von Audit oder Kontrolle festgestellte Abweichungen. Professioneller Umgang heißt: Ursache klären, nicht den Einzelfall bereinigen.

Governance Vorwort

Die Summe aller qualifizierten Entscheidungen über Zugriff, einschließlich ihrer Verantwortung, Kontrolle und Nachweisbarkeit.

Governance by Design Kap. 10

Die Verankerung von Governance-Anforderungen vor dem Antrag, also bereits in Systemanbindung, Rollenentwurf und Prozessdesign.

Governance-Debt-Index Kap. 9

Kennzahl für den aufgelaufenen Steuerungsrückstand, etwa aus offenen Ausnahmen, unklarem Ownership und überfälligen Reviews.

Governance-KPI Kap. 9

Kennzahl zur Steuerung der Governance selbst, nicht des Betriebs. Braucht explizite Ausschlusskriterien für unreife Systemanbindungen.

Governance-Objekt Kap. 1

Ein Gegenstand der Steuerung, der einen Owner, einen definierten Zustand und einen Prüfzyklus besitzt. Dazu zählen Identitäten, Rollen, Verträge, Policies und Ausnahmen.

JEA Kap. 7

Just-Enough Access. Die inhaltliche Begrenzung eines Zugriffs auf das für die Aufgabe Erforderliche, komplementär zur zeitlichen Begrenzung durch JIT.

JIT Kap. 7

Just-in-Time. Die zeitliche Begrenzung eines Zugriffs auf das Fenster des konkreten Bedarfs mit automatischem Entzug danach.

Kompensationskontrolle Kap. 5

Eine ersatzweise Kontrolle, die das Risiko einer genehmigten Ausnahme so lange abfedert, wie die Ausnahme besteht.

Kontrollkette Kap. 9

Die durchgängige Abfolge von Entscheidung, Umsetzung, Überwachung und Entzug. Reißt sie an einer Stelle, ist der Nachweis für die gesamte Kette entwertet.

Korrektur Kap. 9

Kontrolltyp, der eine erkannte Abweichung beseitigt und den Sollzustand wiederherstellt.

Least Privilege Kap. 6

Das Prinzip, nur die für die Aufgabe erforderlichen Rechte zu vergeben. Gilt für menschliche und technische Identitäten gleichermaßen.

Legitimation Vorwort

Die fachliche Begründung eines Zugriffs aus Rolle, Aufgabe oder Policy. Ohne Legitimation gibt es keine Entscheidung, sondern nur eine Bereitstellung.

Lifecycle Kap. 4

Die fortlaufende Zustandssteuerung einer Identität und ihrer Zugriffe, nicht eine Kette einzelner Ereignisse.

MVE Kap. 1

Minimum Viable Evidence. Der kleinste Datensatz, der eine Entscheidung nachvollziehbar macht: wer, was, wann, warum.

NPI Kap. 8

Non-Person Identity. Technische Identität wie Servicekonto, API-Schlüssel oder Bot. Unterliegt denselben Governance-Anforderungen wie eine Person.

NPI-Taxonomie Kap. 8

Die strukturierte Klassifizierung technischer Identitäten nach Typ, Zweck und Risiko. Voraussetzung für differenzierte Steuerung.

Onboarding-Gate Kap. 9

Die Prüfung, ob ein System die Governance-Mindestanforderungen erfüllt, bevor es angebunden wird.

Operating Rhythm Kap. 10

Der verbindliche Takt wiederkehrender Governance-Aufgaben wie Reviews, Rezertifizierungen und Ausnahmerückbau.

Ownership Kap. 3

Die zugewiesene fachliche Verantwortung für ein Governance-Objekt. Zu unterscheiden von der operativen Durchführung.

Policy-to-Decision Trace Kap. 2

Die lückenlose Verbindung von Regel über auslösendes Ereignis und Entscheidung bis zum wirksamen Zugriff im Zielsystem.

Prävention Kap. 9

Kontrolltyp, der eine Abweichung von vornherein verhindert.

Privileged Access Kap. 7

Zugriff mit erhöhtem Schadenspotenzial. Eigener Kontrollraum mit verschärften Anforderungen an Befristung, Begründung und Nachvollziehbarkeit.

Privilegien-Eskalation Kap. 7

Das unbemerkte Anwachsen von Rechten, häufig über den Änderungspfad statt über den Antragsweg.

RACI Kap. 3

Zuordnungsschema für Verantwortung: accountable, responsible, consulted, informed. In der IGA vor allem zur Trennung von Entscheidung und Durchführung.

RBAC Kap. 6

Role-Based Access Control. Die Rolle als fachliches Bündel von Rechten und als Träger der Begründung. Fundament der meisten Zugriffsmodelle.

ReBAC Kap. 6

Relationship-Based Access Control. Die Beziehung zwischen Identität und Objekt entscheidet über den Zugriff. Skaliert in großen Organisationen nur begrenzt.

Reconciliation Kap. 4

Der systematische Abgleich von Desired State und Actual State, top-down aus dem Sollmodell oder bottom-up aus dem Zielsystem.

Reifegradmodell Kap. 10

Die stufenweise Einordnung der Governance-Fähigkeit von ad-hoc bis kontinuierlich optimiert. Beschreibt einen Prozess, keinen Endzustand.

Rollenmodell Kap. 6

Die Übersetzung organisatorischer Komplexität in eine steuerbare Zugriffsstruktur. Seine Grenze liegt dort, wo Sonderfälle die Modellpflege übersteigen.

Secret Rotation Kap. 8

Der geplante, wiederkehrende Wechsel von Zugangsgeheimnissen technischer Identitäten.

Segregation of Duties Kap. 3

Die Trennung unvereinbarer Aufgaben auf verschiedene Personen, damit niemand einen Vorgang allein anstoßen und genehmigen kann.

Session Accountability Kap. 7

Die Zurechenbarkeit einer privilegierten Sitzung zu einer natürlichen Person, auch bei Nutzung geteilter Konten.

Sicherheitsvorfall Kap. 7

Ein eigener Lifecycle-Zustand, der Zugriffe unabhängig vom regulären Prozess unverzüglich aussetzt.

Task-/Scope-basierte Zugriffssteuerung Kap. 6

Zuschnitt des Zugriffs auf eine konkrete Aufgabe oder einen abgegrenzten Datenbereich statt auf eine dauerhafte Rolle.

Timeboxing Kap. 2

Die verbindliche Befristung einer Entscheidung. Nach Ablauf endet die Gültigkeit, statt stillschweigend fortzubestehen.

Veränderungsfähigkeit Kap. 10

Die Eigenschaft einer Governance, Reorganisationen, neue Systeme und geänderte Risiken ohne Kontrollverlust zu verkraften.

Vertrag Kap. 4

Das Beschäftigungs- oder Mandatsverhältnis als eigenständiges Governance-Objekt. Sein Ende ist ein verbindlicher Trigger für den Zugriffsentzug.

Merksätze im Überblick

VORWORT · VORWORT: WARUM DIESER TEXT

Governance beginnt dort, wo Verantwortung nicht mehr delegiert wird, sondern getragen werden muss.

Governance ist kein technisches Nebenprodukt, sondern die bewusste Steuerung von Entscheidungsrechten.

KAPITEL 1 · ORIENTIERUNG UND GRUNDLOGIK

Ein Ticket dokumentiert eine Ausführung – erst die Begründung macht daraus eine steuerbare Entscheidung.

Governance ist nicht der Besitz von Berechtigungen, sondern die Beherrschung ihrer Gültigkeit.

Governance ist die Summe aller qualifizierten Entscheidungen entlang des Business-Lifecycles.

KAPITEL 2 · DECISION GOVERNANCE

Ein Ticket dokumentiert den Prozess. Ein Decision Record dokumentiert die Legitimation.

Governance ist erst dann wirksam, wenn jede Entscheidung ihre Wurzeln in einer spezifischen Policy-Version nachweisen kann.

Gute Governance misst nicht, wie viel gearbeitet wurde, sondern wie sicher und regelkonform die Zustände im System sind.

KAPITEL 3 · OWNERSHIP UND VERANTWORTUNG

Die IT verantwortet das Werkzeug, das Business verantwortet die Entscheidung. Wer das vermischt, schafft Verantwortungs-Vakuum.

Wer alles verantwortet, verantwortet am Ende nichts. Ownership braucht eine klare Domäne: System, Logik oder Inhalt.

Governance funktioniert nur, wenn die Person, die Rechenschaft trägt, auch die Person ist, die steuert.

KAPITEL 4 · LIFECYCLE UND DRIFT

Identitätsmanagement ist kein Domino-Effekt, sondern eine dauerhafte Synchronisation. Der Zugriff folgt dem Status, nicht dem Ticket.

Ein interner Wechsel ist kein Domino-Effekt, sondern ein Neustart der Legitimationskette.

Zugriff ist kein dauerhaftes Eigentum, sondern eine temporäre, zweckgebundene Delegation.

KAPITEL 5 · AUSNAHMEN UND SONDERFÄLLE

Eine Ausnahme ist kein Governance-Versagen, sondern ein bewusst zu steuerndes Ereignis.

Exception Debt ist keine technische, sondern eine Governance-Schuld.

Ein Sonderfall ohne Abbaupfad ist kein temporäres Problem, sondern eine dauerhafte Risikoerhöhung.

KAPITEL 6 · ZUGRIFFSMODELLE

Ein Modell ist die Landkarte, nicht das Gelände.

RBAC ist die gemeinsame Sprache zwischen Business und IT – solange nicht jede Ausnahme zu einem eigenen Dialekt wird.

ABAC bietet enorme Flexibilität – aber nur, wenn die zugrunde liegenden Daten konsistent, aktuell und kontrolliert sind.

Technik folgt der Governance. Ein Modellwechsel ohne klare Prozesse verschiebt das Chaos lediglich an eine andere Stelle.

KAPITEL 7 · PRIVILEGED ACCESS

Privilegierter Zugriff ist kein technisches Privileg, sondern ein kritisches Governance-Objekt mit besonders hohen Anforderungen an Steuerung und Nachweis.

Eligibility beschreibt das Potenzial. Activation dokumentiert die tatsächliche Nutzung.

Die Aktivierung eines Notfallzugriffs ist kein Ausstieg aus der Governance, sondern der Beginn eines eigenständigen Hochrisiko-Prozesses.

KAPITEL 8 · NON-PERSON IDENTITIES

Eine Non-Person Identity ist technisch autonom, darf aber organisatorisch niemals eigentümerlos sein.

Systeme handeln automatisiert. Verantwortung bleibt immer menschlich.

Dauerhafte Administratorrechte für technische Identitäten sind selten ein Zeichen stabiler Systeme. Sie sind meist ein Hinweis auf fehlende Governance.

KAPITEL 9 · EVIDENZ & AUDIT

Wer die Kontrollkette beherrscht, beherrscht das Risiko. Wer sie unterbricht, schafft Berechtigungs-Drift.

Logs zeigen, was passiert ist. Evidenz belegt, warum es passieren durfte. Erst beides zusammen macht Governance auditfähig.

Ein Finding zeigt nicht nur einen Fehler. Es zeigt, wo Governance verbessert werden muss.

KAPITEL 10 · REIFEGRAD & SKALIERUNG

Reife entsteht nicht durch mehr Technologie, sondern durch bessere Governance.

Die beste Governance beschleunigt Anträge, weil sie Entscheidungen bereits im Design getroffen hat.

Identity Governance ist kein Projekt mit Enddatum. Es ist ein Betriebsmodell, das jeden Tag neu bewiesen werden muss.

Weiterführende Rahmenwerke

Dieser Text zitiert bewusst keine externen Quellen im Fließtext – die Inhalte entstanden aus eigener Praxiserfahrung und sollen herstellerneutral sowie unabhängig von einzelnen Frameworks lesbar sein. Für Leserinnen und Leser, die einzelne Themen vertiefen möchten, sind nachfolgend etablierte Standards und Rahmenwerke aufgeführt, die inhaltlich an die im Text behandelten Prinzipien angrenzen. Sie waren nicht Grundlage dieses Textes, sondern eine Einordnungshilfe für die weitere Lektüre.

International Organization for Standardization / International Electrotechnical Commission: *ISO/IEC 27001:2022 – Information security, cybersecurity and privacy protection – Information security management systems – Requirements*. Genf: ISO, 2022.

National Institute of Standards and Technology: *NIST Special Publication 800-63-4 – Digital Identity Guidelines*. Gaithersburg, MD: NIST, 2025.

National Institute of Standards and Technology: *NIST Special Publication 800-53, Revision 5 – Security and Privacy Controls for Information Systems and Organizations*. Gaithersburg, MD: NIST, 2020.

ISACA: *COBIT 2019 Framework – Governance and Management Objectives*. Schaumburg, IL: ISACA, 2019.

Über den Autor

Philipp Beermann ist Senior IGA Consultant bei der IS4IT GmbH und begleitet Organisationen bei der Einführung und Weiterentwicklung von Identity & Access Management sowie Identity-Governance-Strukturen. Sein Fokus liegt auf der strategischen und konzeptionellen Seite dieser Disziplin – Governance, Methoden, Prozesse und Compliance, nicht die technische Implementierung einzelner Systeme.

Vor seiner Tätigkeit bei IS4IT war er mehrere Jahre als IAM-Consultant bei Beta Systems Software tätig, wo er in zahlreichen Kundenprojekten praktische Erfahrung mit Identity-Governance-Architekturen sammelte.

Bekannt wurde seine Arbeit durch eine 70-teilige Beitragsserie zu Identity Governance auf LinkedIn, in der er die zentralen Prinzipien moderner IGA – von der Entscheidungslogik über Ownership bis zur Evidenzführung – systematisch aufgearbeitet hat. Aus dieser Serie ist auch dieser Text entstanden.