

ONE-PERSON IT OPERATIONS HANDBOOK

# ひとり情シス大全

ひとり情シスが会社を止めないための実務

版

2026年4月版

発行

haya株式会社

# 目次

## 前付け

はじめに	4
本書の読み方	5

## 第I部 全体像と運営の土台

第1章 ひとり情シスとは何を担う仕事か	8
第2章 最初に整えるべき運営の土台	19
第3章 ひとり情シスの仕事を回す基本原則	30

## 第II部 経営、統制、調達、導入

第4章 経営とITガバナンス	42
第5章 IT予算、契約、調達、ライセンス	54
第6章 ベンダー管理と外部委託	68
第7章 要件定義、導入プロジェクト、変更管理	83

## 第III部 日常運用の基盤

第8章 アカウント、認証、権限管理	99
第9章 入社、異動、退職とアカウント運用フロー	116
第10章 PC、スマホ、端末、資産管理	133
第11章 ネットワーク、拠点、テレワーク	149
第12章 SaaS、クラウド、業務アプリ管理	163
第13章 メール、チャット、ファイル共有の運用	175
第14章 サーバー、Web、ドメイン、DNS、証明書	187
第15章 データ保護、バックアップ、ログ、保持	201
第16章 ヘルプデスク、依頼管理、ナレッジ整備	214

## 第IV部 セキュリティ、法務、監査

第17章 セキュリティ基礎対策	226
第18章 脆弱性管理、パッチ管理、構成管理	237
第19章 個人情報保護、社内規程、監査対応	248
第20章 教育、訓練、委託先を含むセキュリティ運用	261

## 第V部 障害、インシデント、事業継続

第21章 障害対応とインシデント管理	274
第22章 ランサムウェア、情報漏えい、不正アクセス対応	287
第23章 BCP、災害対応、復旧訓練	301

## 第VI部 改善、自動化、AI、引き継ぎ

第24章 自動化、スクリプト、業務改善	317
第25章 可視化、指標、コスト最適化	330
第26章 生成 AI 活用と AI ガバナンス	347
第27章 ひとり情シスの働き方、引き継ぎ、次の一手	362

## 付録と巻末

付録A 台帳ひな型	375
付録B 入社、異動、退職チェックリスト	381
付録C 障害、インシデント初動チェックリスト	386
奥付	391

# はじめに

---

ひとり情シスという言葉は、肩書きより状態を表していることが多い。専任の情報システム担当であっても、総務や管理部門との兼務であっても、現場で起きていることは似ている。PC の準備、アカウント運用、SaaS 管理、障害対応、セキュリティ対策、契約更新、ベンダー調整、経営への説明まで、会社の IT に関わる実務が一人に集まりやすい。

この本は、そうした状態に置かれた人が、場当たり対応だけで毎日を終わらせないための本である。単なるツール紹介や気合の話ではなく、会社を止めないために何を知り、何を決め、どう回すかを、できるだけ体系立てて整理した。

本書で重視したのは、ひとり情シスが必要になりうる知識をできるだけ漏らさないことと実務で使える順序に並べることである。そのため、日常運用だけでなく、調達、契約、外部委託、個人情報保護、障害対応、ランサムウェア、BCP、生成 AI ガバナンス、引き継ぎまで含めた。忙しい現場ほど、問題が起きてから必要な知識を探しに行く余裕がないからである。

ただし、すべてを一度に整える必要はない。本書は全部やるべきこと一覧として読むより、いま最も弱い場所を見つけて次の一手を決める本として使った方が役に立つ。小さな会社では、網羅性より順序の方が大事である。

なお、製品名、管理画面名、制度運用、公式ガイドの版は時間とともに変わる。本文では、変わりにくい原則と、変わりやすい実務の線をなるべく分けて書いたが、導入や運用の直前には公式情報の再確認を前提にしてほしい。

本書が目指す到達点は、何でも知っている担当者を育てることではない。自分しか回せない状態を減らし、会社に知識と運営が残る形へ近づけることである。

# 本書の読み方

---

本書は、第1章から順に読んでもよいが、実務では **いま困っている場所** から読む方が使いやすい。次の目安で入ると進めやすい。

## まず土台を作りたい時

- 第1章から第3章で、ひとり情シスの守備範囲、運営の土台、仕事の回し方をつかむ
- 第4章から第7章で、経営、契約、外部委託、導入変更の考え方を固める

## 日常運用を整えたい時

- 第8章から第16章で、アカウント、入退社、端末、ネットワーク、SaaS、共有、サーバーとドメイン、バックアップ、依頼管理を確認する
- 台帳、申請、窓口、標準手順を先に整えると、後半の章も回しやすくなる

## セキュリティや有事対応を見直したい時

- 第17章から第23章で、基礎対策、脆弱性管理、個人情報保護、教育、障害、重大事故、BCPを確認する
- 事故が起きてから読むのではなく、平時に **どこまで準備できているか** を点検する読み方が向いている

## 改善と次年度計画へつなげたい時

- 第24章から第27章で、自動化、可視化、生成 AI、引き継ぎと次の一手を確認する
- 忙しさを減らすには、便利なツール導入だけでなく、運用の持ち方を変える必要がある

## 読み進める時の見方

- 第3章以降の各章では、まず全体像をつかみ、その後に **通常時の例**と、**崩れた時の例**を見る
- 章末の **最低限ここまではやる** は、最初の実行ラインとして使う
- **今日、今週、後でやること** は、その章を読んだ直後の行動へ落とすための欄である

## この本の前提

- 小規模から中規模の組織で、ひとりまたは少人数で IT 運営を担う読者を主対象にしている
- 特定製品の完全操作マニュアルではなく、運営判断と実務の原則を扱う
- 会社ごとに規模、業種、法規制、利用製品は違うため、本文は **最小限の現実解**を基準にしている

第1部

# 全体像と運営の土台

守備範囲と仕事の型を最初に固める。

第1章～第3章

## 第1章

# ひとり情シスとは何を担う仕事か

---

朝、出社すると、新入社員用のノート PC が届いていないことに気づく。先にアカウントだけは発行しておきたいので、クラウドの管理画面を開いてユーザーを追加する。その途中で、営業部から「商談先で Wi-Fi につながらないと困るので、今日中に会議室の無線を見てほしい」と連絡が入る。昼前には、退職者のメールを誰に引き継ぐか総務から相談が来る。午後は、月末で切れる SaaS の契約更新を確認し、夕方には取引先から届いたセキュリティチェックシートに回答する。

これが、ひとり情シスの一日として特別に誇張された光景かというと、そうではない。むしろ、少し静かな日の方に入る。

ひとり情シスの仕事は、こうした雑多な作業の寄せ集めに見えやすい。だからこそ、自分でも周囲でも、「何をやっている人なのか」が曖昧になりやすい。すると、仕事の優先順位が定まらず、引き継ぎもできず、事故が起きたときだけ存在が強く意識される状態になる。

この章の目的は、その曖昧さをなくすことにある。ひとり情シスとは何を担う仕事なのかを、最初に言葉で定義する。ここが曖昧なままだと、後の章で扱う端末管理も、アカウント管理も、セキュリティも、ただの項目集になってしまう。逆に、ここで守備範囲の地図を持てれば、自分に足りないものと、今すぐ整えるべきものが見えやすくなる。

ここで一つ安心してよいことがある。NIST が 2024年2月26日に公開した **NIST Cybersecurity Framework 2.0: Small Business Quick-Start Guide** は、**modest or no cybersecurity plans in place** の SMB を対象にしている。

つまり、小規模組織は最初から完成した体制を前提にされていない。第1章で必要なのは、完璧な制度を覚えることではなく、自分がどんな役割の束を抱えているかを見えるようにすることである。

## ひとり情シスは役職名ではなく、配置の名前である

まず押さえてほしいのは、ひとり情シスは厳密な職種名ではないということだ。会社によって肩書きは、総務、管理部、経営企画、店舗運営、社長室、情報システム担当などさまざまである。実際には、正式な「情報システム部」がなくても、誰か一人が IT に関する判断、調整、設定、問い合わせ、事故対応を引き受けていれば、その人は実質的にひとり情シスである。

この状態を理解するには、「ひとり情シスは一つの専門職ではなく、複数の役割が一人に集まった状態だ」と考える方が分かりやすい。本来なら、会社の IT には、問い合わせ対応をする役割、アカウントを管理する役割、端末やネットワークを運用する役割、ベンダーを調整する役割、セキュリティ事故に備える役割などがある。規模の大きい会社では、これらは複数人や複数部署に分かれる。しかし、中小企業や少人数組織では、そうした役割が分かれず、一人に集まりやすい。

NIST の NICE Framework は、サイバーセキュリティの仕事を共通言語で分けて考える枠組みであり、たとえば **Technical Support Systems Administration** **Network Operations** のように、本来別の役割を分けて扱う。ひとり情シスの現場では、こうした役割がまとめて一人へ寄ってきやすい。

たとえば、ひとり情シスの一日は、次のように複数ロールの仕事が混ざりやすい。

本来は分かれやすい役割	ひとり情シスで起きがちな実務
技術サポート	印刷できない、Wi-Fiが不安定、PCが重いといった一次対応
システム管理	アカウント発行、権限設定、共有フォルダ管理、SaaS管理
ネットワーク運用	Wi-Fi、VPN、回線、拠点接続、会議室機器の維持
セキュリティ対応	MFA適用、不審メール対応、退職者停止、事故初動
ベンダー調整	契約更新、保守問い合わせ、見積比較、責任分界の確認

2024年2月に公開された NIST Cybersecurity Framework 2.0 は、サイバーセキュリティを **Govern Identify Protect Detect Respond Recover** の6つの機能で整理した。ここで重要なのは、守るだけでなく、統治し、把握し、検知し、対応し、復旧するまでが一つの流れとして扱われていることだ。ひとり情シスの実務も同じである。端末を配るだけでも、終わりではない。どの端末を誰が使い、どの権限が与えられ、異常が起きたときにどう対応し、退職時にどう回収し、記録をどう残すかまでを考えなければならない。

NIST の FAQ では、CSF は IT 部門だけでなく、組織全体の ERM の一部として使う時に最も効果的だとされている。さらに、executive leadership から individual operating units までに当てはまるとされている。つまり、ひとり情シスの仕事は現場作業に見えても、本来は経営判断や取引先対応と切り離せない。

本書では、このように複数の役割が一人または少人数へ集中している状態を **ひとり情シス** と呼ぶ。つまり、ひとり情シスとは「PCに詳しい人」ではない。会社の IT サービスを、限られた人員で何とか運用し続けるために、本来分かれている複数の役割をつないでいる人のことだ。

## ひとり情シスの守備範囲は、想像よりずっと広い

ひとり情シスの守備範囲は、次の七つに大きく分けて考えると整理しやすい。

第一に、問い合わせ対応と依頼対応である。パスワード再設定、ソフト導入、アカウント発行、印刷できない、VPN につながらない、といった日々の困りごとを受け入れる入口になる。周囲から最も見えやすい仕事なので、ここだけがひとり情シスの仕事だと思われやすい。

第二に、アカウント、認証、権限の管理である。入社時の発行、異動時の変更、退職時の停止、MFA の適用、共有アカウントの整理、管理者権限の分離などが入る。最近の IT 環境では、端末やネットワークそのものより、まず ID と認証が入口になる。ここが弱いと、多くの対策が意味を失う。

第三に、端末、資産、ネットワークの管理である。PC やスマートフォンのキitting、配布、回収、交換、紛失対応、Wi-Fi や回線、会議室機器や複合機まで、業務に使う物理的な基盤を支える。

第四に、SaaS、クラウド、データの管理である。今の会社では、業務の多くがクラウド上で動いている。メール、ファイル共有、チャット、勤怠、経費、営業支援、顧客管理などがそれぞれ別の SaaS に分かれていてもおかしくない。ひとり情シスは、それらを個別に管理するだけでなく、アカウント連携、外部共有、データの保存先、解約時のデータ返却まで見なければならない。

第五に、セキュリティ、インシデント対応、復旧である。IPA の中小企業向けガイドラインでも、経営者向けの指針と実務担当者向けの手順の両方が必要だとされている。ウイルス対策ソフトを入れるだけで終わる話ではない。怪しいメールをどう防ぐか、管理者権限をどう分けるか、事故が起きたら誰に何を連絡するか、復旧後に何を見直すかまで含む。

第六に、予算、契約、ベンダー、導入である。現場はここを情シスの仕事と思っていないことが多い。しかし、何を買うか、どの契約で入れるか、誰に委託するか、障害時の責任分界をどうするかは、運用品質を大きく左右する。安く入れたつもりが、後から運用負荷が膨らむことは珍しくない。

第七に、規程、監査、教育、改善である。利用ルール、台帳、申請フロー、監査証跡、教育、ナレッジ整備、業務改善、自動化などがここに入る。普段は目立たないが、これがないと、すべての仕事が個人依存になる。

七つの領域は、覚えるためだけでなく、**何が抜けると何が起きるか**を見えるようにするために使うとよい。

領域	よくある作業	見落とすと起きやすいこと
問い合わせ対応	一次受付、切り分け、回答	依頼が散り、未対応や重複対応が起きる
アカウント、認証、権限	入退社、MFA、権限変更	退職者アカウント残留、権限過多が起きる
端末、資産、ネットワーク	キitting、配布、回収、Wi-Fi管理	端末所在不明、接続障害、交換漏れが起きる
SaaS、クラウド、データ	契約、外部共有、データ保全、解約対応	野良 SaaS、保存先不明、返却漏れが起きる
セキュリティ、復旧	不審メール対応、初動、バックアップ確認	被害拡大、復旧不能、報告遅延が起きる
予算、契約、ベンダー、導入	見積比較、更新判断、保守依頼	期限切れ、責任分界不明、無駄な契約が起きる
規程、監査、教育、改善	ルール化、棚卸し、教育、ナレッジ化	属人化、再発、説明不能が起きる

この七つの領域は、別々に存在しているわけではない。たとえば退職者対応を考えれば、アカウント停止だけでなく、メール引き継ぎ、端末回収、ライセンス回収、データ保全、管理者権限の棚卸し、記録保存までつながる。ひとり情シスの難しさは、一つひとつの技術が高度だからではない。別の領域だと思われている仕事が、実際には一本の業務としてつながっていることにある。

## ひとり情シスの仕事は、会社の事業継続と信用に直結する

ひとり情シスの仕事は、裏方の雑務だと軽く見られやすい。だが、実態は逆である。会社の仕事は、メール、認証、端末、ネットワーク、ファイル共有、SaaS が動くことを前提に回っている。これらのどれか一つが止まるだけで、営業、受発注、請求、顧客対応、人事手続きまで広く影響を受ける。

さらに、影響は社内だけにとどまらない。IPA が 2025年5月27日に公開した **2024年度 中小企業における情報セキュリティ対策に関する実態調査** では、セキュリティ体制を整備している企業の約6割が、発注元からの要請でサイバーセキュリティ対策を行ったことが取引につながったと回答している。情報システムの整備は、単なる守りではなく、取引上の信頼にもつながる。

また、IPA が 2025年2月14日に公開した速報版では、2023年度にサイバーインシデント被害を受けたと回答した企業 975 社のうち、全体の約3割の「特に無し」を除くと、約7割が取引先に影響があったと回答している。規模が小さいから狙われない、専任情シスがないから仕方がない、という理屈は通りにくい。

情シスの役割は守りだけでもない。経済産業省が 2024年3月27日に公表した **DX支援ガイドンス** は、中堅・中小企業等にとって DX の取組は必要不可欠だが、人材・情報・資金が不足している企業は独力で DX を推進することが難しいとし

ている。だからこそ、現場の業務を知り、既存のIT基盤も把握しているひとり情シスは、守りの土台を支えるだけでなく、SaaS導入や業務改善を実行段階で支える役割も持ちやすい。

## ひとり情シスに対するよくある誤解

ここで、ひとり情シスに関する典型的な誤解を五つ整理しておきたい。

一つ目は、「PCに詳しくれば務まる」という誤解である。確かに、端末やネットワークの基礎知識は必要だ。しかし、実務ではそれだけでは足りない。契約更新、アカウント棚卸し、退職対応、ベンダー調整、利用ルール整備のように、技術だけでは処理できない仕事が多い。

二つ目は、「セキュリティは別の話」という誤解である。実際には、アカウント発行も、端末配布も、SaaS導入も、退職者対応も、すべてセキュリティとつながっている。後からセキュリティを追加すればよいというものではない。

三つ目は、「ベンダーに任せれば終わる」という誤解である。委託は重要だが、責任分界を決めるのは自社である。誰が最終判断をするのか、どの情報を渡すのか、障害時に誰へ連絡するのが曖昧なら、委託は丸投げになり、事故時に最も弱い形で破綻する。

四つ目は、「忙しいから記録は後回しでよい」という誤解である。実際には逆で、忙しいからこそ記録が必要だ。台帳、手順書、申請記録、契約一覧、管理者アカウントの保管ルールがないと、同じ確認を何度も繰り返し、退職や障害のたびにゼロから調べ直すことになる。

五つ目は、「ひとり情シスは一人で全部抱えるのが正しい」という誤解である。ひとり情シスとは、仕事を一人で全部やる人のことではない。本来は分散しているべき仕事を、少人数でなんとかつないでいる状態である。だからこそ、標準化、外注、ルール化、自動化、経営への相談が必要になる。

## ひとり情シスに求められる成果は、作業量ではなく運用品質である

ひとり情シスは、やった作業の数で評価しにくい。チケットを何件処理したかよりも、事故を起こしにくい運用になっているか、止まりにくい基盤になっているか、引き継げる状態になっているかの方が重要である。

本書では、ひとり情シスに求められる成果を、次の五つで捉える。

第一に、止まらないことである。日常業務が回り続ける状態を保つ。大きな改革より、まず止めないことが優先される局面は多い。

第二に、漏れないことである。入社、異動、退職、契約更新、権限変更、バックアップ、ログ保存などで、抜け漏れが起きないようにする。

第三に、残ることである。台帳、手順、判断理由、連絡先、契約情報が記録として残り、自分が不在でも最低限追える状態を作る。

第四に、増えすぎないことである。例外運用、個別対応、野良 SaaS、属人的な設定、管理者権限の乱立を放置すると、将来の自分の首を絞める。仕事を増やさない設計も成果である。

第五に、次に改善できることである。今日の火消しだけで終わらず、同じ問題を減らし、少しずつ標準化し、自動化し、経営と共有できる状態を作る。

この五つを見れば、ひとり情シスの成果は「なんでもすぐやること」ではないと分かる。むしろ、なんでも自分で瞬間対応してしまうほど、運用は残らず、後で崩れやすくなる。

## ひとり情シスが詰まりやすい場所

ひとり情シスは、能力不足より、構造上の詰まりで苦しくなることが多い。典型的なのは次のような場面である。

問い合わせの入口がばらばらで、メール、チャット、口頭、電話に依頼が散っている。すると、優先順位がつけにくく、未対応が発生しやすい。

台帳がない、または古い。端末、アカウント、契約、管理者権限のどれか一つでも曖昧だと、退職対応や障害対応で必ず詰まる。

退職者対応が単発作業になっている。アカウント停止はしたが、共有フォルダの所有権移管やライセンス回収が漏れる、といったことが起きる。

契約更新や証明書更新の期限管理が弱い。普段は問題にならないが、切れた瞬間に事業影響が出る。

バックアップはあるが、復元確認をしていない。いざ戻そうとして初めて、手順がない、権限がない、世代が足りない、保存先が巻き込まれていた、と気づく。

管理者権限が個人依存している。前任者しか知らないアカウント、古いスマートフォンにしか入っていない認証、誰も管理していない共有メールボックスは、典型的な事故の種である。

こうした詰まりは、どれも特別な大事故ではない。日常の延長線上で起きる。そして、起きてから対処すると必ず高くつく。だから本書では、第2章以降で、土台、回し方、対象別管理、有事対応を順に扱っていく。

## 最低限ここまではやる

この章の段階では、すべてを完璧に理解する必要はない。ただし、次の四つを押さえれば十分である。

一つ目。ひとり情シスは、問い合わせ対応係ではなく、会社の IT サービス全体をつなぐ役割である。

二つ目。その仕事は、技術、統制、調整、記録、改善が混ざった複合業務である。

三つ目。業務範囲を意識的に整理しないと、事故は個人の忙しさではなく、構造の問題として起きる。

四つ目。だから、本書では「一つひとつを詳しく知る」前に、「全体をどう捉えるか」を先に整える必要がある。

### 確認したいこと

- 自社で自分が担っている IT 業務を 10 個以上書き出せる
- その業務が、問い合わせ、アカウント、端末、SaaS、セキュリティ、契約、規程のどこに属するか分類できる
- 自分の仕事を「PC の設定」ではなく「会社の IT サービス運営」と言い換えられる
- 今の自分の仕事で、個人依存しているものを 3 つ挙げられる
- 後続章のうち、まず読むべき章を自分で選べる

## 今日、今週、後でやること

今日やることは、30分だけ取り、自分が抱えている仕事を棚卸しすることである。端末、アカウント、契約、問い合わせ、障害対応、外部委託、規程の七つに分けて書き出すだけでよい。その作業をすると、自分が何を担っていて、何が抜けているかが見え始める。

今週やることは、書き出した仕事を **問い合わせ** **日常運用** **導入変更** **セキュリティ** **契約調達** **有事対応** のようなまとまりで見直し、どこに個人依存が強いかを三つ選ぶことである。

後でやることは、本書を読み進めながら、自分の棚卸し結果へ章ごとの学びを追記していくことである。そうすると、この本は読むだけの本ではなく、自社の運営整理ノートとして使える。

## 第2章

## 最初に整えるべき運営の土台

---

朝からチャットで「印刷できません」と連絡が来る。数分後に口頭で「新しいメンバーのアカウント、今日中に作れますか」と聞かれる。その間に、メールで「来月更新の SaaS、契約どうしますか」と問い合わせが届く。昨日頼まれたノート PC の手配は、どこまで進めたか記録がない。先週設定した共有フォルダの権限変更も、誰に何を許可したのか思い出せない。

ひとり情シスが苦しくなるのは、仕事の件数が多いからだけではない。もっと本質的な理由は、依頼が散り、記録が残らず、必要な情報がどこにあるか分からないまま、毎回その場で判断しているからである。

ここでよく起きる誤解がある。「もう少し落ち着いたら整えよう」「まずは詳しいツールを調べよう」「時間ができたら手順書を書こう」という考え方だ。しかし実際には逆で、土台がないから落ち着かず、仕組みがないから毎回時間がなくなる。

この章では、ひとり情シスが最初に整えるべき五つの土台を示す。窓口、記録、台帳、手順、優先順位である。どれも地味だが、この五つがない状態では、どんな高度なセキュリティ対策も、どんな便利な SaaS も、長くは回らない。

### まず整えるべきものは何か

運営の土台として、最初に必要なのは次の五つである。

一つ目は、窓口である。誰が、どこに、何を依頼するのかを決める。

二つ目は、記録である。何を受け、誰が持ち、どこまで進んだかを後から追えるようにする。

三つ目は、台帳である。端末、アカウント、契約、ライセンスなど、運用の対象を一覧で把握できるようにする。

四つ目は、手順である。よくある作業を毎回考え直さずに済むようにする。

五つ目は、優先順位である。声の大きさではなく、影響の大きさに仕事を並べる。

この五つは、それぞれが独立しているわけではない。窓口が一本化されると記録しやすくなる。記録が増えると、よくある依頼が見える。よくある依頼が見えると、手順が書ける。手順が書けると、優先順位に従って落ち着いて処理しやすくなる。土台とは、こうして互いに支え合うものだ。

ITIL では、service desk は利用者にとっての **single point of contact** として整理される。また、CISA の Cybersecurity Performance Goals は asset inventory を baseline に置いている。第2章の五つは、こうした考え方を小規模組織向けに最小化したものである。

最初に作りたい流れは、次の形で十分である。

土台	何のために必要か	最初の形の例
窓口	依頼の入口を一つにする	共有メールアドレス、フォーム
記録	受けた仕事を後から追えるようにする	スプレッドシート、チケット
台帳	何を持っているかを見えるようにする	端末、アカウント、契約、ライセンス一覧
手順	定型作業を毎回考え直さない	チェックリスト、短い手順書

土台	何のために必要か	最初の形の例
優先順位	先着順や声の大きさを振り回されない	高・中・低の簡易ルール

重要なのは、最初から完璧な仕組みを作らなくてよいということである。共有メールアドレスとスプレッドシートは、最小構成の一例として十分に使える。最初に必要なのは、立派さではなく、一つの流れを作ることだ。

## 問い合わせ窓口を一本化する

仕事が崩れる最初の原因は、入口が多すぎることにある。メール、チャット、口頭、電話、廊下ですれ違った時の一言まで、すべてを依頼として受けていると、どれが未対応で、どれが急ぎで、どれが完了したのか分からなくなる。

だから、最初に決めるべきなのは窓口である。ITIL の service desk も、incident と request の入口を一つに集める **single point of contact** として整理している。理想を言えば、チケットツールやポータルがあるとよい。しかし、最初の一歩としては共有メールアドレスでも十分だ。**it@company.co.jp** のような窓口を一つ作り、「IT の依頼はまずここへ送ってください」と決める。それだけでも、仕事の見え方は大きく変わる。

ここで大切なのは、窓口を作ること以上に、そこへ寄せる習慣を作ることだ。チャットで直接頼まれたとしても、「対応はするので、この窓口にも入れてください」と返す。口頭で言われたら、自分で代筆してもよい。最初から全員が守るわけではないが、窓口へ寄せる動きを続けると、少しずつ入口が揃ってくる。

特に意識したいのは、**連絡に使う道具** と **受付記録として残す場所** を分けることである。チャットは連絡には便利だが、受付記録が人ごとの DM や個人チャンネルに散ると、後から追えない。窓口を一本化すると、連絡手段を一つにすることより、**受けた仕事が必ず一か所へ落ちる** 状態を作ることだ。

窓口を一本化すると、三つの効果がある。第一に、依頼の見落としが減る。第二に、今どれだけ仕事を抱えているかが見える。第三に、後から振り返って「何に時間を使っているのか」を把握できる。これは、後の改善や外注判断にもつながる。

## 依頼を記録し、分類する

窓口を一本化したら、次は記録である。ここでいう記録とは、立派な運用台帳を作ることではない。最低限、後から追えることが大事だ。

ここで作りたいのは、単なるメモではなく、**後から追える一か所の記録** である。KCSではこれを **system of record** として扱う。何を受けて、誰が持ち、どこまで進み、どう終わったかが分かる場所が一つ必要だ。

記録には、少なくとも次の項目が必要である。

項目	何のために必要か
受付番号	同じ案件を後から特定できるようにする
受付日時	滞留や即日性を判断できるようにする
依頼者	誰が困っているかを把握する
内容	何を求められているかを残す
分類	処理の考え方を分ける
優先度	順番を決める
状態	未着手、対応中、保留、完了を追う
担当	今誰が持っているかを明確にする
完了日	いつ閉じたかを残す

もしチケットツールを使うなら、これらは自然に入る。使わないなら、スプレッドシートで十分である。重要なのは、どの道具を使うかではなく、毎回同じ項目で残すことだ。

分類は、最初から細かくしなくてよい。まずは三つでよい。

- 依頼
  - アカウント発行、ソフト導入、ライセンス追加など
- 障害
  - メールが使えない、Wi-Fiが落ちた、VPNが繋がらないなど
- 変更
  - 設定変更、権限設計変更、新しい運用ルール導入など

ITIL では、service request と incident は別の実践として整理される。service request は、パスワード再設定やソフト導入のように、あらかじめ定義しやすい依頼である。一方、incident は、メールが止まった、VPNが繋がらないといった復旧を急ぐ事象である。変更はさらに、影響範囲、承認、切り戻しを考える必要がある。だから、全部を同じ箱に入れない方がよい。

この三つを分けるだけでも、仕事の性質が見えやすくなる。

分類	典型例	最初に見たいこと
依頼	アカウント発行、ライセンス追加、ソフト導入	必要情報、承認、標準手順があるか
障害	メール停止、Wi-Fi不調、VPN障害	何人に影響するか、代替手段、復旧優先度
変更	権限見直し、設定変更、新ツール導入	影響範囲、実施日時、戻し方、記録

ここでありがちな失敗は、「記録すること自体が面倒だから、頭の中で管理する」ことである。頭の中の管理は、忙しい間は回っているように見える。しかし、人に聞かれた時、二週間前の依頼を思い出したい時、自分が休んだ時に必ず破綻する。ひとり情シスほど、記憶力ではなく記録に頼るべきである。

## 台帳を作る

ひとり情シスの仕事で詰まりやすい場面の多くは、「何を持っているか分からない」ことから始まる。端末が何台あるのか、誰がどの PC を持っているのか、どのアカウントが生きているのか、どの SaaS を契約していて更新日はいつか。これが曖昧だと、退職対応も、契約更新も、障害対応も、棚卸しもすべて苦しくなる。

最初に必要な台帳は、四つでよい。

一つ目は端末台帳である。端末名、利用者、機種、貸与日、保管場所、状態が分かればよい。

二つ目はアカウント台帳である。氏名、部署、利用サービス、権限の種類、管理者権限の有無が見えるようにする。

三つ目は契約台帳である。サービス名、契約先、契約期間、更新日、解約条件、連絡先、支払方法を記録する。

四つ目はライセンス台帳である。契約数、使用数、未使用数、割当先、見直し時期を残す。

CISA の Cybersecurity Performance Goals では、asset inventory は baseline に置かれている。NIST の SMB 向け Quick-Start Guide でも、まず asset inventory を作り維持することが勧められている。つまり、台帳は余裕ができたなら作るものではなく、運営とセキュリティの出発点である。

ここでも最初から完璧な項目を揃える必要はない。大事なのは、「何がどこに書いてあるか」が決まっていることだ。退職者が出た時に、端末台帳を見ればPCが分かる。アカウント台帳を見れば止めるべきサービスが分かる。ライセンス台帳を見れば回収できるものが分かる。契約台帳を見ればベンダー連絡先と更新条件が分かる。この状態が作れば、仕事は急に軽くなる。

最初の四つの台帳は、次の列から始めれば十分である。

台帳	最低限ほしい列	最初の用途
端末台帳	端末名、利用者、機種、貸与日、保管場所、状態	配布、回収、交換、所在確認
アカウント台帳	氏名、部署、利用サービス、権限、管理者権限の有無	入退社、棚卸し、権限整理
契約台帳	サービス名、契約先、更新日、解約条件、連絡先、支払方法	期限管理、見直し、問い合わせ
ライセンス台帳	契約数、使用数、未使用数、割当先、見直し時期	回収、削減、再割当て

## 手順を残す

ひとり情シスの疲労は、仕事の量だけでなく、「毎回同じことを最初から考える」ことでも増える。パスワード再設定、新規PCの準備、共有フォルダの権限付与、退職者対応、ライセンス追加。何度も発生する作業は、手順として残しておくべきである。

ここで注意したいのは、手順書を完璧なマニュアルにしようとしないことだ。最初は短くてよい。見出しと箇条書きでもよい。重要なのは、対応しながら残すことだ。

たとえば、パスワード再設定の依頼が来たら、その対応の流れを5行でもよいので書く。新しいPCを渡したら、キitting時に確認した項目をチェックリストにする。退職者対応をしたら、止めたサービスや回収物を一覧化する。これを繰り返すと、頭の中にしかない知識が、少しずつ組織の知識に変わっていく。

KCSの考え方では、knowledgeはworkflowの中で使い、直し、足していく。ひとり情シスにとって、この考え方は特に重要である。後で書こうとして書けるほど、仕事は軽くなるからだ。

最初のルールは難しくない。たとえば次の三つで十分である。

- 同じ問い合わせが二回出たら、短い手順にする
- 対応したその日に、五行でも残す
- 既存の手順があれば、新規作成より更新を優先する

## 優先順位を決める

依頼が増えた時、最後に物を言うのは気合いではなく、優先順位である。優先順位が決まっていないと、近くにいる人、声の大きい人、役職が上の人、先に連絡してきた人に引っ張られやすくなる。しかし、本来の基準はそこではない。

最初に必要なのは、単純でよいので、緊急度と影響度の考え方を持つことだ。NISTの最新のincident response guidanceでも、対応要員には限りがある以上、インシデントは単純な先着順で処理するべきではないと整理している。

緊急度とは、どれだけ早く対処が必要かである。影響度とは、どれだけ多くの人や重要な業務に影響するかである。

たとえば、一人だけが困っているプリンタ設定の依頼と、全社のVPN障害は同じではない。新規ソフト導入の相談と、退職者のアカウント停止も同じではない。前者は急ぎに見えても、後者の方が事業影響やリスクが大きいことがある。

最初の運用では、次のような単純な基準で十分である。

- 高
  - 全社または重要業務が止まる
  - セキュリティ事故の可能性がある
  - 退職や権限停止のように即日性が高い
  
- 中
  - 一部部署が困る
  - 期限が近い
  - 業務影響はあるが代替可能
  
- 低
  - 個別依頼
  - 後日でも支障が小さい
  - 定型作業として計画的に処理できる

大切なのは、この基準を自分の中だけに置かないことだ。紙一枚でもよいので残し、説明できる形にしておく。そうすると、依頼者に対しても「なぜ今すぐできないのか」「なぜこちらを先にやるのか」を伝えやすくなる。

## 最初から完璧を目指さない

ここまで読むと、「結局やることが多い」と感じるかもしれない。だが、第2章で伝えたいのは、完成形を一気に作れという話ではない。むしろ逆である。最初から立派なITSMを作ろうとすると、仕組みだけが重くなり、運用が止まる。

だから、最初の三か月は次のような最小構成でよい。

- 共有窓口を一つ決める
- 依頼記録を一つの表で始める
- 端末、アカウント、契約、ライセンスの台帳を仮でも作る
- よくある作業を三つだけ手順化する
- 優先順位ルールを一枚で決める

この状態でまず回してみる。すると、不足が見えてくる。問い合わせの分類が足りない、契約台帳の項目が不足している、退職対応のチェックリストが必要だ、といった具体的な改善が見える。その段階で初めて、道具や制度を足せばよい。

ひとり情シスの運営基盤は、最初から設計し切るものではない。回しながら整えるものである。ただし、何も無い状態から回しながら整えることはできない。だからこそ、最初の骨格だけは先に作る必要がある。

## 最低限ここまではやる

この章の段階では、次の五つがあれば十分に前進である。

- 一つ目。ITの依頼窓口が一つ決まっていること。
- 二つ目。依頼や障害を後から追える記録があること。
- 三つ目。端末、アカウント、契約、ライセンスの最低限の台帳があること。
- 四つ目。よくある作業の手順が少しでも残っていること。
- 五つ目。優先順位を説明できること。

これだけで、仕事は突然楽になるわけではない。しかし、少なくとも毎回ゼロからやり直す状態からは抜けられる。ひとり情シスに必要なのは、まずこの変化である。

### 確認したいこと

- ITの問い合わせ窓口が一つ決まっている
- 依頼を受付日と状態つきで記録している
- 端末台帳がある
- アカウント台帳がある
- 契約台帳とライセンス台帳がある
- よくある作業の手順が三つ以上ある
- 優先順位ルールを言葉で説明できる

### 今日、今週、後でやること

今日やることは多くない。まず、共有窓口を一つ決める。次に、スプレッドシートでよいので、依頼記録と四つの台帳を作る。最後に、よくある作業を三つだけ選び、簡単な手順を書き出す。ここまでできれば、第2章の目的は果たせている。

今週やることは、依頼の状態欄と優先順位ルールを実際の案件へ当てはめ、記録が回るか試すことである。口頭依頼やチャット直投げが残るなら、どこで崩れるかも確認したい。

後でやることは、退職対応、権限変更、端末配布のような定型業務を、申請書やチェックリストへ少しずつ落とすことである。最初から制度化しなくてもよいが、窓口と台帳だけは流れの中心に置いた方がよい。

## 第3章

# ひとり情シスの仕事を回す基本原則

---

朝の時点では、今日は比較的落ち着いていると思っていた。ところが、出社して 30 分で、プリンタ設定の相談、退職者アカウント停止の依頼、来週導入する SaaS の権限設計確認、Wi-Fi の不調、ライセンス追加の問い合わせが一気に来る。どれも急ぎに見える。とりあえず目の前から片づけようとして、チャットに返事をし、管理画面を開き、別の依頼を思い出して中断し、メールを返し、また最初の作業に戻る。気づけば夕方になり、どれも終わっていない。

ひとり情シスの仕事は、件数が多いだけで苦しくなるのではない。仕事の回し方に原則がないまま、すべてを同じ重さで受けてしまうと苦しくなる。標準化できる仕事まで毎回個別判断し、緊急と重要を区別せず、手を付ける案件を増やしすぎ、自分しか知らない状態を放置すると、どれだけ頑張っても仕事は軽くない。

第2章では、窓口、記録、台帳、手順、優先順位という土台を整えた。この章では、その土台の上で仕事をどう回すかを扱う。ここで必要なのは、気合いではなく原則である。

## まず原則を持つ

ひとり情シスの現場では、毎日違う問題が起きているように見える。しかし、仕事の回し方として必要な原則はそれほど多くない。本書では、まず次の五つを基本原則とする。

第一に、標準化できるものは早く標準化する。

第二に、例外だけを個別に考える。

第三に、優先順位は緊急度と重要度で決める。

第四に、同時に抱える仕事を増やしすぎない。

第五に、自分で持つ仕事と外に出す仕事を分ける。

この五つがあると、すべての仕事を自力で瞬間判断し続ける状態から抜けやすくなる。逆に、この五つがないと、仕事はいつまでも「来た順」「近い順」「強く言われた順」に流される。

ここで大事なものは、原則は自分を縛るためのものではなく、自分を守るためのものだということである。ひとり情シスは、頼まれれば何でも対応してしまいやすい。だからこそ、頼まれ方ではなく、回し方で仕事を決める必要がある。

第3章の原則は精神論ではない。ITIL の service request、incident、problem、change の整理や、NIST の優先順位付け、Kanban の WIP 制限の考え方を、小規模組織の現場向けに引き直したものである。

## 標準化できるものを早く標準化する

ひとり情シスの仕事には、毎回判断が必要なものと、実は毎回同じように処理できるものが混ざっている。たとえば、パスワード再設定、ライセンス追加、アカウント発行、端末キッティング、共有フォルダ作成、退職時の基本停止作業などは、ある程度まで定型化できる。

Atlassian の service request management が示す通り、こうした recurring request は repeatable procedure で処理する方がよい。つまり、毎回考えない方がよい。毎回考えている限り、同じ仕事に同じ時間を払い続けることになる。

ここで必要なのは、完璧な標準化ではない。まずは、「これは毎回ほぼ同じだ」と気づくことだ。そのうえで、受付条件、必要情報、処理手順、完了条件を決める。例外が出たら、その例外だけを考えればよい。

ITIL でいう service request は、あらかじめ定義された利用者依頼である。だから、少なくとも recurring な依頼は、**毎回考える仕事** にしない方がよい。ひとり情シスの実務では、次の三つへ分けると扱いやすい。

扱い方	典型例	基本の考え方
標準化しやすい	パスワード再設定、ライセンス追加、定型アカウント発行	受付条件と手順を先に決める
例外として個別判断する	特殊権限付与、例外的なソフト導入、退職直前の特殊対応	必要情報と承認を追加で確認する
慎重に扱う変更	全社設定変更、切り戻しが難しい変更、広範囲の権限見直し	影響、実施日時、戻し方を明確にする

たとえば、新しいソフトを入れてほしいという依頼を考えてみる。毎回その場で「入れてよいか」「誰が承認するか」「ライセンスはあるか」「端末要件は合うか」を考えていると、依頼ごとに判断疲れが起きる。そこで、申請時に必要な情報を決め、承認者を決め、標準的な導入条件を決めておけば、例外だけに頭を使える。

変更作業も同じである。ITIL の change enablement も、リスクを評価しつつ successful changes を増やすことを重視している。すべての変更を重い仕事として扱う必要はない。日常的で影響範囲が読みやすい変更は、標準変更として扱いやすい。逆に、全社に影響する設定変更や、切り戻しが難しい変更は、慎重に扱うべきである。毎回全部を大仕事にすると、変更が滞る。全部を軽く扱うと事故になる。だから、重み付けが必要になる。

標準化の目的は、仕事を機械的にすることではない。判断を減らすことで、本当に考えるべき仕事へ時間を回すことである。

## 緊急度と重要度を分ける

ひとり情シスの仕事で最も崩れやすいのは、緊急と重要が混ざることである。今すぐ返事が必要に見える依頼が、会社にとって本当に重要とは限らない。逆に、今すぐ困っている人は少なくとも、後回しにすると大きな事故になる仕事もある。

ここで分けて考えたいのが、緊急度と重要度である。NIST の最新の incident response guidance でも、対応資源には限りがある以上、インシデントは単純な先着順で処理するべきではないと整理している。ひとり情シスでも同じである。

緊急度は、どれだけ早く処理しなければならないかを示す。たとえば、全社 VPN 障害、退職当日のアカウント停止、役員会直前の会議室接続不良などは緊急度が高い。

重要度は、事業やリスクへの影響の大きさを示す。たとえば、管理者権限の棚卸し、契約更新日の整理、バックアップ復元確認、共有アカウントの見直しは、今この瞬間に困る人が少なくとも重要度が高い。

問題は、緊急度が高い仕事ばかりを追い続けると、重要だが緊急ではない整備が永遠に進まないことである。そして、その整備が進まないからこそ、緊急案件が増え続ける。

優先順位を付けるときは、少なくとも次の四つで見るとよい。

観点	見たいこと	典型例
影響人数	何人が止まるか	一人のプリンタ不調か、全社 VPN 障害か

観点	見たいこと	典型例
事業影響	重要業務が止まるか	受発注、会計、給与、顧客対応に影響するか
リスク	セキュリティや法務へ波及するか	退職者停止漏れ、権限過多、情報漏えいの可能性
即日性	締切や即時対応が必要か	役員会直前、当日退職、期限当日の更新判断

この軸で見れば、声の大きい依頼より先にやるべき仕事が見えやすくなる。優先順位は、自分の気分で決めるものではない。仕事の性質で決めるものである。

## 同時に抱える仕事を減らす

仕事が終わらない原因の一つは、件数そのものより、途中で止まっている仕事が増えすぎることである。Atlassian の Kanban が示す WIP limit の考え方は、ひとり情シスにもよく当てはまる。手を付けた仕事を増やすほど、切り替えコストが増え、どれも終わりにくくなる。

たとえば、五つの仕事を少しずつ進めるより、まず二つを終わらせた方が、全体の流れは良くなることが多い。にもかかわらず、ひとり情シスは依頼が来るたびに反応しやすい。その結果、「対応中」が増え続ける。

これを防ぐには、進行中案件の上限を意識的に決めるのがよい。Kanban の WIP limit は、ボトルネックを見えるようにし、multitasking を減らし、まず終わらせることへ意識を戻すための考え方である。厳密なルールでなくてよい。「今は三件までを進行中にし、それ以上は待ち行列へ置く」と決めるだけでも、かなり違う。緊急障害のように割り込みが必要なものは別として、通常の依頼や改善作業は、抱えすぎない方が早く終わる。

ここでの **三件** は一例である。重要なのは数字そのものではなく、**進行中は増やしすぎない** と明示することだ。

ここで重要なのは、「忙しいから全部同時にやる」は、実際には解決策になっていないということだ。忙しい時ほど、途中の仕事を減らし、終わらせる順序を決める必要がある。

## 火消しだけで終わらせない

ひとり情シスの仕事は、放っておくと火消し中心になる。Wi-Fi が遅い。印刷できない。ライセンスが足りない。同じ人から何度も同じ問い合わせが来る。毎回その場で直して終わりにしていると、確かに目の前の問題は消える。しかし、仕事は減らない。

ITIL の problem management が扱うのは、こうした再発問題の actual / potential causes であり、workaround や known error を記録・管理することで incident の発生確率と影響を減らすことにある。ひとり情シスが大掛かりな problem management を導入する必要はないが、最低限、「繰り返し起きる問題」を別で残すだけでも意味がある。

たとえば、毎月同じ VPN 接続問い合わせが来るなら、手順書や FAQ に落とせないかを見る。同じ部署で権限申請の不備が繰り返されるなら、申請フォームや案内文を見直す。特定の会議室だけ接続トラブルが続くなら、機器構成や利用方法を点検する。こうして一段深く見ると、応急処置で終わっていた仕事が、再発防止の対象へ変わる。

火消しだけで終わらせないとはい、毎回原因分析会をするという意味ではない。少なくとも、「また起きた」を記録し、同じ仕事を減らす方向へ一歩進めるという意味である。

最小限、次の三点だけでも残したい。

- 何が繰り返し起きているか
- その場しのぎの回避策は何か
- 次に手を入れるならどこか

## 何を自分で持ち、何を外に出すか

ひとり情シスが苦しくなるもう一つの理由は、「自分がやるべき仕事」と「自分が持つ必要はない仕事」が混ざっていることである。ここで外注を考えると、「自分が苦手な仕事を投げること」と誤解されがちだが、実際にはそれだけではない。

仕事を外に出すかどうかは、少なくとも次の四つで考えるとよい。

一つ目は頻度である。年に一回しか触らないのに、失敗すると影響が大きい作業は、外部支援を使った方が安全なことがある。

二つ目は専門性である。ネットワーク機器の高度な障害解析や、特殊な法務対応のように、深い専門知識が必要なものは、常時内製する価値が低い場合がある。

三つ目は事故影響である。ミスした時の被害が大きい作業は、自社での理解が必要か、外部の専門家に頼るべきかを慎重に考える。

四つ目は自社理解の必要性である。入社、異動、退職、権限設計、現場の業務フローの理解が必要なものは、自社側で握っていた方がよいことが多い。

ここで大切なのは、外注は責任放棄ではないということだ。NIST の C-SCRM は、組織が **smart acquirer** になること、supplier requirements を定義し communicated requirements を持つことを重視している。要求と責任分界を持たずに頼めば、単なる丸投げになる。外部へ出すなら、「何を依頼するか」「何を自社で確認するか」「障害時に誰が判断するか」を明確にしなければならない。

外注の線引きは、次のように置くと考えやすい。

観点	外に出しやすい仕事	自社で持つ価値が高い仕事
頻度	年に一回程度しか触らない	日常的に発生する
専門性	深い専門知識が必要	現場理解が重要
事故影響	外部保守の方が安全に対応しやすい	自社判断なしでは危ない
自社理解	会社固有事情が少ない	入退社、権限、業務フローに密接

つまり、外に出すべき仕事とは、自社で一から抱える価値が低く、外部の専門性や継続保守の方が効果的な仕事である。一方で、会社の業務理解や日々の判断が必要な仕事は、自社で持つ価値が高い。

## 自分が最大のボトルネックにならない

ひとり情シスは、放っておくと自分自身が最大のボトルネックになる。自分しか知らない管理者設定、自分しか見られない契約情報、自分しか触れない共有メール、自分しか分からない手順。これらは、本人が優秀であるほど増えやすい。

しかし、これは強さではなく、構造的な弱さである。自分が休んだ時、退職した時、緊急対応中で手が離せない時に、一気に詰まるからだ。

KCSの collective ownership や、Kanban が重視するボトルネック低減の考え方を、ひとり情シスの現場向けに言い換えるなら、「自分しかできない仕事を減らす」ことである。すべてを誰でもできるようにする必要はない。だが、少なくとも次のものは残したい。

- 緊急連絡先
- 管理者権限の所在
- よく使う手順
- ベンダー連絡先
- 契約更新の確認方法

ひとり情シスの仕事を回すとは、自分が頑張って回すことではない。自分が詰まっても、全部は止まらない状態を少しずつ作ることでもある。

## 通常時の例と、崩れた時の例

ここで、同じ会社でも回し方でどう差が出るかを見てみよう。

通常時の例では、新規ソフト導入の依頼が来た時、申請フォームで用途、利用者、端末、予算、承認者を確認する。標準条件に合うものはそのまま処理し、例外だけを確認する。進行中案件は三件までに抑え、緊急障害がなければ順番に処理する。よく来る問い合わせは手順化し、毎月繰り返す問題は FAQ に落とす。ネットワーク機器の高度障害だけは外部保守へ即連絡する。結果として、目の前の依頼を全部すぐ処理できなくても、仕事は前へ進む。

崩れた時の例では、依頼が来た順に全部へ反応する。ソフト導入の基準がなく、その場で判断する。進行中案件は常に十件以上あり、どれも途中で止まる。問い合わせが再発しても記録がなく、毎回一から説明する。ネットワーク障害が起きた時の外部連絡先が分からず、自分が調べ始める。結果として、本人は一日中忙しいのに、重要な仕事ほど進まない。

この差を生むのは、能力差より原則の差である。

## 最低限ここまではやる

第3章の段階では、次の六つができれば十分に前進である。

- 一つ目。定型作業と例外作業を分けて考える。
- 二つ目。緊急度と重要度を区別して仕事を並べる。
- 三つ目。進行中案件の数を意識的に制限する。
- 四つ目。繰り返し起きる問題を別で残す。
- 五つ目。外に出せる仕事を一つでも見つける。
- 六つ目。自分しか知らない情報を一つでも減らす。

これだけでも、仕事は「何でも即応する状態」から「流れを設計する状態」へ変わり始める。

### 確認したいこと

- 定型作業と例外作業を分けている
- 優先順位を緊急度と重要度で説明できる
- 進行中案件の件数を把握している
- 繰り返し起きる問題を記録している

- 外注候補を一つ以上挙げられる
- 自分しか知らない設定や情報を減らしている

## 今日、今週、後でやること

今日やることは三つでよい。まず、今抱えている仕事を一覧にして、定型と例外に分ける。次に、進行中案件の上限を一つ決める。最後に、自分しか知らない情報を一つだけ文書化する。

今週やることは、標準化できる作業を三つ選び、手順化することだ。加えて、繰り返し起きている問い合わせや障害を一つ選び、再発防止の観点で見直す。

後で整えることは、外注先の見直しや、自動化候補の整理である。ここは今すぐ全部やらなくてよいが、自分で持ち続けるべき仕事かどうかは意識しておきたい。

第11部

# 経営、統制、調達、導入

判断の所在と責任分担を曖昧にしない。

第4章～第7章

## 第4章

# 経営と IT ガバナンス

---

退職者のアカウント停止が遅れる。バックアップの復元確認は、気になっているが後回しになっている。来月更新の SaaS は、費用を続けるべきか判断が付かない。ネットワーク機器の保守を延長するかも決まっていない。ひとり情シスは、それぞれに問題があることを知っている。しかし、どこまで自分が決めてよいのか分からない。経営へ相談しても、「それって IT の話でしょう」と返される。結局、自分で抱えたまま時間だけが過ぎる。

この状態は、現場の努力不足ではない。会社として、何を守り、何を優先し、誰が何を決めるかが曖昧だから起きる。ひとり情シスがどれだけ真面目でも、経営判断が必要なことまで一人で抱え込めば、いつか必ず限界が来る。

この章で扱う IT ガバナンスとは、重い制度や大きな委員会のことではない。小さな会社でも必要な、「判断の所在を明確にし、現場の運用を経営とつなぐ仕組み」のことである。経営と IT ガバナンスを整えると、ひとり情シスの仕事は急に減らないかもしれない。しかし、少なくとも「自分だけが不安を知っている」状態からは抜けやすくなる。

## IT ガバナンスとは何か

IT ガバナンスという言葉は、実態より大げさに聞こえやすい。だが、本質は単純である。会社として、IT と情報セキュリティに関して、誰が何を決めるかを明確にすることだ。

NIST Cybersecurity Framework 2.0 が 2024年2月26日に公開された時、注目されたことの一つが **Govern** を独立機能として置いた点である。これは、保護策だけでは十分ではなく、組織の文脈、リスク管理方針、役割、監督を先に整える必要があるという考え方を示している。現場で何をするかの前に、会社として何を重視し、どこまでのリスクを受け、誰が責任を持つのが必要だということだ。

NIST の FAQ でも、CSF は IT 部門に閉じず、組織全体の ERM の一部として使う時に最も効果的だとされている。つまり、第4章で扱うガバナンスは、IT 部門だけの作法ではない。会社としての判断の仕組みそのものである。

だから、ガバナンスは会議を増やすことではない。承認欄を増やすことでもない。現場に丸投げしないこと、そして現場が一人で抱え込まなくてよい状態を作ることだ。

ひとり情シスの現場では、ガバナンスがないと、現場担当が次のような判断まで背負いやすい。

- この契約更新は続けるべきか
- この障害リスクは受け入れるのか
- バックアップ強化に費用を使うのか
- 外部委託へ切り替えるのか
- どの業務を優先的に守るのか

これらは、技術判断だけではない。本来は会社の判断である。ガバナンスが必要なのは、そこを分けるためだ。

経済産業省の **サイバーセキュリティ経営ガイドライン Ver 3.0**も、組織幹部が自ら役割を認識し、リーダーシップを発揮して対策強化と適切対応を進める必要があるとしている。ひとり情シスが不安を抱え込むのではなく、経営判断が必要な論点を上げることは、現場の弱さではなく、ガバナンス上の正しい動きである。

## 経営が持つべき論点と、現場が持つべき論点

ひとり情シスが楽になるためには、「何でも経営に聞く」状態を作ればよいわけではない。逆に、「現場で全部決める」状態も危うい。必要なのは、経営が決めるべきことと、現場が決めるべきことを分けることである。

経営が持つべき論点は、大きく四つある。

第一に、何を優先して守るかである。売上、受発注、顧客対応、給与、法令対応など、止まって困る業務の優先順位は経営判断に属する。

第二に、どのリスクをどこまで受けるかである。古い機器をすぐ更新できない、すべての運用を即時に強化できない、という現実はある。その時に、何を許容し、何を許容しないかは会社として決める必要がある。

第三に、どこへ投資するかである。MFA 導入、バックアップ強化、端末更新、外部保守、監査ログ強化などは、費用と優先順位を伴う。

第四に、どこまでを外部へ任せるかである。ネットワーク保守、監視、ヘルプデスク、法対応支援など、外部委託の方針は現場だけでは決めきれない。

一方で、現場が持つべき論点もある。設定手順、日常運用、問い合わせ対応、標準手順、軽微な改善、日々の記録は現場で回すべきだ。毎回経営判断を仰いでいては運用が止まる。

ここで重要なのは、ひとり情シスが「何を自分で決めてよいか」を把握することだ。経営判断が必要な論点まで自分で抱えようと、判断の重みが増すだけでなく、後で「なぜその判断をしたのか」を説明しにくくなる。

NISTのFAQでは、senior leadersがcybersecurity riskを理解し、方向を示し、優先順位付けやコミュニケーションを改善する役割を持つと説明している。つまり、現場から見て **相談しないと決められない** と感じる論点は、往々にして本当に経営が持つべき論点である。

迷った時は、次のように切り分けると判断しやすい。

論点	主に経営が決めること	主に現場で回すこと
優先順位	どの業務を先に守るか	その優先順位に沿って何から着手するか
リスク受容	どこまでの停止や未整備を許容するか	現状の弱点と改善案を示すこと
投資	予算を付けるか、先送りするか	必要性、影響、選択肢を比較して出すこと
委託	外部へ任せる範囲をどこまで広げるか	日常の窓口、運用、監督を回すこと
ルール	会社として守る方針を承認すること	手順、記録、台帳へ落とすこと

## 何を守るのかを整理する

ガバナンスの出発点は、制度でも方針書でもなく、「何を守るのか」を整理することにある。守る対象が曖昧なままでは、優先順位も報告も作れない。

ここで最初に見たいのは、重要業務である。たとえば、受発注、売上計上、請求、顧客サポート、給与計算、在庫管理、店舗運営など、止まると事業に直接影響する業務を挙げる。全部を同じ重さで扱う必要はない。まずは三つから五つに絞る。

次に、その業務を支える重要資産を見る。どのシステム、どのデータ、どの端末、どのアカウント、どの外部サービスが止まると業務が止まるかを確認する。ここで初めて、「この SaaS は便利だから重要」ではなく、「この業務を止めないために重要」という見方ができる。

さらに、その業務と資産に対して、どんなリスクがあるかを考える。アカウント停止漏れ、契約更新漏れ、バックアップ不足、外部共有ミス、ネットワーク障害、ベンダー依存、管理者権限の属人化。こうして見ていくと、技術課題が事業課題として並び始める。

たとえば、受発注業務が重要なら、それを支えるメール、クラウドストレージ、認証、ネットワークの安定性が重要になる。給与計算が重要なら、人事データ、委託先、アクセス権、バックアップが重要になる。このつながりが見えると、経営へも説明しやすくなる。

## 方針と優先順位を決める

何を守るかが見えたら、次は方針と優先順位である。ここで難しく考える必要はない。NIST の Organizational Profiles や、IPA の可視化系ツール、DX 推進指標に共通しているのは、現状と目標の差を見えるようにすることだ。

まず、現状を書く。たとえば「管理者権限の一覧がない」「退職者対応のチェックリストがない」「バックアップはあるが復元確認をしていない」「契約更新管理が担当者依存している」といった状態である。

次に、目標を書く。たとえば「三か月後には管理者権限一覧がある」「退職対応チェックリストを運用している」「重要システムの復元確認を年一回実施している」「契約更新日を一覧で見られる」といった状態である。

最後に、その差の中から優先順位を決める。ここで全部を一気に埋めようとしていないことが大切だ。重要業務に直結するもの、事故時の影響が大きいもの、すぐ着手できるものから順に並べる。

現状と目標は、次の形で十分である。

項目	現状の書き方	目標の書き方
管理者権限	一覧がなく、担当者依存している	一覧があり、四半期ごとに見直している
退職対応	担当者ごとにやり方が違う	チェックリストで同じ手順で止めている
バックアップ	あるが復元確認をしていない	重要システムは年1回復元確認している
契約更新	更新日がメール依存で見えない	90日前に確認できる台帳がある

この時に役立つのが、短い基本方針である。大きな規程集をいきなり作る必要はない。SECURITY ACTION 二つ星でも、まず自己診断と情報セキュリティ基本方針を出発点にしている。小さな会社なら、まずは次のような短い方針で十分だ。

- 重要業務を止めないことを優先する
- アカウントと権限は記録して管理する
- 退職、契約更新、バックアップは期限管理する
- 事故や重大リスクは経営へ報告する

方針が短くても、会社として合意されていることに意味がある。ひとり情シスの個人ルールを、会社のルールへ変える最初の一步になるからだ。

ここで注意したいのは、IPA の **サイバーセキュリティ経営可視化ツール** は原則として 300 名以上の企業向けだという点である。小規模組織では、そのツールをそのまま使うより、**現状と目標の差を見える化する考え方** を借りる方が実務的である。一方、IPA の **中小企業の情報セキュリティ対策ガイドライン** や **SECURITY ACTION** は、小規模組織が直接出発点として使いやすい。

## 経営へどう報告するか

ガバナンスが弱い会社では、報告も崩れやすい。技術用語ばかり並び、経営が何を見ればよいか分からないか、逆に「特に問題ありません」だけで終わってしまうかのどちらかになりやすい。

NIST の ERM Quick-Start Guide が 2024年10月21日に示したように、重要なのはサイバーリスク情報を enterprise risk management に統合することだ。言い換えれば、技術の出来事を経営が判断できる言葉へ翻訳する必要がある。

経営への報告は、一枚でよい。たとえば、次の四項目だけでも十分に意味がある。

- 今月の出来事
  - 重大障害、事故、更新、棚卸し結果
- 今ある主なリスク
  - 重要業務に影響する未対応事項
- 近い期限
  - 契約更新、証明書更新、保守期限、重要な見直し時期
- 判断が必要なこと
  - 投資、許容リスク、委託、優先順位変更

最初の一枚報告は、次の型で十分に使える。

欄	書くこと	例
今月の出来事	起きた事実だけを書く	退職者対応フローを運用開始した
主なリスク	放置すると何が止まるかを書く	VPN 機器が保守切れで、在宅勤務停止リスクがある
近い期限	日付つきで書く	5月15日に回線更新判断が必要
判断が必要なこと	経営に決めてほしいことを書く	VPN 更新費用を今期計上するか判断が必要

この一枚報告では、技術用語より次の三点が見えることを優先したい。

- 何の業務が止まりうるか
- そのまま放置すると何が起きるか
- 会社として何を決める必要があるか

ここで大切なのは、報告の主語を技術から事業へ移すことだ。たとえば「VPN 機器が古い」ではなく、「在宅勤務と拠点接続の停止リスクが高いので、更新判断が必要」と言う。「管理者アカウントの棚卸しが未了」ではなく、「退職や権限漏れの事故リスクが残っている」と言う。この翻訳ができると、経営は初めて判断しやすくなる。

## 小規模組織で現実的に回るガバナンス

ここまで読むと、ガバナンスは手間がかかりそうに見えるかもしれない。だが、小規模組織で必要なのは大掛かりな委員会ではない。むしろ、重い制度は続かない。

現実的なのは、月一回三十分でもよいので、定例の場を持つことだ。参加者は、経営者か責任者、ひとり情シス、必要に応じて総務や経理で十分である。そこで毎回同じ型で確認する。

- 重要業務に関わる問題は何か
- 期限が近いものは何か
- 今月起きた出来事は何か
- 判断が必要なことは何か
- 次に優先する整備は何か

もし最初から定例を取れないなら、まずは一枚報告を同じ型で毎月送るだけでもよい。宛先は経営者だけでなく、総務や経理など、契約更新、支払、入退社に関わる責任者でもよい。重要なのは、相談のたびに口頭で消える状態をやめ、判断が必要な論点を会社に見える形で残すことである。

この型があるだけでも、ひとり情シスが一人で不安を抱える状態はかなり減る。さらに、DX推進指標のように **現状を見て次の一歩を決める** 発想を借りて、現状と目標を年に一度でも見直せば、会社としての進み方が見える。

重要なのは、会議の長さではなく、見るべきものと決めるべきものが定まっていることだ。逆に、定例の場があっても、何を見て何を決めるかが曖昧なら、ガバナンスにはならない。

## 通常時の例と、崩れた時の例

通常時の例では、月初にひとり情シスが一枚報告を作る。内容は、先月の障害、今月の更新期限、残っている主なリスク、判断が必要な事項の四つだけである。月一回、社長と管理部門責任者と三十分話す。そこで、「今月は退職者対応フ

ロー整備を優先する」「VPN 機器更新は来期予算へ入れる」「契約更新管理は総務と分担する」といった判断を決める。すべてを解決できなくても、少なくとも会社としての優先順位がそろろう。

崩れた時の例では、ひとり情シスが不安だけを抱えている。更新期限が近い契約があるが、誰に相談すべきか分からない。退職者対応の漏れが気になるが、業務フローを変える権限がない。バックアップの弱さも分かっているが、費用を使う判断は取れない。結果として、現場の不安は増え、経営は何も知らず、事故が起きた時だけ「なぜ言わなかったのか」となる。

この差を生むのは、立派な制度の有無ではない。判断の所在と報告の型があるかどうかである。

## よくある失敗

第4章で特に避けたい失敗は五つある。

一つ目は、経営に何も上げないことだ。現場で抱えた方が早い場面はあるが、経営判断が必要な論点まで抱え込むと、後で必ず詰まる。

二つ目は、逆に何でも経営へ上げることだ。日常運用まで上げていては、現場が止まる。役割分担が必要である。

三つ目は、報告が技術用語だらけになることだ。経営は技術詳細を知らなくてよい。必要なのは、事業影響と判断事項である。

四つ目は、重要業務を整理しないことだ。これがないと、優先順位が永遠にぶれる。

五つ目は、会議だけ増やして何も決めないことだ。ガバナンスは資料作りのためにあるのではなく、意思決定のためにある。

## 最低限ここまではやる

第4章の段階では、次の五つができれば十分に前進である。

- 一つ目。止められない業務を三つ書き出せること。
- 二つ目。経営が決めるべき論点と、現場で回す論点を分けられること。
- 三つ目。短い基本方針を持てること。
- 四つ目。月次で共有する一枚報告の型を持てること。
- 五つ目。月一回でも、経営とITを話す場があること。

これだけでも、現場の孤立はかなり減る。ひとり情シスの仕事を安定させるとは、現場が全部強くなることではない。会社として判断に参加することでもある。

### 確認したいこと

- 重要業務を三つ以上挙げられる
- その業務を支える重要システムや資産を説明できる
- 経営判断が必要な論点を言語化できる
- IT課題を事業影響で説明できる
- 月次で見る項目を決めている
- 現状と目標の差を説明できる

## 今日、今週、後でやること

今日やることは三つでよい。まず、止められない業務を三つ書く。次に、今気になっている IT やセキュリティ上の不安を、事業影響の言葉へ書き換える。最後に、月次一枚報告のひな型を作る。

今週やることは、経営者が責任者と話す時間を確保することだ。長い会議は要らない。三十分でよいので、重要業務、残っているリスク、必要な判断を共有する。すぐに時間が取れないなら、一枚報告を先に送り、判断が必要な論点だけでも文面で残したい。

後で整えることは、方針の文書化、可視化ツールの活用、目標設定の見直しである。最初から全部を制度化する必要はないが、現場の不安を会社の判断へ変える流れだけは早めに作りたい。

## 第5章

## IT 予算、契約、調達、ライセンス

---

一番安いプランを選んだ。営業からは、今なら割引が大きいと言われた。機能も一通りそろっているように見えた。だから導入した。ところが、数か月後から空気が変わる。設定変更のたびに追加費用が発生する。問い合わせても、そこはサポート対象外だと言われる。人員が減っても年間契約なので席数を減らせない。退職者のアカウントを止めたのに課金は続いている。解約しようとする、データの取り出しや移行に思った以上の手間がかかる。

こうした詰まり方は珍しくない。むしろ、ひとり情シスの現場ではよく起きる。理由は単純である。調達を「買う瞬間」の話として見てしまい、「使い続ける」「増減する」「引き継ぐ」「やめる」まで含めて考えないからだ。

第4章では、経営と現場の間で何を守り、何を優先し、誰が何を決めるかを整理した。第5章は、その判断を具体的な購入と契約に落とす章である。ここで扱うのは節約術ではない。後で運用が破綻しないための判断軸である。IT の予算、契約、調達、ライセンスを別々の話としてではなく、一つの流れとして見る。

### 調達は買い物ではなく設計である

ひとり情シスの現場では、調達が「どの製品を選ぶか」という話に縮みやすい。だが、本当はもっと広い。調達とは、導入後の運用のしやすさ、障害時の支援、権限管理、費用変動、契約更新、解約や移行まで含めて、自社がその仕組みを回せるかを設計することである。

IPA の情報システム・モデル取引・契約書でも、重要なのは責任分界の明確化である。つまり、契約は「発注したら終わり」の紙ではなく、誰が何をするかをはっきりさせるための道具だということだ。ここが曖昧なまま導入すると、障害時に「そこは御社側の作業です」と言われ、変更時には「別料金です」となり、更新時には「その条件では減数できません」となる。

導入前に最低限見ておきたいのは、次の三点である。

見る点	導入前に確認すること	後で詰まりやすいこと
責任分界	初期設定、日常運用、設定変更、障害切り分けのどこまでが相手の責任か	そこは御社作業ですと言われる
サポート	連絡手段、受付時間、一次回答の目安、緊急時の連絡経路	問い合わせても返答が遅い、連絡先が分からない
終了条件	減数、解約、データ返却、更新停止の条件	やめたい時に止められない、移行が間に合わない

価格が安いかどうかは、その一部でしかない。むしろ、価格だけで決めた時ほど、後で高くつく。人の手間、問い合わせ回数、設定の複雑さ、移行の難しさ、解約のしにくさまで含めて初めて調達の判断になる。

第5章でまず持っておきたい視点は一つだけでよい。何をかうかではなく、買った後に自社で回るかを見る。これが調達の出発点である。

## IT 予算は価格ではなく継続コストで考える

予算の話になると、どうしても初期費用や月額費用の比較に意識が寄りやすい。しかし、ひとり情シスが本当に見るべきなのは、払う金額そのものより、その支出が何を減らし、何を増やすかである。

たとえば、月額が安い SaaS でも、設定変更のたびにベンダーへ依頼が必要なら、社内の待ち時間と調整負荷が積み上がる。サポートが弱ければ、結局自分で調べて回る時間が増える。逆に、少し高くても権限管理や監査ログや自動化が整っていれば、日常運用の手間は下がる。予算とは単価の話ではなく、手間と事故と停止リスクまで含めた継続コストの話である。

ここで見たいのは、少なくとも次の五つである。

コストの種類	典型例	見落とすと起きること
初期費用	導入支援、初期設定、移行作業	見積外作業が後から増える
継続費用	月額、年額、保守料、サポート料	単価比較だけで判断しやすい
追加課金条件	席数追加、容量追加、設定変更、API 利用	後から必要な作業で想定外課金が出る
運用工数	入退社対応、棚卸し、問い合わせ、障害切り分け	一人の手間が積み上がり、他業務が止まる
終了コスト	データ移行、エクスポート、解約対応、再設定	やめる時に時間も費用も足りなくなる

このうち、運用工数と終了コストは見落とされやすい。だが、ひとり情シスにとっては非常に大きい。担当者が一人しかいない環境では、「毎月少し面倒」と「やめる時に一気に重い」が重なるだけで、十分に重いコストになるからだ。

だから予算を考える時は、「いくらかかるか」だけでなく、「この製品を選ぶことで毎月何時間減るか、逆に何時間増えるか」を考える必要がある。金額に換算しなくてもよい。だが、手間が増えるか減るかは必ず見たい。

もう一つ重要なのは、重要業務との関係である。第4章で整理した重要業務に直結するシステムなら、安さだけで選ぶべきではない。受発注、売上計上、顧客対応、給与計算のような止まると困る業務では、安いことより、止まりにくいこと、復旧しやすいこと、相談しやすいことの方が重い場合が多い。

## 見積比較では何を見るか

見積を取ると、金額の列が一番目に入りやすい。だが、比較表を作るなら、金額列だけでは足りない。最低でも、次の軸を並べたい。

- 機能
- 非機能
- サポート範囲
- 契約期間と更新条件
- 解約や減数のしやすさ
- データ移行と持ち出しのしやすさ
- 追加課金の条件

比較表には、軸だけでなく、最低限どこまで確かめるかも入れておきたい。

項目	最低限見ること	差が出やすい点
機能	主要業務が回るか	一見似ていても細かな制約が違う
非機能	ログ、権限、バックアップ、運用しやすさ	障害時や監査時に差が出る
サポート	受付時間、回答手段、障害時連絡	困った時に使えるかどうか
契約条件	契約期間、自動更新、減数可否、途中解約	単価差より総額差が出る

項目	最低限見ること	差が出やすい点
データ移行	エクスポート形式、引き継ぎ方法、削除後の扱い	乗り換え時の難しさが違う
追加課金	席追加、容量追加、設定変更、保守作業	導入後に予算超過しやすい

機能は分かりやすい。欲しいことができるかどうかだ。しかし、実際に差が出やすいのはその先である。たとえば、同じように見えるクラウドサービスでも、障害時の連絡手段、問い合わせ窓口、回答速度、バックアップの扱い、監査ログの有無、権限の細かさ、APIの使いやすさはかなり違う。

また、契約期間の違いは非常に大きい。たとえば Google Workspace の公式案内では、Flexible Plan は月次課金で、利用者の追加削除と解約が随時できる。一方、Annual/Fixed-Term Plan は年単位以上のコミットがあり、更新時まで減数できず、途中解約では残契約分の支払いが必要になる。人員変動が大きい会社で単価の安い年間契約を選ぶと、結果的に使わない席の費用を払い続けることがある。これは Google Workspace 固有の一例だが、**単価が安い契約ほど増減が硬いことがある**という読み方は他製品でも有効である。

見積比較で大切なのは、「導入時の一回勝負」で表を作らないことだ。日常運用、組織変更、障害、契約更新、解約の各場面で何が起きるかを想像して並べる。比較表は、製品選定のためだけでなく、後で自分を守るための記録でもある。

## 非機能要件を先に確認する

調達で詰まりやすいのは、機能不足より非機能の見落としである。IPA の重要情報を扱うシステムの要求策定ガイドが示している通り、重要情報を扱うシステムでは、利便性だけでなく、自律性や統制を保てるかも見なければならない。つまり、業務で使うシステムを選ぶ時は、できることだけでなく、止まりにくさ、支えやすさ、管理しやすさを見なければならない。

非機能という言葉は抽象的に見えるが、ひとり情シスの実務に引き直すと分かりやすい。見るべきことは、たとえば次のような項目である。

- 障害が起きた時にどこまで支援されるか
- バックアップや復旧はどうなっているか
- ログはどこまで残るか
- 権限を細かく分けられるか
- 管理画面は一人で扱える複雑さか
- 外部連携やデータ出力はしやすいか
- 保守や変更作業に追加費用が発生するか

たとえば機能面では十分でも、監査ログが弱く、管理者権限の分離もできず、退職者データの保持も難しい製品なら、後で確実に困る。逆に、機能が少し地味でも、権限、ログ、データ出力、サポートがしっかりしていれば、長く安定して使いやすい。

小さな会社で全部の様式を作る必要はない。だが、少なくとも **便利か** だけでなく、**止まった時に自社で支えられるか** **管理を一人で維持できるか** を並べて考える必要はある。非機能を見るときは、立派な資料を作るのではなく、平常時と異常時の両方で回るかを先に考えることである。

ひとり情シスが非機能を見る理由は、立派な要件定義書を書くためではない。後で自分だけが無理をしないためである。導入後の運用を一人で背負う可能性が高い以上、運用負荷を先に見ておくのは当然の防御である。

## 契約で確認すべきポイント

契約は法務部門だけの話ではない。もちろん法的な詳細条項を全部一人で判断する必要はない。だが、現場として確認すべき論点はある。それは、この契約で何を期待でき、何を期待できないかを明確にすることだ。

最低でも確認したいのは、次のような点である。

- 誰が何をやる契約なのか
- サポート対象はどこまでか
- 問い合わせ方法と受付時間はどうなっているか
- 障害時の連絡やエスカレーションはどうか
- データの保存場所や取り出し条件はどうなっているか
- 自動更新の有無と解約期限はどうなっているか
- 再委託や外部サービス利用の扱いはどうなっているか

実務では、次の形で一枚にしておくで見落としにくい。

項目	確認したいこと	典型的に後で揉める点
導入支援	初期設定、移行、教育のどこまで含むか	導入支援あり だが実作業は対象外
サポート	連絡手段、受付時間、回答目安	緊急時に連絡できない、返答が遅い
障害対応	誰が切り分け、誰が復旧し、どこでエスカレーションするか	障害時に責任が押し合いになる

項目	確認したいこと	典型的に後で揉める点
データ返却	どの形式で返せるか、終了後いつまで取れるか	乗り換え時に出せない、時間切れになる
更新と解約	自動更新、解約期限、途中解約、減数条件	気づいた時には次年度が始まっている
再委託	どの外部サービスを使い、どこにデータが置かれるか	想定外の委託先や保存先が後で分かる

ここで重要なのは、期待と現実のずれを潰すことである。たとえば「導入支援あり」と書かれていても、実際は初期設定の説明だけで、データ移行や運用設計は対象外かもしれない。「サポートあり」と書かれていても、メールのみで回答に数日かかるかもしれない。「バックアップあり」と書かれていても、利用者側が復元依頼できる範囲は限定的かもしれない。

また、解約条件は後回しにされやすいが、むしろ導入前に見ておきたい。どのくらい前までに連絡が必要か。契約期間中の減数はできるのか。データはどの形式で返せるのか。終了後の保持期間はどうなっているのか。これらは、やめる時だけでなく、価格交渉や継続判断にも効いてくる。

## ライセンス管理は調達の一部である

ライセンス管理は、購入後の事務作業のように見えやすい。だが実際には、調達の時点で設計しておかないと後で苦しくなる。なぜなら、ライセンス管理は課金管理だけでなく、権限管理、人の出入り、データ保持とつながっているからだ。

Microsoft 365 でも Google Workspace でも、ライセンスは誰にどう配るかで運用負荷が変わる。個別に手で割り当てるのか。グループベースで付与するのか。新規入社時に自動で付くのか。不要になった時にどの手順で外すのか。こうした設計を曖昧にしたまま使い始めると、増員時に毎回作業がばらつき、棚卸し時には未使用ライセンスが見つからず、退職時には権限と課金の両方で漏れが出る。

ここは、製品の自動化機能も踏まえて考えたい。Google Workspace には組織単位の自動ライセンス付与があり、Microsoft 365 にはグループベースのライセンス付与がある。つまり、ライセンス配布は **人に手で配る作業** ではなく、入社、異動、退職の流れと一緒に設計できるテーマである。ただし、自動化にも前提条件がある。Google Workspace では反映まで時間がかかることがあり、Microsoft 365 のグループ付与では Usage location の設定が前提になる。自動化機能があるだけで、設計なしに回るわけではない。

ひとり情シスの現場では、ライセンス管理を次の三つで同時に見たい。

- 課金
- 権限
- データ保持

課金だけを見て削減すると、必要なデータまで失うかもしれない。権限だけを見て管理者を増やすと、ライセンス操作の統制が崩れるかもしれない。データ保持だけを優先すると、不要な高額ライセンスを抱え続けるかもしれない。だから、この三つを分けずに考える。

また、誰がライセンスを操作できるかも重要である。Microsoft の公式資料でも、必要最小限の権限を使う考え方が明示されている。ひとり情シスが全部を持つ場面は多いが、それでも高権限をむやみに広げない原則は持っておきたい。**ライセンスを配れる人** と **全体設定まで触れる人** を同じにしないだけでも、事故は減らしやすい。

## 停止、削除、アーカイブを混同しない

ライセンス運用で特に危ないのが、「止めたから費用も止まるだろう」「削除したからきれいに終わっただろう」と考えてしまうことだ。実際には、停止、ライセンス削除、アーカイブ、削除は意味が違う。

Google Workspace の公式説明では、Flexible Plan では停止した利用者でも active user と同率で課金が続く。つまり、ログインできない状態にしたことと、費用が減ることは同じではない。また、Google Workspace はライセンスを外すとデータを失う可能性があると案内しており、データを保持したいなら Archived User ライセンスを案内している。ここでは Google Workspace の具体例だが、他製品でも **使えなくする操作** **費用を止める操作** **データを残す操作** が分かれていることは多い。

判断を急ぐ時ほど、次の表で整理すると事故が減る。

やりたいこと	操作候補	注意点
今すぐ使えなくしたい	停止	使えなくなっても課金は続くことがある
費用を下げたいがデータは残したい	アーカイブ	製品や契約によって使える仕組みが違う
ライセンスだけ外したい	ライセンス削除	データ消失や機能停止につながることもある
利用もデータも整理したい	削除	事前のデータ移管、保持期間、法務要件の確認が必要

ここで整理したいのは、三つの問いである。

- その人を今すぐ使えなくしたいのか
- その人のデータを残したいのか
- その費用を止めたいのか

この三つは似ているようで別の話である。入社、異動、休職、退職のたびにここを混同すると、課金漏れ、データ消失、権限残りのどれかが起きやすい。第9章では人の出入りの運用を詳しく扱うが、第5章ではその前提として、「費用」「利用可否」「保持」は同じ操作ではないと理解しておきたい。

## 小さな会社で現実的に回る判断

ここまでを読むと、調達前に確認することが多すぎるように見えるかもしれない。だが、小さな会社に必要なのは完璧な調達プロセスではない。後で詰まりやすい点を、最低限先に見ておくことだ。

現実的には、まず次の四つだけでも十分に効果がある。

- 固定費になっている契約について、契約名、契約主体、更新日、解約期限、減数可否を一覧にする
- 更新日と解約期限をカレンダーへ入れる
- 見積比較表に非機能、減数条件、データ返却、解約条件の列を足す
- 月に一度、未使用ライセンス、休職者、退職者、使っていない契約をまとめて見る

この四つがないと、判断はいつも場当たりになる。逆に、この四つがあるだけで、「どれを続けるか」「どれを減らすか」「どれを標準にするか」をかなり落ち着いて考えられる。

もう一つ大事なものは、標準製品を決めることである。部門ごとに似たようなツールが乱立すると、ライセンス管理、問い合わせ対応、教育、権限管理が全部重くなる。全部を一つに統一する必要はないが、メール、ストレージ、チャット、会議、端末管理のような基盤部分は、できるだけ標準を持った方がよい。標準化は第3章で扱った運営原則の延長であり、調達でも同じように効く。

## 通常時の例と、崩れた時の例

通常時の例では、新しい SaaS を検討する時に、ひとり情シスが三社比較表を作る。列は金額だけではなく、サポート、ログ、権限、データ出力、契約期間、減数条件、解約期限、管理画面の扱いやすさまで入っている。経営へは「A が最安だが、減数しにくく運用負荷が高い。B は少し高いが、権限管理と解約条件が良い。重要業務に使うので B を推奨する」と説明する。導入後は、ライセンス付与方法、棚卸しの頻度、退職時の扱いまで最初に決める。結果として、日常運用が安定する。

崩れた時の例では、営業提案の勢いで一番安い年間契約を選ぶ。比較表は金額と機能だけで、減数条件もデータ移行条件も見えていない。入社時は都度手作業でライセンスを付け、退職時は停止だけして終える。半年後、使っていないライセンスが積み上がり、解約したいが更新期限を過ぎており、古いデータをどう残すかも決まっていない。ひとり情シスは、安く入れたはずの製品の後始末に時間を取られ続ける。

この差は、特別な調達部門の有無ではない。買う前に、後で起きる運用を想像したかどうかである。

## よくある失敗

第5章で特に避けたい失敗は五つある。

一つ目は、単価だけで決めることだ。安いことは大事だが、安さだけでは十分ではない。

二つ目は、機能だけを見て非機能を見ないことだ。ログ、権限、バックアップ、サポート、移行性を後から足すのは難しい。

三つ目は、契約を読まずに運用を始めることだ。特に自動更新、減数条件、解約期限、支援範囲は必ず見たい。

四つ目は、ライセンス管理を総務作業のように軽く見ることだ。実際には課金、権限、データ保持に直結する。

五つ目は、停止、削除、アーカイブを同じものとして扱うことだ。ここを混同すると、費用かデータか統制のどこかで事故になる。

## 最低限ここまではやる

第5章の段階では、次の五つができれば十分に前進である。

- 一つ目。契約中の主要 SaaS と保守契約を一覧にできること。
- 二つ目。各契約について、更新日と解約期限を把握していること。
- 三つ目。新しい製品を比較する時に、金額以外の列を持てること。
- 四つ目。未使用ライセンスと退職者アカウントの状態を確認できること。
- 五つ目。停止、削除、アーカイブの違いを説明できること。

これだけでも、調達はかなり安定する。ひとり情シスの仕事を守るとは、導入時に頑張ることではない。導入後に無理をしなくて済む形を先に作ることもある。

### 確認したいこと

- 単価以外の比較軸を持っている
- 非機能要件を確認している
- 契約更新日と解約期限を把握している
- サポート範囲と責任分界を説明できる

- ライセンス棚卸しの方法を持っている
- 停止、削除、アーカイブの違いを説明できる

## 今日、今週、後でやること

今日やることは三つでよい。まず、現在契約している主要 SaaS と保守契約を一覧にする。次に、その中で更新日が近いものと、使っていない可能性があるライセンスを一つずつ洗う。最後に、今後の見積比較表へ「非機能」「解約条件」「データ移行」の列を足す。

今週やることは、主要契約について、更新条件、減数条件、サポート範囲、解約期限を確認することだ。可能なら、経営や責任者とも一度共有し、「単価ではなく運用まで見て選ぶ」という基準をそろえておきたい。

後で整えることは、ライセンス付与の標準化、契約更新カレンダーの運用、標準製品の整理である。これらは一度に完成しなくてよいが、放っておくほど後で整理が難しくなる。

## 第6章

## ベンダー管理と外部委託

---

ネットワーク保守は外部へ任せている。SaaS の初期設定は導入会社が入った。PC のキッティングも一部は業者に頼んでいる。だから自分の仕事は軽くなるはずだった。ところが実際には、障害が起きると、どこへ連絡すべきかを自分が判断する。設定変更の相談が来ると、どこまで業者へ頼めるのかを自分が確認する。契約更新が近づけば、継続するかどうかを自分が考える。業者が複数いれば、「そこは別ベンダーの範囲です」と言われ、そのつなぎ役も自分になる。

ひとり情シスにとって、外部委託は仕事をなくす魔法ではない。むしろ、うまく使えば自分だけでは回し切れない部分を補えるが、使い方を誤ると、見えない仕事と調整仕事を増やす。

第5章では、何を買ひ、どう契約し、どうライセンスを持つかを扱った。第6章で扱うのは、その次である。契約した相手と、どう付き合い、どう監督し、どう終わらせるかだ。ここで言うベンダー管理は、相手を疑うための仕組みではない。自社に残る責任を見失わず、委託先の力を安定して使うための仕組みである。

### 外部委託は責任放棄ではなく責任分担である

外部委託をすると、作業の一部は外へ出せる。しかし、責任まで全部が外へ移るわけではない。ここを誤解すると、外部委託は必ず崩れる。

Microsoft Learn の **Shared responsibility in the cloud** が 2026年1月12日更新版で明確に示している通り、クラウドでは Microsoft 側へ移る責任がある一方で、どのサービス形態でも customer は data や identities を持ち続け、accounts と access management の責任も retain する。つまり、SaaS やクラウドへ任せても、誰が使えるか、どの権限を持つか、どのデータを守るかは自社の責任に残る。

これはクラウド以外の委託でも同じである。ネットワーク保守を外へ任せても、その変更を承認するのは誰か、どの障害を重大とみなすか、どの時間帯まで対応を求めるか、どこまでの費用を受け入れるかは、自社で決める必要がある。

第6章でまず押さえないのは、次の区別である。

- 委託先が行う作業
- 自社が持つ判断

作業は任せられる。だが、優先順位、例外判断、重要変更の承認、受け入れ可否、継続可否は自社に残る。この区別が曖昧だと、委託先は「指示待ち」になり、自社は「任せつつもり」で止まる。

Microsoft Learn の shared responsibility はクラウドの例だが、ここでの読み方は分かりやすい。委託しても、少なくとも次は自社に残ると考えた方がよい。

委託しても自社に残るもの	具体例	なぜ残るか
アカウントと権限	誰にどこまで触らせるか	customer 側の access management に当たる
データと業務影響	何が重要で、止まると何が困るか	業務責任は自社にある
重要変更と継続判断	いつ変えるか、続けるか、やめるか	費用と受け入れは自社が決める

## 何を任せて、何を自社に残すか

外部委託がうまくいかない理由の一つは、任せる範囲が曖昧なことだ。曖昧なままでは、委託先は守備範囲を狭く解釈し、自社は広く期待する。そのずれが、障害時と変更時に表面化する。

任せやすいのは、たとえば次のような作業である。

- 定型的な監視
- 定常保守
- 定型設定
- 一次受付
- 一部のキittingや展開作業

一方で、自社に残すべきものもある。

- 何を優先するか
- 例外を認めるか
- 重要変更を承認するか
- 業務影響をどう評価するか
- どこまで費用をかけるか

たとえば、ネットワーク機器の保守を委託するのはよい。しかし、「設定変更の実作業」は委託しても、「業務影響を伴う変更をいつ行うか」「切り戻すかどうか」の最終判断まで委託先へ流してはいけない。そこは自社の業務都合と責任があるからだ。

また、ヘルプデスクの一次受付を委託することはできる。だが、「どの依頼を標準対応とし、どこから例外申請にするか」というルールは自社で握っておきたい。委託先にすべてを解釈させると、運用ルールそのものが外へ出てしまう。

外部委託をうまく使うとは、仕事を減らすことではなく、どの作業を外へ出し、どの判断を自社に残すかを明確にすることである。

線引きが曖昧になりやすいところは、次のように見ると整理しやすい。

任せやすい作業	自社に残す判断	線引きが曖昧だと起きること
定型監視、一次受付、キッキング	何を優先するか、どこから例外か	受付はされたが、順番が決まらない
定常保守、定型設定、更新作業	重要変更の承認、実施時刻、切り戻し判断	作業は進むが、業務影響を見誤る
定型レポート、月次報告	継続可否、費用許容、標準製品化の判断	数字はあるが、次の判断が決まらない

## 委託先に求める最低条件

委託先ごとに長大な評価票を作る必要はない。だが、最低限確認し続ける項目は必要である。NISTのC-SCRM Quick-Start GuideやSP 1326が示す通り、重要なのは、相手に対して最低限の理解を持ち、要求事項を定義し続けることだ。

第6章の段階では、委託先ごとに次の点が見えていればよい。

- 何を担当する相手か
- どこまでが対応範囲か
- 誰が窓口か
- 平常時と緊急時の連絡方法は何か

- 再委託はあるか
- どのデータや設定に触れるか
- どんな作業報告が返ってくるか

NIST SP 1326 は、supplier に対する due diligence を **minimum amount of understanding** と表現している。つまり、第6章で必要なのは完璧な監査票ではなく、委託先ごとに最低限の理解を持つことである。小さな会社なら、まず一枚の一覧に次を入れればよい。

項目	最低限入れたい内容	見えていないと困ること
役割	何を任せている相手か	誰に何を頼むか分からない
対応範囲	どこまでが対象か	そこは対象外で止まる
窓口	社内窓口、先方窓口、緊急連絡先	障害時の初動が遅れる
触れる対象	触るデータ、設定、アカウント	権限や説明責任が曖昧になる
再委託	有無、再委託先に渡る業務	視界の外で処理が進む
証拠	監査報告、認証、制度登録、説明資料	営業説明だけで判断してしまう
更新情報	更新時期、終了条件、作業報告の形	継続判断や終了作業で詰まる

ここで大切なのは、相手の会社案内を読むことではない。自社の運用に必要な情報が取れているかを見ることだ。たとえば、障害時の連絡窓口が営業担当しか分からない状態は危うい。設定変更を依頼した時、作業後に何が報告されるのか分からない状態も危うい。再委託があるのに、どこまで先へ情報が渡るのが見えていない状態も危うい。

委託先管理で必要なのは、完璧な評価ではなく、見えていない部分を減らすことである。

クラウドや SaaS では、営業説明だけでなく、第三者評価や制度上の登録状況も証拠になる。たとえば ISMAP は、政府が求めるセキュリティ要求を満たしたクラウドサービスを評価・登録する制度である。すべての会社が ISMAP を使うわけではないが、**相手の説明以外に何を証拠として見るか** という発想自体は第6章で持っておきたい。

## 連絡経路とエスカレーションを先に決める

障害時に最も無駄が出るのは、原因調査そのものより、「誰に連絡するか」で止まる時間である。平常時は気にならなくても、実際に止まった瞬間には、窓口の曖昧さがそのまま初動遅れになる。

少なくとも、次の点は先に決めておきたい。

- 一次窓口は誰か
- 緊急時の連絡先は何か
- 社内で判断を引き受ける人は誰か
- 複数ベンダーが絡む時に誰が主導するか

これも、口頭だけでなく一枚にしておくとう迷いにくい。

決めること	最低限の答え
一次窓口	最初に社内の誰が受けるか
緊急連絡	営業担当以外の緊急連絡先は何か
社内判断者	業務停止、切り戻し、社内周知を誰が決めるか
主導者	複数ベンダーが絡む時に、事実集約と次連絡を誰が回すか
記録場所	時刻、判断、依頼内容をどこへ残すか

ここで重要なのは、ベンダーごとに窓口を持つだけでは足りないということだ。自社側で、「まず自分が受けるのか」「総務が受けるのか」「社長への段階で上げるのか」を決めておく必要がある。

また、複数ベンダーが関わる場面では、押し付け合いが起きやすい。回線事業者、ネットワーク保守業者、クラウドベンダー、社内担当がそれぞれ別だと、「自社設備の問題ではない」「アプリ側の問題ではない」「回線側の調査待ち」となりやすい。そういう時に必要なのは、誰が正しいかをその場で決めることではない。まず誰が事実を集め、誰が暫定判断を出し、誰が次の連絡を回すかを決めることだ。

この章では詳細なインシデント手順までは扱わないが、少なくとも、障害時の連絡経路と承認経路はベンダー管理の一部だと理解しておきたい。

## 外部委託先のアクセス管理と承認

委託先管理で抜けやすいのが、外部委託先のアカウントと権限である。作業のたびに都度渡しているつもりでも、気づくと古いアカウントが残り、共有アカウントが使われ、誰が何をしたのか追えなくなる。

本書の詳しい権限管理は第8章で扱うが、第6章の段階では、まず次の考え方をもちたい。

- 共用ではなく、できるだけ個人単位で渡す
- 必要最小限の権限にする
- 付与と削除を記録する
- 契約終了時に必ず止める

CISA の MSP 向け助言も、ここをかなり具体的に書いている。MSP アカウントには MFA を求め、least privilege を適用し、委託先アカウントを管理対象だけへ絞り、使っていない時は無効化するべきとしている。第6章では、次の形で押さえると実務に落としやすい。

原則	実務での意味
個人単位	共有の管理者アカウントではなく、担当者単位で渡す
必要最小限	委託対象のシステムだけへ権限を絞る
MFA	委託先アカウントにも多要素認証を要求する
期限付き	作業時だけ広い権限を使い、常設しない
記録と停止	付与、変更、削除を記録し、契約終了時に止める

委託先だからまとめて一つの管理者アカウントでよい、という考え方は危うい。委託先の中でも担当者は変わる。契約終了後にそのアカウントが残れば、外部からの不要な入口になる。さらに、何か起きた時に、誰が行った操作か追えない。

また、ひとり情シスの現場では、「今すぐ見てもらうために、とりあえず広い権限を渡す」が起きやすい。だが、それを後で縮めるのは難しい。だから、最初に広く渡さない方がよい。定常的に必要な権限と、作業時だけ必要な権限は分けて考えたい。

外部委託先のアクセス管理は、相手を信用しないためのものではない。契約、担当者、作業内容が変わっても、入口を見失わないための管理である。

## 定例、レビュー、課題管理を回す

委託先監督というと、重い監査や細かいチェックリストを想像しやすい。だが、個人情報保護委員会の Q&A でも、立入検査そのものが必須とはされていない。個人データの内容や規模に応じて、口頭確認を含む適切な方法でよいとされている。小さな会社では、この考え方が重要である。

つまり、毎回大げさな監査をする必要はない。その代わりに、定例と確認項目を持つ。これが現実解になる。

たとえば、主要な委託先とは月一回か隔月で短い定例を持てばよい。確認するのは次のような項目で十分である。

- 先月の障害や問い合わせ
- 未解決課題
- 次回の変更予定
- 契約や更新が近いもの
- 権限やアカウントの見直しが必要なもの

ここで大切なのは、会議を長くすることではない。課題一覧を一つ持ち、期限と担当をはっきりさせることだ。委託先管理が崩れるのは、多くの場合、重大な障害よりも、小さな宿題が誰のものか分からなくなる時である。

また、更新前レビューも有効である。更新時は価格の話だけに寄りやすいが、実際には「この一年で何が詰まったか」「サポートは期待通りだったか」「課題は残ったか」を振り返る良い機会でもある。契約更新を、価格交渉だけの場にしなないことが大切だ。

PPC Q&A が示す通り、委託先監督は立入検査だけが方法ではない。小さな会社では、短い定例、口頭確認、作業報告、課題一覧でも十分に意味がある。逆に、何も確認しないまま年一回の更新だけ迎える方が危うい。

CISA の資料は、主要サプライヤを incident response や business continuity planning に含め、incident notification の protocol を明確にするよう勧めている。重要な委託先だけでも、**障害時に誰が何分以内にどう知らせるか** を普段から揃えておきたい。

## 再委託と個人データの扱いを見落とさない

委託先管理で特に見えにくいのが、再委託である。自社は A 社へ頼んでいるつもりでも、実際の運用や保守の一部は B 社や C 社が担っていることがある。この時に、「うちは A 社としか契約していないから関係ない」と考えるのは危ない。

個人情報保護委員会のガイドラインでも、必要かつ適切な監督を行っていない場合、再委託先の不適切な取扱いが元の委託元側の問題になり得ると示されている。つまり、再委託は自社の視界の外へ置いてよい話ではない。

第6章で最低限押さえないのは、次の三点である。

- 再委託があるか
- どの業務やデータが再委託されるか
- 再委託時に事前報告や承認が要るか

個人情報保護委員会のガイドラインが示しているように、再委託は **あるかどうか** だけでなく、どう監督するかまで見たい。最低限、次の形で整理しておくとうい。

確認すること	最低限の答え
再委託の有無	あるか、ないか
対象業務	何を再委託しているか
対象データ	どのデータや設定が渡るか
事前報告や承認	いつ、どの条件で知らせるか
監督方法	何の証跡や報告で確認するか
国外保管	どこに保管されるか、国外移転があるか

加えて、個人データや重要データを扱う委託では、どこに保管されるのか、国外移転があるのか、どのように削除されるのかも確認したい。ここで法制度の詳細解説までは行わないが、「委託した相手の先」まで見ないと説明に詰まる、という現場感だけははっきり持っておくべきである。

## 契約終了時に何を回収するか

ベンダー管理で最後に崩れやすいのは、終了時である。始める時には会議も資料も多いのに、終わる時は急に雑になる。だが、本当は逆である。終了時ほど、データ、権限、接続、手順の回収が必要になる。

最低でも、次の項目は確認したい。

- データ返却
- データ削除確認
- 管理者アカウント削除
- 接続経路の停止
- 手順書や設定情報の引き継ぎ
- 未解決課題や保守履歴の回収

終了時は、次の形で一つずつ潰した方が安全である。

回収項目	何を確認するか	見落とすと起きること
データ返却	必要な形式で出せたか	乗り換え後に使えない
データ削除確認	相手側で削除された証跡があるか	契約終了後もデータが残る
アカウントと鍵	管理者アカウント、APIキー、共有秘密が止まったか	終了後も入口が残る
接続経路	VPN、IP許可、外部連携が止まったか	サービス停止後も接続できない
引き継ぎ資料	手順書、設定値、保守履歴、未解決課題を受け取ったか	次の担当や新ベンダーが引き継げない

特に危ないのは、「サービスは止めたが、接続は残っている」「契約は終わったが、ベンダーアカウントが残っている」「データを出したつもりだが、必要な形式で取れていない」という状態である。これらは、終了時にしか気づきにくい。

第5章で解約条件やデータ持ち出しを確認したが、第6章ではそれを実際の終了作業へつなぐ。やめる時まで含めて運用である。

## 小さな会社で現実的に回るベンダー管理

ここまで読むと、委託先ごとに重い台帳や監査計画が必要に見えるかもしれない。だが、小さな会社で本当に必要なのは、見える形を一枚持つことだ。

現実的には、次の四つから始めればよい。

- 主要委託先の一覧を作る

- 各委託先の窓口と緊急連絡先を書く
- 外部アカウントを洗い出す
- 終了時の回収項目を決める

主要委託先の一覧には、少なくとも「会社名」「何を任せているか」「社内窓口」「先方窓口」「緊急連絡先」「更新時期」を入れておきたい。これだけでも、障害時と更新時の迷いはかなり減る。

さらに、委託先を重さで分けて考えると回しやすい。たとえば、基幹に近い委託先、日常運用に関わる委託先、単発に近い委託先では、同じ頻度で見なくてよい。重要な相手には定例を持ち、軽い相手は更新前確認だけにする。全部を同じ重さで管理しようとしなくても、ひとり情シスには重要である。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスが主要委託先を一枚にまとめている。ネットワーク保守、回線、クラウド基盤、SaaS 導入支援の四社について、社内窓口、先方窓口、緊急連絡先、任せている範囲、更新時期、外部アカウントを一覧にしている。月一回の短い定例では、未解決課題、今月の変更予定、権限見直し、次回更新を確認する。障害時は、まず社内一次窓口が受け、必要に応じてどの委託先へ上げるかを定める。契約終了時は、データ返却、接続停止、アカウント削除のチェックリストに沿って回収する。結果として、委託先がいても運用の主語は自社のままでいられる。

崩れた時の例では、各委託先の窓口が担当者のメールの中に埋もれている。障害時には、誰へ先に電話すべきかが分からず、回線業者と保守業者が互いに原因を押し付ける。ベンダーには共有の管理者アカウントを渡しており、担当交代も把握していない。契約終了後もアカウントが残り、データ削除の確認もない。委託したはずなのに、見えない不安と後始末がひとり情シスに積み上がる。

この差を生むのは、ベンダーの数でも会社の規模でもない。任せる範囲と見続ける項目を、自社側で持っているかどうかである。

## よくある失敗

第6章で特に避けたい失敗は五つある。

一つ目は、委託した瞬間に自社責任が薄くなると考えることだ。実際には、判断や説明責任は残る。

二つ目は、任せる範囲を曖昧にしたまま運用を始めることだ。曖昧さは、障害時と変更時に一気に問題になる。

三つ目は、窓口と緊急連絡先を整理しないことだ。初動の遅れは、技術不足より連絡不足から起きやすい。

四つ目は、外部アカウントを渡しっぱなしにすることだ。契約終了や担当変更のたびに危険が残る。

五つ目は、再委託と終了時の回収を軽く見ることだ。見えない相手と終わり方の雑さは、後から大きく効く。

## 最低限ここまではやる

第6章の段階では、次の五つができれば十分に前進である。

一つ目。主要委託先を一覧にできること。

二つ目。各委託先の窓口、緊急連絡先、任せている範囲を説明できること。

三つ目。外部委託先のアカウントと権限を把握できること。

四つ目。重要な委託先とは、短くても定例かレビューの場を持つこと。

五つ目。契約終了時の回収項目を持っていること。

これだけでも、外部委託はかなり安定する。ベンダー管理とは、相手を細かく縛ることではない。自社が見失ってはいけない責任を、見える形にして持ち続けることである。

### 確認したいこと

- 委託先ごとの責任分界を説明できる
- 障害時の連絡経路を決めている
- 外部委託先のアカウントを把握している
- 再委託の有無を確認している
- 定例やレビューの頻度を決めている
- 契約終了時の回収項目を持っている

### 今日、今週、後でやること

今日やることは三つでよい。まず、主要委託先を五社以内で書き出す。次に、それぞれについて、社内窓口、先方窓口、緊急連絡先、任せている範囲を書き添える。最後に、委託先へ渡しているアカウントや接続経路を洗い出す。

今週やることは、重要な委託先について、短いレビューの場を決めることだ。そこで、未解決課題、今後の変更、権限見直し、更新時期を確認できる形にする。

後で整えることは、再委託確認、終了時チェックリストの整備、主要委託先ごとの運用記録の蓄積である。全部を最初から完璧にしなくてよいが、見えないままにしないことが重要である。

## 第7章

## 要件定義、導入プロジェクト、変更管理

---

新しい勤怠システムを入れた。打刻はできる。申請もできる。ベンダーのデモ通りに動いている。だから導入は成功したはずだった。ところが月末になると、締め処理の流れが現場に伝わっていない。CSV の出力仕様が給与計算側と合わない。誰が問い合わせを受けるのかも曖昧で、結局ひとり情シスが全部を拾う。機能は足りていたのに、運用は回らない。

こうした詰まり方は、導入や変更を「実施できるかどうか」だけで進めた時に起きる。技術的に可能でも、目的、影響、戻し方、受け入れ方が曖昧なら、本番後に必ずどこかで詰まる。

第5章では、何を買ひ、どう契約するかを整理した。第6章では、委託先とどう付き合い続けるかを扱った。第7章では、その次の段階として、要件定義、導入プロジェクト、変更管理を一つの流れで扱う。ここで扱うのは大規模プロジェクト管理の一般論ではない。ひとり情シスが現場で直面する、小さな導入と小さな変更を安全に進めるための型である。

### 課題の整理と導入目的の明確化

導入や変更が崩れる最初の原因は、要件が甘いことではない。その前に、何を解決したいのかが曖昧なことである。

IPA の **重要情報を扱うシステムの要求策定ガイド** が、特性評価、問題・リスクと利便性の整理、必要な対策の選定という流れを取っているのは、いきなり対策や機能から入ると判断を誤りやすいからだ。まず整理すべきなのは、今どこで困っていて、誰が困っていて、何を变えたいのかである。

たとえば、勤怠システムを変えたい場合でも、本当の課題は次のどれかで全く意味が違う。

- 打刻漏れが多い
- 承認フローが遅い
- 給与計算への連携が手作業で重い
- テレワーク時の運用に合っていない
- 管理者しか設定変更できず属人化している

課題が違えば、必要な製品も、移行の重さも、導入後の確認点も変わる。にもかかわらず、「今のシステムが古いから変えたい」「新しい方が便利そうだから入れたい」だけで進めると、後で必ず論点が増える。

ここで最初に決めたいのは、次の点である。

- 何を解決したいのか
- 誰にとっての改善なのか
- 何ををもって成功とするのか

IPA のアジャイル開発版モデル契約が、契約前のチェックリストで **目的・ゴール** **ステークホルダー** **完了基準** **品質基準** を確認しているのも同じ理由である。開発方式にかかわらず、導入前に終わり方が見えていない案件は崩れやすい。

最初は、次の形で一枚にしておくとおぼれにくい。

項目	最初に書くこと	曖昧だと起きること
課題	今どこで何が詰まっているか	要望が後から増える
対象者	誰の業務や負担を変えたいか	使う人の確認が抜ける
成功条件	何ができれば成功と言うか	一応入ったから完了になる
制約	いつまでに、何を止めずに進めるか	現実的でない計画になる

この三つが決まると、導入や変更はかなり楽になる。逆にここが曖昧だと、途中で要望が膨らみ、ベンダーとの会話もぶれ、受け入れ確認も曖昧になる。

## 機能要件と非機能要件を分けて考える

目的が見えたら、次は要件である。ここで重要なのは、機能要件と非機能要件を分けて考えることだ。

IPA の非機能要求グレードが、要求には機能要求と非機能要求があり、非機能要求には要件定義上の課題があると整理しているのは、実務でもこの区別が崩れやすいからである。欲しい機能は言いやすい。だが、止まりにくさ、速さ、ログ、権限、バックアップ、保守性は後回しになりやすい。

機能要件とは、何ができるかである。たとえば、承認フローが二段階で組める、CSV出力ができる、モバイルから打刻できる、といったものだ。

非機能要件とは、どう支えられるかである。たとえば、次のような点だ。

- どのくらい止まりにくい
- どのくらいの応答速度が必要か
- どこまでログを残せるか
- 権限をどこまで細かく分けられるか
- バックアップや復元はどうなっているか
- 追加設定や保守を誰がどう行うか

ひとり情シスの現場では、後で苦しくなるのは多くの場合こちらである。機能は足りているのに、管理者権限が雑、操作ログが弱い、保守のたびにベンダー依頼が必要、障害時の切り分けが難しい。こうした詰まり方は、導入時に非機能を言葉にしていなかったことから起きる。

非機能を全部厳密に定義する必要はない。だが、少なくとも重要項目から順に確認する考え方は持ちたい。可用性、権限、ログ、バックアップ、保守性。このあたりは、小さな会社でも先に見ておく価値が高い。

IPA の非機能要求グレードが、重要な項目から順に要求レベルを確認する使い方を示しているのも参考になる。第7章の段階なら、まず次を順に見ればよい。

非機能項目	まず確認したいこと	抜けると起きやすいこと
可用性	止められない時間帯はいつか	本番後に業務停止へつながらず
権限	管理者と利用者をどう分けるか	権限が広すぎて運用事故が起きる
ログ	どこまで残し、誰が見られるか	原因追跡や説明ができない
バックアップ	戻せる範囲と時間はどうか	障害時に戻せない
保守性	設定変更や追加作業を誰が行うか	ベンダー依存と属人化が進む

## ステークホルダーと役割分担を決める

導入や変更は、技術担当だけでは終わらない。利用部門、承認者、ベンダー、経営、総務、経理など、関係者が必ずいる。ここを曖昧にすると、途中で「それは聞いていない」「誰が決めるのか分からない」が始まる。

IPA のアジャイル開発版モデル契約が、契約前チェックリストの中で、目的・ゴール、ステークホルダー、ビジョン、初期計画、完了基準、品質基準、役割分担の理解を確認しているのは、この認識ずれが失敗の大きな原因だからである。これはアジャイル開発に限らない。SaaS 導入でも設定変更でも同じだ。

第7章で最低限決めたい役割は、次の通りである。

- 誰が利用部門の代表か
- 誰が最終承認するか

- 誰が実施するか
- 誰が本番実施の判断をするか
- 誰が問い合わせを受けるか
- 誰が不具合時の判断を引き受けるか

役割は、人数が少なくても分けて考えた方がよい。

役割	最低限の責任	同じ人に寄せすぎると起きること
利用部門代表	業務要件と受け入れ確認	現場の使い勝手が抜ける
承認者	費用、優先順位、本番可否の判断	誰も最終判断を引き受けない
実施担当	設定、移行、作業実行	実作業と承認が混ざる
本番判断者	当日 go / no-go を決める	失敗時にその場判断になる
問い合わせ窓口	導入後の受け先を持つ	ひとり情シスへ全部流れる
不具合時判断者	障害時の暫定判断と切り戻し判断	不具合発生時に判断が止まる

ひとり情シスが全部を背負いがちなのは分かる。だが、利用部門の要件確認や受け入れ判断まで一人でやると、後で必ず「現場の使い勝手が違う」という話になる。使う人の確認と、承認する人の判断は、できる限り分けたい。

## 導入計画、移行計画、切り戻し計画

本番移行の失敗は、当日発生するようになって、実際にはその前の計画不足で起きる。ここで分けて考えたいのが、導入計画、移行計画、切り戻し計画である。

導入計画は、全体の段取りである。いつ準備し、いつ試し、いつ本番に入れるかを決める。

移行計画は、何をどの順で移すかである。データ、設定、アカウント、周辺手順のどこまでが対象かを明確にする。

切り戻し計画は、失敗した時にどう戻すかである。ここがないと、本番当日に「このまま続けるしかない」という危険な判断に追い込まれる。

たとえば Wi-Fi 設定変更なら、導入計画は「準備、検証、切り替え、確認」の流れでよい。移行計画では、対象拠点、対象 SSID、対象端末、影響時間帯を整理する。切り戻し計画では、「何分以内に接続不良が一定数を超えたら旧設定へ戻す」といった条件を先に置く。

切り戻しは、単に「元に戻す」と書けばよいわけではない。何を基準に戻すのか、誰が戻す判断をするのか、戻した後何を確認するのかまで必要である。

また、導入や変更の計画では、連絡も計画に含めるべきである。誰に事前通知し、実施中にどう伝え、失敗時に誰へ上げるか。これがないと、技術的には小さい変更でも、現場から見ると突然止まっただけになる。

三つの計画は、似ているようで役割が違う。

計画	決めること	抜けると起きること
導入計画	準備、検証、本番までの全体段取り	当日までに準備不足が残る
移行計画	何をどの順で移すか	データや設定の抜け漏れが出る
切り戻し計画	どの条件で戻し、誰が決めるか	失敗しても続けるしかなくなる

## 本番導入と受け入れ確認

本番投入は終わりではない。ここで大事なのは、「入れた」と「使える」を分けることである。

本番前には、少なくとも次を確認したい。

- 対象は正しいか
- 事前バックアップや退避は終わっているか
- 関係者への連絡は済んでいるか
- 切り戻し条件と手順は確認済みか
- 当日の連絡先はそろっているか

本番後には、動作確認だけでなく、受け入れ確認が必要になる。動作確認は、技術的に動くかを見る。受け入れ確認は、業務として使えるかを見る。この二つは違う。

たとえば勤怠システムなら、ログインできる、打刻できる、申請できる、だけでは足りない。締め処理が正しく回るか、承認ルートが運用に合うか、給与計算へ必要なデータが出るか、現場の問い合わせに答えられるかまで見たい。

ここで重要なのが、受け入れ基準を先に置くことである。アジャイル開発版モデル契約が完了基準や品質基準の明確化を重視しているのも同じ理由だ。終わり方が曖昧だと、「とりあえず本番に入ったから完了」になってしまう。

動作確認と受け入れ確認は、次のように分けて考える。

確認の種類	何を見るか	例
動作確認	技術的に動くか	ログイン、保存、出力、連携が動く

確認の種類	何を見るか	例
受け入れ確認	業務として回るか	締め処理、承認フロー、問い合わせ対応が回る
完了判断	何をもって終わるか	重大不具合なし、主要部門確認済み、引き継ぎ完了

## 運用引き継ぎまでやって初めて終わる

導入後に特に起きやすい失敗は、「本番開始後の運用」が空白になることだ。ベンダーの作業は終わった。画面も動く。だが、社内では誰が何をするか決まっていない。これでは導入完了とは言えない。

少なくとも、次のようなものは残したい。

- 手順書
- 問い合わせ窓口
- 標準設定
- 追加変更の流れ
- 不具合時の連絡先
- 保守や監視の範囲

運用へ渡す時は、次を残しておくで空白が減る。

残すもの	最低限の中身	ないと起きること
手順書	日常操作、定例作業、追加変更方法	ひとり情シスしか回せない
問い合わせ先	社内窓口、ベンダー窓口、緊急連絡先	本番後の質問が迷子になる
標準設定	初期値、例外の扱い、管理権限	変更が毎回ばらつく

残すもの	最低限の中身	ないと起きること
保守範囲	誰が監視し、誰が直すか	障害時に押し付け合いになる
既知課題	今残っている不具合と回避策	同じ問い合わせが繰り返される

本書では後の章で運用対象ごとの詳細を扱うが、第7章の段階では、「導入したものが日常運用へ渡る形になっているか」を見る。これがないと、ひとり情シスだけが背景を知っていて、現場には何も残らない。

導入をうまく終わるとは、設定作業を終えることではない。後から別の人が見ても回せる状態にすることである。

## 変更管理を日常運用に組み込む

変更管理という言葉を聞くと、大規模システムや厳格な承認プロセスを想像しやすい。だが、NIST SP 800-128 が configuration management を、リスクを最小化しつつ業務上必要な機能を支える discipline としているように、本質はもっと基本的である。変えるなら、何を、なぜ、どこへ、どう戻せるかを持っておくことだ。

ひとり情シスの現場で対象になる変更は多い。

- MFA の強制化
- 権限設計の変更
- Wi-Fi 設定変更
- 端末標準イメージの更新
- 新しい SaaS の全社展開

これらを口頭で進めると、いつ変えたのか、どこまで変えたのか、誰が承認したのか、問題が出たらどう戻すのかが残らない。だから、日常の小変更でも、最低限の変更票を持ちたい。

最低限必要なのは、次の項目である。

- 変更の目的
- 変更対象
- 影響範囲
- 実施日時
- 実施担当
- 競合する変更や障害の有無
- 戻し方
- 実施後確認

Atlassian の change management 関連資料でも、Affected services Planned start Planned end を持ち、同時時間帯の他変更や open incidents を見てリスクを判断する考え方が示されている。小さな会社でも、これを簡略化して使えばよい。少なくとも、「どこに影響する変更か」「いつやるか」「今ほかに不安定なことはないか」は見たい。

変更票は、最低限次の形でよい。

項目	何を書くか
目的	なぜ変えるか
対象	どのサービス、端末、設定を変えるか
影響範囲	どの部門、利用者、時間帯に影響するか
実施予定	予定開始、予定終了
実施担当	誰が行うか

項目	何を書くか
競合確認	同時間帯の他変更、障害、締め作業の有無
戻し方	どの条件で、どう戻すか
実施後確認	何を確認して完了とするか

## 標準変更、個別承認変更、緊急変更を分ける

すべての変更を同じ重さで扱う必要はない。むしろ、全部を重くすると続かない。ここで役立つのが、変更の種類を分ける考え方である。

Atlassian の資料でも、standard change は事前承認済みの変更として扱われている。つまり、低リスクで、手順が固まっていて、繰り返し実施する変更は、毎回一から重い承認を取らなくてもよい。

第7章では、次の三つに分けて考えると現実的である。

- 標準変更
  - 手順が固まっている
  - 低リスク
  - 事前承認済み
- 個別承認変更
  - 影響が読み切れない
  - 利用部門確認や承認が必要
- 緊急変更
  - 障害回避や緊急対処のため即時性が高い
  - 事後記録と事後レビューが重要

たとえば、定例の PC 初期設定更新や、手順化された定型設定変更は標準変更にしやすい。一方で、全社 MFA 強制化や基幹に近い設定変更は個別承認変更になる。障害回避のための一時設定変更は緊急変更になる。

この区別があるだけで、現場はかなり回しやすくなる。重く扱うべき変更だけを重くし、それ以外は標準化する。第3章で扱った標準化の原則が、ここでもそのまま効く。

判断に迷う時は、次の表で十分である。

変更の種類	条件	扱い方
標準変更	低リスク、手順固定、繰り返し、事前承認済み	簡易記録で回す
個別承認変更	影響が広い、読み切れない、利用部門確認が必要	事前承認と受け入れ確認を入れる
緊急変更	障害回避、脆弱性対応など即時性が高い	まず実施し、事後記録と事後レビューを残す

## 小さな会社で現実的に回るやり方

ここまで読むと、プロジェクト管理表や change calendar や会議体が多く必要に見えるかもしれない。だが、小さな会社ではもっと軽くてよい。必要なのは、変更と導入の型を一つ持つことだ。

現実的には、まず次の四つから始めればよい。

- 導入目的を一文で書く
- 変更票に切り戻し欄を入れる
- 受け入れ基準を先に書く
- 変更予定をカレンダーで見える化する

たとえば共有スプレッドシートでもよい。変更一覧に「何を変えるか」「いつやるか」「誰がやるか」「戻し方」「確認結果」を残すだけでも、口頭変更よりはるかに安定する。

また、利用部門との確認は、長い会議でなくてよい。変更前に五分でも、「何が変わるか」「何を確認してほしいか」を共有するだけで、受け入れの質はかなり上がる。

変更予定をカレンダーで見える化するのも効果が高い。Atlassian が change calendar で日、週、月単位の scheduled changes を見る考え方を示している通り、同じ日に大きな変更を重ねないだけでも事故は減る。小さな会社なら、共有カレンダーやスプレッドシートで十分である。

## 通常時の例と、崩れた時の例

通常時の例では、全社 MFA 強制化を進める前に、ひとり情シスが目的を一文で書く。「管理者権限を含むアカウントの不正利用リスクを下げるため、対象者を段階的に MFA 必須へ切り替える」である。その上で、対象者、例外端末、問い合わせ増加、切り戻し条件、当日の連絡先を整理する。受け入れ基準としては、「対象者が翌営業日までにログインできる」「管理者アカウントの MFA が有効」「主要業務で重大な支障がない」を置く。導入後は、問い合わせ対応手順を残し、標準運用へ渡す。結果として、変更が一度のイベントで終わらず、日常運用に接続される。

崩れた時の例では、ベンダーの勧めで新しい勤怠システムを急いで入れる。比較は機能中心で、締め処理、CSV 出力、権限、問い合わせ運用は深く見ていない。移行当日はデータ投入に追われ、うまくいかなければその場で考えるしかない。受け入れ基準もなく、利用部門は「一応動くなら開始しましょう」となる。本番後に細かな不整合と問い合わせが噴き出し、ひとり情シスは導入後の火消しを続けることになる。

この差を生むのは、プロジェクトの規模ではない。変える前に、目的、影響、戻し方、受け入れ方を言葉にしたかどうかである。

## よくある失敗

第7章で特に避けたい失敗は五つある。

一つ目は、課題整理をせずに製品や設定の話から始めることだ。これでは要件がすぐぶれる。

二つ目は、機能要件だけを見て非機能要件を後回しにすることだ。後で運用が苦しくなるのはたいていこちらである。

三つ目は、移行計画と切り戻し計画を分けて考えないことだ。失敗時に戻せない変更は危うい。

四つ目は、受け入れ確認を「動いたかどうか」だけで済ませることだ。業務として使えるかを見る必要がある。

五つ目は、日常変更を記録せず、口頭やチャットだけで進めることだ。後で原因も責任も追えなくなる。

## 最低限ここまではやる

第7章の段階では、次の五つができれば十分に前進である。

一つ目。導入目的を一文で説明できること。

二つ目。機能要件と非機能要件を分けて書けること。

三つ目。移行計画と切り戻し計画を持てること。

四つ目。受け入れ基準を先に決めていること。

五つ目。日常の変更について、最低限の記録を残していること。

これだけでも、導入と変更はかなり安定する。ひとり情シスの仕事を守るとは、全部を慎重に遅くすることではない。変える時に、必要な確認だけは先に済ませることである。

### 確認したいこと

- 導入目的を一文で説明できる
- 機能要件と非機能要件を分けている
- 移行計画と切り戻し計画を持っている
- 受け入れ基準を決めている
- 導入後の運用引き継ぎを考えている
- 日常変更を記録している

### 今日、今週、後でやること

今日やることは三つでよい。まず、今進んでいる変更や導入を一件選び、その目的を一文で書く。次に、その案件について、非機能要件を三つだけ書き足す。最後に、切り戻し条件と切り戻し手順を一行でも入れる。

今週やることは、変更票や導入メモのひな型を作ることだ。そこに、目的、対象、影響、実施日時、戻し方、確認結果の欄を入れる。これだけで、場当たり変更はかなり減る。

後で整えることは、標準変更の整理、変更予定のカレンダー化、導入後ナレッジの蓄積である。全部を最初から制度化する必要はないが、口頭変更のままにしないことが重要である。

第III部

# 日常運用の基盤

人、端末、SaaS、データを崩れにくく回す。

第8章～第16章

## 第8章

## アカウント、認証、権限管理

---

退職者のメールは止めた。社用 PC も返却された。だからアカウント対応は終わったはずだった。ところが数週間後、共有フォルダのアクセス履歴を見ると、その人が所属していたプロジェクトの権限がまだ残っている。さらに調べると、プロジェクト管理ツール、経費精算、ファイル共有にも同じような残り方があった。加えて、設定変更は共有の `admin@` アカウントで行われていたため、誰がどこを変えたのかも追えない。

こうした問題は、アカウント管理を「作る」「消す」という作業で捉えた時に起きる。実際には、それだけでは足りない。誰として扱うか。どう本人確認するか。どこまで入れるか。後で追えるか。この四つがそろって初めて、アカウント管理が回り始める。

第7章では、導入や変更を安全に進めるための型を整理した。第8章では、その変更のたびに必ず論点になるアカウント、認証、権限管理を扱う。ここで扱うのは、製品ごとの設定手順ではない。小さな会社でも崩れにくい、IAM の最小原則である。

### ID、認証、権限、監査を分けて捉える

まず言葉を分けておきたい。ここが混ざると、議論がすぐ曖昧になる。

ID とは、誰として扱うかである。社員本人の個人アカウントなのか、管理者用の別アカウントなのか、連携用のサービスアカウントなのか。まず主体を切り分ける必要がある。

認証とは、その ID の本人性をどう確かめるかである。パスワードだけなのか、MFA を使うのか、SSO でまとめるのか、セキュリティキーやパスキーまで使うのか。ここは「ログイン方法」の話である。

権限とは、認証が通ったあとにどこまで入れるかである。閲覧だけか、編集までか、管理者か、監査ログまで見られるか。認証と権限は別物である。本人確認ができて、何でもできてよいわけではない。

監査証跡とは、誰がいつ何をしたかを後で追えるようにすることである。管理者権限の付与、グループ変更、MFA の再設定、共有設定の変更、緊急用アカウントの利用などが追えなければ、問題が起きた時に原因も責任も分からない。

この四つを一体で見ると、アカウント管理の見え方が変わる。たとえば退職者対応は、メール停止だけでは終わらない。本人の ID を無効化し、関連する認証手段を止め、残っている権限を外し、実施記録を残して初めて終わる。第9章ではその流れを詳しく扱うが、第8章では、その前提となる考え方を固めたい。

最初にこの表で切り分けておくと、話がぶれにくい。

項目	何を決めるか	例	抜けると起きること
ID	誰として扱うか	個人、管理者、共有、サービス	主体が曖昧になる
認証	本人性をどう確かめるか	パスワード、MFA、SSO、パスキー	なりすましやロックアウトが起きる
権限	認証後にどこまで入れるか	閲覧、編集、管理者、監査	権限過多や残存権限が出る
監査証跡	後で何を追えるようにするか	権限変更、MFA再設定、緊急用利用	原因と責任を追えない

## パスワード、MFA、SSO を一体で考える

認証の話になると、今でも「強いパスワードを定期変更する」という発想に寄りがちだ。だが、NIST の現在の整理はかなり違う。長く、固有で、管理しやすいパスワードを使い、根拠なく頻繁な変更を強制しない。むしろ、長いパスフレーズ、パスワードマネージャー、コピーアンドペーストの許容、SSO のような利用者負荷を減らす設計を重視している。

実務で大事なものは二つである。一つ目は、同じパスワードを複数サービスで使い回さないこと。二つ目は、パスワードだけで守ろうとしないことだ。IPA も、不正ログイン被害の増加に対して、ID とパスワードだけでは弱く、パスキーや多要素認証の導入を勧めている。漏えい、推測、リスト型攻撃、フィッシング。こうした現実を考えると、特に管理者権限や重要データへ触るアカウントでは、MFA は前提と考えた方がよい。

ただし、MFA を入れれば何でもよいわけではない。NIST が AAL2 でフィッシング耐性のある選択肢を提供すべきとしているように、認証アプリは SMS よりリスクプロファイルが異なり一定の堅さがある一方、セキュリティキーやパスキーはフィッシング自体を防げる点で一段強い。ひとり情シスの現場でも、最低でも管理者アカウントから優先的に、可能ならフィッシング耐性の高い方式へ寄せたい。

ここで忘れやすいのが、復旧手段である。MFA は強いが、端末故障、機種変更、紛失、回線障害で自社管理者が入れなくなる事故も起きる。強くすること、復旧できることは両立させなければならない。この論点は後で改めて扱う。

SSO も、便利だから入れるだけではもったいない。NIST が federation によって認証負荷を下げられると示している通り、SSO は利用者の手間を減らす。そのうえで、ひとり情シスにとってはもっと重要な価値がある。認証ポリシー、

MFA、無効化、ログ、棚卸しを一か所へ寄せやすくなることだ。SaaSが増えるほど、アプリごとに個別ログインを持つ運用は崩れやすい。全てを一度にSSO化できなくても、全社利用SaaS、管理者権限があるSaaS、退職時に漏らしたくないSaaSから順に寄せる価値は高い。

ただし、SSOに寄せるほど、IdP側の管理者権限と復旧手段はより重要になる。認証を集中させるとは、守るべき中枢を一つに絞ることもあるからだ。

認証は、次の形で整理すると古い慣習を外しやすい。

論点	今の整理	ありがちな誤り
パスワード	長く、固有で、管理しやすくする	複雑さルールだけを増やす
定期変更	漏えい時など根拠がある時に行う	根拠なく定期変更だけを強制する
パスワード管理	パスワードマネージャーやSSOを使う	覚えやすさ優先で使い回す
MFA	重要アカウントでは前提で考える	パスワードだけで守ろうとする
認証方式	可能なら認証アプリ、セキュリティキー、パスキーへ寄せる	SMSだけで十分と考える
復旧手段	予備キー、バックアップコード、緊急用アカウントを持つ	強化だけして自社が入れなくなる

## 権限設計は個人直付けではなく役割で行う

権限管理が崩れる一番の原因は、個人ごとの例外設定が増え続けることだ。入社時に少し足す。異動時に一部だけ残す。急ぎの依頼で直接付与する。これを繰り返すと、半年後には誰がなぜその権限を持っているのか分からなくなる。

ここで必要なのが、役割で考えることだ。CISもMicrosoftも、最小権限を個人への我慢ではなく、役割、範囲、期間で整理する方向を示している。つまり、「営業担当ならここまで」「経理担当ならここまで」「情シス管理者でも日常利用はここまで」という標準形を先に作る。

現場では、次の三つで考えると分かりやすい。

- 役割

- 一般利用者
- 部門責任者
- 経理や人事など機微情報を扱う担当者
- 情シス管理者
- 外部委託先

- 範囲

- どのアプリか
- どのフォルダか
- どの組織、どのプロジェクトか

- 期間

- 恒久か
- プロジェクト期間限定か
- 緊急対応中だけか

この考え方にすると、権限付与は人ごとの職人芸ではなく、グループやロールへの所属管理へ変わる。異動時も、個別権限を一つずつ見直すより、所属グループを変える方が速く、漏れにくい。第3章で扱った標準化が、ここでもそのまま効く。

役割、範囲、期間は、次のように切ると扱いやすい。

観点	何で切るか	例
役割	何の仕事か	一般利用者、部門責任者、 経理、人事、情シス、外部 委託先
範囲	どこまで触れるか	全社、部門、プロジェクト、 特定フォルダ、特定ア プリ
期間	いつまで必要か	恒久、プロジェクト期間、 障害対応中だけ

最小権限という言葉は、厳しさだけで理解すると誤る。これは事故が起きた時の被害面積を減らす設計であり、同時に、従業員のプライバシーや業務分掌を守る設計でもある。Google が管理者権限を必要最小限へ絞るよう整理しているのも、そのためである。見られなくてよい情報は見られない方がよい。

また、権限は永続でなくてよい。システムが対応していれば、申請時だけの昇格、一定時間で失効する付与、承認付きの一時権限を使うと崩れにくい。そこまで高度な機能がなくても、「この権限は今月末で見直す」と期限を持たせるだけで、放置される権限は減る。

## 管理者権限を日常利用と分離する

通常利用アカウントと管理者アカウントを分ける。これは第8章の中でも特に重要な原則である。

管理者権限を持つアカウントで、メールを読み、Web を見て、日常の事務作業をするのは危うい。万一そのセッションや端末が侵害された時、被害は一気に広がる。Microsoft も Google も、日常利用と高権限利用を分離することを明確に勧めている。ひとり情シスであっても、この原則は変わらない。

理想は、次のように分けることだ。

- 通常利用アカウント
  - メール
  - チャット
  - 文書作成
  - 一般的な業務利用
- 管理者アカウント
  - 設定変更
  - 権限付与
  - 監査ログ確認
  - 管理画面操作

加えて、最上位権限は少人数へ絞りたい。Microsoft は Global Administrator を 5 人未満に抑えることを勧めている。製品名は違ってても考え方は同じで、何でもできる権限を広く配らない方がよい。

ひとり情シスの現場では、「自分一人しかいないから分けても意味がない」と思いやすい。だが、意味はある。一つは、通常作業中の事故面積を減らせること。もう一つは、監査証跡が見やすくなることだ。通常ログインと管理操作が混ざらないだけでも、後で追いやすくなる。

もし利用しているサービスが対応しているなら、常時管理者にするのではなく、必要時だけ昇格できる仕組みが望ましい。PIM のような機能がなくても、管理者権限を持つ人数を絞り、付与や変更の記録を残すだけでかなり違う。

Microsoft と Google の実務指針をまとめると、最低限次を持ちたい。

項目	実務での考え方
通常利用と管理者の分離	同じ人でもアカウントは分ける
最上位権限の人数	何でもできる権限は少人数へ絞る
最上位権限の多重化	一人だけにしない
MFA	管理者アカウントでは前提にする
日常利用	super admin / Global Admin でメールや Web をしない
記録	権限付与、剥奪、昇格、緊急利用を追えるようにする

## 緊急用アカウントと復旧手段を持つ

認証を強くするほど、設計を誤った時のロックアウトも重くなる。ここは実務で非常に重要だが、見落とされやすい。

Google は 2SV 保護アカウントの復旧にあたり、予備のセキュリティキー、バックアップコード、追加の super administrator（最上位管理者）を勧めている。Microsoft も緊急用アカウントの考え方を明確に示している。要するに、最後の復旧手段まで同じ前提で縛ってはいけないということだ。

小さな会社でも、次のものは持ちたい。

- 緊急用の管理者アカウント
- 予備の認証手段
  - 予備セキュリティキー
  - バックアップコード
  - 回復用連絡先
- 利用時の通知
- 保管場所と取り出し手順

緊急用アカウントは、日常利用しない。個人のメール用途にも使わない。平時は封印し、使ったら必ず記録し、なぜ使ったかを残す。利用時に経営者や責任者へ通知が飛ぶ運用にできるとさらによい。

重要なのは、MFA 強制や条件付きアクセスの変更、SSO 切り替えのような大きな認証変更を行う時に、緊急用アカウントまで同じ変更で巻き込まないことである。最後の一枚の鍵まで同時に締めると、自社が中へ入れなくなる。

また、復旧手段は持っているだけでは足りない。年に一度でもよいので、予備キーの所在、バックアップコードの更新、緊急用アカウントのサインイン可否を確認したい。災害備品と同じで、必要な時に使えなければ意味がない。

復旧手段は、次を一式で持つと崩れにくい。

用意するもの	目的	注意点
緊急用管理者アカウント	通常経路が使えない時の最後の入口	日常利用しない
予備セキュリティキー	紛失、故障、機種変更時の予備	保管場所を決める
バックアップコード	認証器が使えない時の一時復旧	利用後は更新する
追加の最上位管理者	一人が締め出されても復旧できるようにする	通常利用と混ぜない
利用通知	緊急利用を見逃さない	責任者へ高優先度で知らせる
定期確認	必要時に本当に使える状態を保つ	持っているだけで満足しない

## 共有アカウントとサービスアカウントを例外管理する

原則は個人アカウントである。誰が何をしたかを追うには、それしかない。

共有アカウントは、一見便利だ。代表メール、店舗端末、共用 PC、機器管理画面など、事情がある場面もある。だが、共有アカウントは監査証跡を壊しやすく、退職や異動の影響範囲も見えにくい。Google が共有管理者アカウントを避けるよう勧めているのも当然である。

やむを得ず共有アカウントを残すなら、少なくとも次を持たせたい。

- 用途
- 所有者
- 利用者の範囲
- 認証情報の保管場所
- MFAをどう扱うか
- パスワード変更時期
- 廃止できる見込み

特に管理者権限を共有アカウントで持つのは避けたい。共有の代表メールが必要でも、管理者権限は個人アカウントへ委任した方がよい場面が多い。

サービスアカウントも放置されやすい。API 連携、バックアップ、ジョブ実行、スクリプト、監視。便利な一方で、人の異動と無関係に生き残るため、見直しから漏れやすい。CISもサービスアカウントの棚卸しを独立論点にしている。

サービスアカウントには、少なくとも次を残したい。

- 何のためのアカウントか
- どのシステムが使うか
- 所有者は誰か
- 秘密情報はどこに保管されているか
- 権限は何か

- 次の見直し日はいつか

共有アカウントとサービスアカウントは、次のように台帳へ残したい。

種別	最低限持ちたい項目	見落とすと起きること
共有アカウント	用途、所有者、利用者範囲、認証情報の保管場所、MFAの扱い、廃止見込み	誰が使ったか追えない
サービスアカウント	用途、所有者、利用システム、秘密情報の保管場所、権限、見直し日	人の異動と無関係に残り続ける

人が日常利用するアカウントを、そのまま自動処理へ流用するのは避けたい。逆に、サービスアカウントで人が手作業ログインするのも避けたい。主体が違うからである。

## 定期棚卸しと監査証跡を回す

権限は、放っておくと必ず増える。異動、兼務、臨時対応、外部委託、導入時の暫定設定。だから棚卸しが必要になる。

ここで重要なのは、棚卸しを単なる一覧確認にしないことだ。Microsoft の access review が、対象、レビュー担当、周期、未回答時の扱い、結果反映まで設計させているのは、見るだけでは元に戻らないからである。

棚卸しでは、少なくとも次を決めたい。

- 何を見直すか
  - 管理者権限
  - グループ所属
  - 共有フォルダ権限

- 外部委託先権限
- サービスアカウント
- 誰が判断するか
  - システム所有者
  - 部門責任者
  - 情シス
- いつやるか
  - 四半期
  - 半年
  - 年次
- 未回答ならどうするか
- 不要と判断した権限をいつ外すか
- 記録をどこへ残すか

棚卸しは、次の形で回すと **見ただけ** で終わりにくい。

項目	最低限の決め方
対象	管理者、グループ、共有フォルダ、外部委託先、サービスアカウント
判断者	システム所有者、部門責任者、情シスの誰が見るか
周期	四半期、半年、年次のどれか
未回答時の扱い	そのまま維持するか、外すか、再確認するか
結果反映	不要権限をいつ削除するか

項目	最低限の決め方
記録	何を見て、どう判断し、いつ反映したかを残す

最初から厳密な仕組みは要らない。だが、周期は決めたい。実務では、管理者権限、外部委託先、機微データアクセスは四半期ごと、一般的な権限は半年ごとから始めるのが現実的である。CIS は休眠アカウント無効化の目安として 45 日を示しているが、実務では休職や季節業務もあるため、自社基準として何日で確認対象にするかを先に決めておく方が回しやすい。

監査証跡として特に見たいのは、次のようなものだ。

- 管理者権限の付与と剥奪
- グループ所属変更
- MFA の再設定
- 共有設定変更
- 緊急用アカウントの利用
- 失敗したログインや異常なサインイン

共有管理者アカウントを使っていると、ここが全部ぼやける。誰がやったかではなく、「その共有 ID で誰かがやった」までしか分からない。これは運用上かなり弱い。

## 小さな会社で現実的に回るやり方

ここまで読むと、専用の IAM 製品や高度な統制基盤が必要に見えるかもしれない。だが、小さな会社ではもっと単純でよい。まずは台帳と周期を持つことから始めればよい。

最低限、一覧にしたい項目は次の通りである。

- 氏名または所有者
- アカウント名
- 種別
  - 個人
  - 管理者
  - 共有
  - サービス
- 対象システム
- 所属グループまたは役割
- MFA 状態
- 復旧手段の有無
- 作成日
- 最終見直し日
- 終了予定日

この一覧だけでも、かなり見え方が変わる。特に、管理者アカウント、共有アカウント、サービスアカウントを分けて見られるだけで、危ない場所が浮く。

運用としては、次の形が始めやすい。

- 管理者権限一覧は四半期ごとに見直す
- 一般権限一覧は半年ごとに見直す
- 共有アカウントは用途と所有者を毎回確認する
- 休眠アカウントは月次で抽出する
- 変更管理の記録と権限変更をひも付ける

全ての SaaS をすぐ SSO 化できなくても、SSO 非対応アプリを一覧へ明示し、個別に無効化が必要だと分かる状態にしておけば、退職時や棚卸し時に漏れにくくなる。

## 通常時の例と、崩れた時の例

通常時の例では、新しいプロジェクト管理ツールを導入する際に、ひとり情シスが最初に個人アカウント原則を置く。ログインは SSO へ寄せ、部門ごとのグループで基本権限を配る。管理者操作は通常利用アカウントとは別の管理者アカウントで行い、管理者権限は四半期ごとに見直す。一般権限は半年ごとに棚卸しし、外部委託先は契約更新のたびに確認する。MFA を強制する前に、予備キーとバックアップコード、緊急用管理者アカウントも整える。異動が起きても、グループを変えるだけで大半の調整が終わる。誰が設定を変えたかも追える。

崩れた時の例では、各 SaaS に個別ログインがあり、権限は依頼のたびに直接付与している。管理者権限は共有の `admin@` で回し、複数人が同じ認証情報を知っている。MFA は慌てて有効化したが、予備手段は準備していない。ある日、役員のスマートフォン故障をきっかけに管理者が入れなくなり、さらに過去の退職者権限も残っていたことが判明する。調べようにも、共有アカウント運用のため誰が何を変えたか追えない。問題は個別の設定ミスではなく、アカウント管理を統制として見ていなかったことにある。

## よくある失敗

第8章で特に避けたい失敗は五つある。

一つ目は、アカウント管理を作成削除だけの話にすることだ。これでは権限残りや監査不全が起きやすい。

二つ目は、パスワードを厳しくすれば十分だと思うことだ。今は、固有性、長さ、MFA、復旧手段の方が重要である。

三つ目は、管理者権限を通常利用と混ぜることだ。事故面積が広がり、操作追跡もしにくくなる。

四つ目は、共有アカウントやサービスアカウントを例外扱いせず放置することだ。たいてい後で棚卸しから漏れる。

五つ目は、棚卸しを予定だけで終わらせ、結果反映と証跡を残さないことだ。見たが直していない、が一番危うい。

## 最低限ここまではやる

第8章の段階では、次の五つができれば十分に前進である。

一つ目。個人、管理者、共有、サービスのアカウント種別を分けて把握していること。

二つ目。管理者アカウントを通常利用アカウントと分けていること。

三つ目。管理者アカウントと重要 SaaS に MFA を設定し、復旧手段も持っていること。

四つ目。共有アカウントとサービスアカウントに所有者と用途があること。

五つ目。管理者権限は四半期、一般権限は半年を目安に棚卸しする予定があること。

これだけでも、アカウント管理はかなり安定する。ひとり情シスが全部を覚えている状態から、後で見返しても回る状態へ一歩進めるからだ。

## 確認したいこと

- ID、認証、権限、監査を分けて説明できる
- 管理者アカウントは通常用と分かれている
- 最上位権限の人数を把握している
- MFA と復旧手段をセットで整えている
- 共有アカウントとサービスアカウントに所有者がいる
- 定期棚卸しの予定と証跡がある

## 今日、今週、後でやること

今日やることは三つでよい。まず、管理者権限を持つアカウントを全部一覧にする。次に、その中で通常利用と管理者利用が混ざっているものへ印を付ける。最後に、共有アカウントとサービスアカウントの所有者欄を埋める。

今週やることは、MFA 未設定の管理者アカウントをなくし、予備の復旧手段を一つ用意することだ。同時に、次回の棚卸し日をカレンダーへ入れる。予定が入っていない棚卸しは、だいたい実施されない。

後で整えることは、SSO 対象の拡大、期限付き権限付与、監査ログ確認の定例化である。全部を一度にやる必要はないが、個人、管理者、共有、サービスの区別だけは曖昧にしない方がよい。

## 第9章

## 入社、異動、退職とアカウント運用フロー

---

退職者のメールは止めた。だから対応は終わったと思っていた。ところが午後になると、営業から「引き継ぐはずの取引先メールが見られない」と連絡が来る。総務からは「共有フォルダの中身も必要らしい」と言われる。さらに調べると、その退職者はスマートフォンからまだ一部のアプリに入れ、委託先向けの共有リンクも生きていた。止めたつもりだったのに、仕事もアクセスも中途半端に残っていた。

こうした詰まり方は、入社、異動、退職を単発作業で捉えた時に起きる。実際には、人が動くたびに、アクセス、データ、端末、記録の状態も変えなければならない。

第8章では、ID、認証、権限、監査を一体で捉える原則を整理した。第9章では、その原則を人のライフサイクルへ落とし込む。ここで扱うのは、人事総務の一般論ではない。ひとり情シスが漏らしやすい、入社、異動、退職、休職、委託終了時のIT運用を、崩れにくい順番で整理する章である。

### 入社、異動、退職は統制イベントである

人の状態が変わると、会社から見たアクセス状態も変わる。だから、入社、異動、退職は人事イベントであると同時に、統制イベントでもある。

入社では、まだ何にも入れない人を、必要な範囲だけ入れる状態へ変える。異動では、前の役割で持っていた権限を見直し、新しい役割へ合わせ直す。退職では、入っていたものを止め、残すべきデータを移し、不要なものを閉じる。

この三つを別々の作業として見ると、抜けが起きやすい。JML という言い方をすることもあるが、要点は **人の状態変更に合わせて、アクセスとデータの状態も変える** ことである。最初にこの表で整理しておくこと、考え方がぶれにくい。

イベント	変えるべきもの	最初に確認すること	よくある抜け
入社	アカウント、所属、ライセンス、標準権限、MFA、端末	入社日、所属、上長、勤務地、使う主要サービス	アカウントだけ作り、ライセンスとMFAが後回し
異動	旧権限の剥奪、新権限の付与、承認権限、メーリングリスト	実施日、旧所属、新所属、兼務有無、残す例外	新しい権限だけ足し、古い権限を残す
退職、委託終了	サインイン遮断、セッション失効、回復手段切り離し、端末、データ移管、最終状態	最終出社時刻、引き継ぎ先、保持要件、削除可否	メール停止だけで終え、セッションと共有が残る

第8章で見たように、アカウント管理とは、誰として扱い、どう本人確認し、どこまで入れ、後で追えるかを崩さず回すことだった。第9章では、その考え方を時間軸に沿って扱う。人に紐づく状態は固定ではない。変化に合わせて更新する必要がある。

## 入社時に最初から整えるべきもの

入社対応で最も避けたいのは、「とりあえず使える」状態のまま始めることだ。初日に使えないのも困るが、急いだ結果として危うい状態を作るのも困る。

Microsoft 365 も Google Workspace も、新規ユーザー作成時に、アカウント名、初期パスワード、ライセンス、所属、権限、連絡先をまとめて整える前提になっている。つまり、入社時に必要なのはメールアドレス発行だけではない。

最低限、次のものは入社時にそろえたい。

項目	最低限決めること	抜けると起きること
個人アカウント	氏名、ユーザー名、社内利用かゲストか	共有アカウントで開始し、後で追えなくなる
所属部署や組織情報	部署、上長、組織単位、勤務地	誤ったポリシーや配布先になる
標準ライセンス	その職種で初日から必要なサービス	サインインできても仕事が始まらない
標準権限	役割グループ、メーリングリスト、承認経路	個人直付けの例外が増える
MFA 設定と復旧手段	何で登録するか、予備手段をどう持つか	初日から弱い認証か、逆に締め出しが起きる
端末準備とのひも付け	どの端末を渡すか、管理下に置くか	私物端末から無管理で使い始める
本人への引き渡し方法	初期情報をどこへどう渡すか	その場しのぎの受け渡して情報が漏れる

ここで重要なのは、個別例外から始めないことだ。第8章で整理したように、権限は個人直付けより、役割とグループで持たせた方が崩れにくい。入社時は、その人に一つずつ権限を付けるより、「営業の標準」「経理の標準」「情シスの標準」といった型へ乗せる方がよい。

また、初回情報の渡し方も軽視しない方がよい。Google の新規ユーザー作成では、二次連絡先へ情報を送る前提が示されている。Microsoft 365 でも、資格情報を印刷や PDF で安全に共有する流れがある。少なくとも、誰へ、どの経路で、どこまで渡すかを決めたい。共有アカウントを一時的に渡して済ませるやり方は、最初から避けた方がよい。

入社時は、準備の順番も大事である。現実的には次の流れが回しやすい。

1. 人事または依頼元から確定情報を受ける
2. アカウントと所属情報を作る
3. ライセンスと標準権限を付ける
4. 端末準備とひも付ける
5. MFA を有効にする
6. 初期情報を安全に渡す
7. 初日確認をする

Google Workspace では、新規アカウントが各サービスへ反映されるまで最大24時間かかることがある。つまり、「当日朝に依頼が来てもすぐ全部使える」とは限らない。ひとり情シスの現場では、入社連絡の締切を持った方がよい。今日入る人の依頼が今日の朝に来る運用は、事故の元である。

## 異動時は旧権限の剥奪を先に考える

異動時に起きやすい失敗は、新しい権限を足すことに意識が向き、古い権限を外し忘れることである。

異動は退職ほど緊張感が出にくい。本人は社内に残るので、つい「使えなくなると困るから、とりあえず今のままにしておこう」になりやすい。だが、その積み重ねが権限肥大化を生む。

異動時に見直したいのは、少なくとも次のものだ。

- 所属グループ
- 共有フォルダ権限
- メーリングリスト

- 承認権限
- 管理者権限
- 外部サービスのロール

この時に大事なのは、**外すかどうか**の判断軸を決めておくことだ。

見直す対象	外す判断の基本	残すなら書くべきこと
所属グループ	新所属で標準利用しない	兼務理由、終了予定日
共有フォルダ権限	現在の業務で不要	引き継ぎ期間、閲覧だけで足りるか
メーリングリスト	受信が不要	代理対応の期間
承認権限	前任者としての権限だけだった	誰の代行か、いつまでか
管理者権限	運用責任が離れた	障害対応要員として残す期限
外部サービスのロール	案件、取引先、契約が変わった	契約上の理由、見直し日

考え方としては、まず旧所属の標準権限を外し、そのうえで新所属の標準権限を付ける。兼務の場合だけ、両方を残す理由を明示する。これが基本である。

ここで重要なのは、「この人は前から使っていたから」という理由だけで残さないことだ。前部署の共有フォルダ、旧案件の管理画面、前任者として持っていた承認権限。こうしたものは本人の利便性には見えても、会社から見ると不要アクセスかもしれない。

異動時にアカウントを作り直す必要は、普通はない。むしろ、同じ人である以上、履歴やデータを保ちつつ、所属と権限を変える方が自然である。削除と再作成は、履歴の分断や引き継ぎ漏れを生みやすい。特に Microsoft Entra や Google Workspace の標準運用は、同じアカウントを前提に profile や group、license を変える考え方である。

異動対応で現実的に回しやすいのは、依頼票へ次の欄を持つことだ。

- 旧所属
- 新所属
- 外す権限
- 残す権限
- 新たに付ける権限
- 実施日
- 確認者

この一行があるだけで、「足したが外していない」をかなり減らせる。

## 休職、長期不在、委託終了は削除ではなく一時遮断を基本にする

人の状態が変わったからといって、いつも削除が正しいわけではない。休職、産休育休、長期出張、調査中、委託先の一時停止。こうした場面では、一時遮断の方が適切なことが多い。

Google Workspace の suspend は、アクセスを止めるがデータは消さない構造になっている。しかも、共有ドキュメントの共同編集者アクセスは残る。Microsoft 側でも、block sign-in、パスワード変更、セッション失効を組み合わせ、まずアクセスだけ止められる。これは実務上かなり重要だ。

一時遮断が向くのは、次のようなケースである。

- 休職中
- 長期不在
- 不正利用の疑いがあり調査中
- 委託先の作業が一時停止
- 退職確定前だが先にアクセスを止めたい

状況ごとに、基本操作と追加確認を分けて考えると崩れにくい。

状況	基本操作	追加で見ること
休職、長期不在	suspend または sign-in block	端末保管、回復手段、代理受信の要否
調査中、不正疑い	パスワード変更、全セッション sign-out、suspend または block	OAuth token、アプリパスワード、ログ保全
委託先の一時停止	委託先アカウントやゲストの停止	VPN、共有フォルダ、チケット、チャット
退職確定前	先に sign-in を止める	最終出社時刻、引き継ぎ先、端末回収

削除してしまうと、戻す時に手間が増え、履歴も切れやすい。逆に停止なら、必要時に復元しやすい。

ただし、一時遮断でも放置してよいわけではない。端末回収、MFA トークンの扱い、共有リンクの確認、委託先の VPN やゲスト権限の停止などは別途必要になる。アクセス遮断と周辺整理は分けて考えたい。

委託先対応で特に危ういのは、「社員ではないから人事イベントに乗らない」ことだ。実際には、契約終了は退職に準じるイベントである。委託先の個人アカウント、ゲストアカウント、共有フォルダ、チケットシステム、VPN、チャット参加権限は、終了日に合わせて止めなければならない。

## 退職時の初動は、遮断、失効、切り離しである

退職時に最初にやるべきことは削除ではない。遮断である。

Microsoft の元従業員対応ガイドでも、最初の工程はサインイン防止であり、その後にデータ保存や端末ワイプへ進む。Google でも、退職後のデータ保護として、端末ワイプ、復旧用連絡先削除、パスワード変更が示されている。つまり、初動でやるべきことは一貫している。

退職時の初動は、次の順番で考えるとよい。

順番	やること	ここで押さえる知識
1	新規サインインしにくい状態を作る	Microsoft 365 は <code>block sign-in</code> だけだと反映に最大 24 時間かかりうるので、まずパスワード変更を併用する
2	既存セッションやトークンを失効させる	<code>sign out of all sessions</code> や <code>Revoke-EntraUserAllRefreshToken</code> のように、既発行トークンを別で切る必要がある
3	回復手段を切り離す	復旧用メール、電話番号、security key、app password、OAuth token を残さない

順番	やること	ここで押さえる知識
4	端末や BYOD 上の業務データを消す	モバイル端末の遠隔ワイプは、アカウント停止とは別工程である
5	その後に引き継ぎとライセンス処理へ進む	メールやファイルの移管前に削除しない

ここで「セッションやトークンを失効させる」を独立させるのが大事だ。パスワードを変えても、すでに発行済みのセッションやトークンが生きていれば、すぐには切れないことがある。Microsoft 365 の **sign out of all sessions** は 1 時間以内が目安で、Outlook on the web は即時に切れないことがある。Microsoft Entra でも refresh token とブラウザセッション失効の機能が独立しているのは、そのためである。

また、復旧用メールアドレスや電話番号が本人の私物にひも付いたままだと、後から回復経路として使える可能性がある。Google Workspace でも、退職時に recovery 情報、OAuth token、sign-in cookies、security key、app password の剥奪まで求めている。退職時は、認証だけでなく回復経路も切る必要がある。

即日退職や不正疑いの場面では、この初動がさらに重要になる。この時は、データ引き継ぎやライセンス処理より先に、まず入れない状態を作る。後続作業はそのあとでよい。

## メールとファイルの引き継ぎを分けて考える

退職対応でよくある誤解は、「データ引き継ぎ」と一言でまとめてしまうことだ。実際には、メールとファイルでは考え方が違う。

実務では、次のように分けて考えると漏れにくい。

対象	主な選択肢	先に決めること	間違えやすい点
メール	一定期間の転送、共有メールボックス化、PST 書き出し、閲覧権付与	誰が引き継ぐか、いつまで必要か、削除時期	転送設定だけで元アカウントを消す
ファイル	OneDrive への管理者アクセス、別担当者への権限付与、所有権移管、共有ライブラリや共有ドライブへの移動	新所有者、保存先、保持期限	閲覧権と所有権を同じだと思う

この違いを意識しないと、「見られるようにしたつもりだが所有権は元のまま」「転送はしたが過去メールは見られない」といったズレが起きる。

Google Drive の所有権移管では、共有権限そのものは変わらず、有効な社内アカウントへの移管が前提とされている。つまり、誰が所有者かと、誰が見られるかは別論点である。Microsoft 365 でも、OneDrive へのアクセス付与と Outlook データの引き継ぎは別工程として整理されている。第9章では、この違いを曖昧にしない方がよい。

また、共有メールボックスやメール転送を使う場合、Microsoft 365 では元アカウントがアンカーとして必要になる。つまり、転送や共有化を設定するのに元アカウントをすぐ削除すると成り立たないことがある。順番を間違えないことが重要である。

実務では、退職対応票に次の欄を持つと漏れにくい。

- メール転送の要否
- 共有メールボックス化の要否
- 過去メール保存の要否

- OneDrive や Drive の引き継ぎ先
- 所有権移管の要否
- 実施期限

## 停止、ライセンス削除、アーカイブ、削除を使い分ける

第9章で特に整理したいのが、状態の違いである。ここが混ざると、不要な削除や無駄な課金起きる。

まず、停止である。これはアクセスを止めるが、データは保持する。退職、調査中、退職直後の初動に向いている。

次に、ライセンス削除である。これは課金対象サービスの利用権を外す操作であり、必ずしもアカウント削除ではない。Microsoft 365 のガイドでも、ライセンス削除とユーザー削除は別工程になっている。

次に、アーカイブである。Google Workspace では archived user にすることで、アクセスを止めつつデータを保持し、アクティブライセンスを再利用できる。ただし、対応エディションと Archived User subscription が前提である。

最後に、削除である。これは最終工程である。2026年3月21日時点の代表例では、Microsoft では通常約 30 日、Google Workspace では 20 日の復元可能期間があるが、その期間を過ぎると戻せない。しかも、何が保持され、何が残らないかはサービスごとに違う。

状態を混ぜないために、次のように整理したい。

やりたいこと	向く操作	補足
今すぐ使わせない	停止、suspend、block sign-in、セッション失効	退職直後、調査中、長期不在の初動

やりたいこと	向く操作	補足
戻る可能性がある	停止	休職、長期不在、契約一時停止
課金対象サービスだけ外したい	ライセンス削除	データ保持や機能停止の条件は製品ごとに違う
データを保持しつつ active ライセンスを空けたい	アーカイブ	Google Workspace では条件付きで使える
完全終了したい	削除	引き継ぎ、保存、保持確認の後で行う

削除を急ぐと、後で困る。特に、復元可能期間とデータ保持条件を把握せずに削除すると、「必要なファイルがあった」「法務確認でメールが要る」となった時に詰まる。

また、ハイブリッド環境では注意が増える。Microsoft 365 側のアカウントがローカル Active Directory から同期されているなら、削除と復元はローカル Active Directory 側で行う必要がある。見えている管理画面だけで完結するとは限らない。

## 外部委託先、ゲスト、短期利用者も同じ型で扱う

社員の入退社は意識されやすいが、外部委託先やゲストは見落とされやすい。だが、権限が残れば同じように危うい。

特に確認したいのは、次の対象だ。

- ゲストユーザー
- 委託先個人アカウント
- 一時的な共有フォルダ権限
- プロジェクトごとのチャンネル参加

- VPN やリモート接続
- チケットシステムや監視画面の閲覧権

開始時と終了時に見たい項目をそろえておくと、社員以外も回しやすい。

対象	開始時に記録すること	終了時に止めるもの
ゲストユーザー	招待元、目的、終了日	共有フォルダ、チャンネル、招待状態
委託先個人アカウント	契約責任者、担当業務、使うシステム	VPN、チケット、監視、共有リンク
短期利用者	利用開始日、終了日、上長	ライセンス、端末、MFA、共有権限
一時的な共有リンク	作成者、対象データ、期限	共有設定、公開リンク

委託先の終了日は、人事システムに乗らないことが多い。だからこそ、契約終了日とアカウント停止日を結び付ける必要がある。第6章で見たように、外部委託は責任分担であり、終了時の回収まで含めて設計しなければならない。

短期利用者も同じである。短いから大丈夫ではなく、短いからこそ終了確認を忘れやすい。終了日を持ち、終わったら止める。この単純な運用が重要である。

## 小さな会社で現実的に回るやり方

JML を厳密なワークフローシステムで回せる会社ばかりではない。小さな会社では、まず1枚のチェックリストでも十分効果がある。

最低限、次の項目は一枚にまとめたい。

列	何を書くか
イベント種別	入社、異動、休職、退職、委託終了

列	何を書くか
対象者	氏名、所属、上長、依頼元
実施日、実施時刻	いつ切り替えるか。退職は時刻まで持つ
アカウント	作成、停止、削除、guestかどうか
ライセンス	付与、削除、再利用先
権限	外す権限、残す権限、付ける権限
MFAと回復手段	登録確認、削除確認、予備手段
セッション失効	実施有無、実施時刻
端末	受け渡し、回収、ワイプ
メール引き継ぎ	転送、共有メールボックス化、保存
ファイル引き継ぎ	OneDrive、Drive、所有権移管、保存先
最終状態	停止、ライセンス削除、アーカイブ、削除
実施記録	実施者、確認者、備考

運用としては、次のように分けると回しやすい。

タイミング	主にやること
事前	入社準備、通常退職準備、異動予定確認、引き継ぎ先決定
当日	サインイン遮断、権限変更、セッション失効、端末回収
後続	データ移管、ライセンス再利用、アーカイブまたは削除、記録保存

特に退職時は、当日やることと後でやることを分けた方がよい。当日は止めることに集中し、引き継ぎや保存はその後に順序立てて進める方が安全である。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員の入社日が一週間前に共有される。ひとり情シスは、アカウント、ライセンス、部署グループ、端末、MFA を事前に整え、初日朝に安全な方法で初期情報を渡す。異動者については、旧部署の権限一覧を確認し、残すものと外すものを先に決める。退職者については、最終入社時刻に合わせてパスワード変更、サインイン遮断、セッション失効、回復手段の切り離しを実施し、その後にメール転送、共有メールボックス化、OneDrive や Drive の引き継ぎを進める。削除やアーカイブは、移管確認後に判断する。結果として、人の出入りが起きても慌てない。

崩れた時の例では、入社連絡は当日朝に来る。アカウントだけ急いで作り、ライセンスも MFA も後回しになる。異動では、旧部署権限が残ったまま新部署権限を足す。退職時には、メール停止だけ先にやり、後で必要になったファイルが見られず、復元期間内に慌てて戻すことになる。委託先アカウントは契約終了後も共有フォルダへ入り続ける。問題は忙しさだけではない。人の状態変更を、統制イベントとして設計していなかったことにある。

## よくある失敗

第9章で特に避けたい失敗は五つある。

一つ目は、入社時にアカウントだけ作って終わることだ。ライセンス、MFA、端末、標準権限が抜ける。

二つ目は、異動時に足すことばかり考え、外すことを後回しにすることだ。権限肥大化の典型である。

三つ目は、退職時に削除を急ぐことだ。まず遮断し、その後に引き継ぎと保存を行うべきである。

四つ目は、メールとファイルの引き継ぎを同じだと思ふことだ。閲覧権、所有権、保持期間は別々に見る必要がある。

五つ目は、外部委託先やゲストを本流のチェックリストから外すことだ。見落としやすいが、危険度は低くない。

## 最低限ここまではやる

第9章の段階では、次の五つができれば十分に前進である。

一つ目。入社、異動、退職、委託終了を一枚のチェックリストで管理していること。

二つ目。異動時に旧権限を外す欄があること。

三つ目。退職時の初動が、削除ではなく、遮断、失効、回復手段切り離しになっていること。

四つ目。メール引き継ぎとファイル引き継ぎを分けて考えていること。

五つ目。停止、アーカイブ、削除の違いと、復元可能期間を把握していること。

これだけでも、人の出入りに伴う事故はかなり減る。ひとり情シスの負荷を減らすとは、全部を自動化することではない。抜けやすい順番を先に整えることである。

### 確認したいこと

- 入社、異動、退職の標準手順がある

- 異動時に旧権限を外す欄がある
- 退職時の初動が削除ではなく遮断になっている
- メールとファイルの引き継ぎ方法を分けている
- 停止、アーカイブ、削除の使い分けを説明できる
- 復元可能期間と証跡保存を把握している

## 今日、今週、後でやること

今日やることは三つでよい。まず、入社、異動、退職、委託終了を同じ様式で扱えるチェックリストを作る。次に、その様式へ「旧権限を外す欄」と「セッション失効欄」を足す。最後に、主要サービスごとの復元可能期間を一覧にする。

今週やることは、直近の異動者か退職者を一件選び、実際にそのチェックリストへ当てはめてみることだ。そこで詰まった欄が、自社の抜けである。

後で整えることは、人事連絡の締切、端末回収手順との接続、外部委託先終了の自動通知である。全部を最初からシステム化する必要はないが、人が動くたびにゼロから考える運用はやめた方がよい。

## 第10章

## PC、スマホ、端末、資産管理

---

退職者のアカウントは止めた。だから対応は終わったと思っていた。ところが数日後、棚に置かれた返却 PC を次の入社者へ回そうとして手が止まる。前の利用者のローカルファイルは消えたのか。暗号化は有効だったのか。管理画面ではまだその人の端末として見えている。貸与スマートフォンも一台行方が曖昧で、誰が持っているかすぐには言えない。

こうした詰まり方は、端末管理を「買って配る作業」と見た時に起きる。実際には、PC もスマートフォンも、配布して終わりではない。標準化して、管理下へ載せて、利用中の状態を追い、紛失時に止め、返ってきたら消して、再び配るか廃棄するかを決める必要がある。

第9章では、人のライフサイクルに合わせてアカウントと権限を変える運用を見た。第10章では、その裏側にある端末のライフサイクルを扱う。ここで扱うのは、キッキングの小技ではない。ひとり情シスが、少ない人数でも端末を崩れにくく回すための基本設計である。

### 端末管理はライフサイクル管理である

端末管理というと、購入、初期設定、貸与の三つだけを思い浮かべやすい。だが実際には、端末は次の流れで動く。

1. 調達する
2. 管理対象として登録する
3. 標準構成を適用する

4. 利用者へ配布する
5. 利用中の状態を監視し、更新する
6. 紛失、盗難、故障に対応する
7. 回収する
8. 再配備するか、退役するかを決める

このどこかが曖昧だと、端末はすぐに管理の外へ落ちる。たとえば、購入時に台帳へ載っていない端末は、誰が持っているか分からなくなる。配布時に管理下へ入っていない端末は、更新や遠隔操作ができない。返却後の消去手順がない端末は、前任者のデータを抱えたまま次の人へ渡りかねない。

CIS Controls では、組織が保有、接続、管理する資産を特定し、承認されていない資産を把握することを基礎統制として置いている。つまり、第10章で最初に持つべき感覚は、「今ある端末が全部言えるか」である。

最初に、各段階で何を定める章なのかを表で見ておくと整理しやすい。

段階	ここで決めること	抜けると起きること
調達	所有区分、標準機種、購入先	例外機種だらけで保守が重くなる
登録	資産番号、シリアル、管理方式、所有者	誰の端末か分からなくなる
標準構成適用	OS、暗号化、更新、必須アプリ、権限	同じ機種でも中身がそろわない
配布	利用者、貸与日、付属品、返却予定	故障時や退職時に追えない
利用中管理	更新、接続状況、例外権限、状態	未更新端末、休眠端末が埋もれる
事故対応	lock、wipe、回線停止、代替機	紛失時に止めるべきものが遅れる

段階	ここで決めること	抜けると起きること
回収	本体、充電器、SIM、貸与記録	返却漏れや付属品欠品が残る
再配備、退役	消去、再設定、管理解除、廃棄	消していない端末を再利用してしまう

## 先に標準構成を決める

端末管理で最も崩れやすいのは、毎回その場で設定を決める運用である。急ぎの入社者が出るたびに、昨日の設定を思い出しながらアプリを入れ、権限を付け、ブラウザを調整する。これでは、同じ機種でも中身がそろわない。

先に決めるべきなのは、機種より標準構成である。最低限、次の項目は持っておきたい。

- OS とバージョン方針
- 必須アプリ
- 業務ブラウザと拡張
- 画面ロック
- ディスク暗号化
- ウイルス対策や EDR の有無
- ローカル管理者権限の扱い
- 更新の適用方針

標準構成票は、少なくともこの粒度で書けると実務へ落ちやすい。

項目	最低限決めること	抜けると起きること
OS と edition	何を標準にするか、いつまで使うか	機種ごとに更新条件がばらける

項目	最低限決めること	抜けると起きること
更新方針	自動適用の範囲、延期条件、例外手順	更新停止端末が増える
画面ロックと認証	PIN、パスワード、ロック時間	置き忘れ時の不正利用が起きやすい
ディスク暗号化	有効化方法、回復情報の保管先	紛失時にローカルデータ保護が弱い
必須アプリ	業務アプリ、ブラウザ、管理エージェント	人ごとに入っているものが違う
セキュリティ対策	EDR、ウイルス対策、証明書	最低限の保護がそろわない
ローカル管理者権限	誰に持たせるか、例外の期限	勝手な変更が蓄積する
ブラウザとネットワーク	拡張、VPN、Wi-Fi、証明書	利用開始後に個別調整が増える

ここで重要なのは、完璧な標準を目指しすぎないことだ。ひとり情シスの現場では、最初から部署別に細かく分けすぎると維持できない。まずは、全社共通の標準PC、必要なら少数の例外構成、というくらいに絞った方がよい。

ローカル管理者権限も、端末管理を崩しやすい論点である。何でもできる状態にしておくと、利用者が困った時に自分で入れられて楽に見える。だが、その結果、勝手に入ったアプリ、止まった更新、原因不明の設定変更が積み上がる。第10章では、「管理者権限を配るかどうか」ではなく、「標準から外れる変更をどう管理するか」で考えたい。

Microsoft の Windows Autopilot は、モデルごとの custom image を維持しなくてよい方向を示している。Apple の Automated Device Enrollment も、初回起動時から設定適用や FileVault 強制を前提にしている。製品は違って共通するのは、標準構成を **人が思い出して入れる設定** ではなく **管理基盤から再現できる設定** として持つ方が崩れにくいということである。

## 台帳は台数確認ではなく状態把握に使う

会計上の資産台帳があっても、運用として十分とは限らない。資産番号と購入日だけ分かっても、今の運用判断には足りないからである。

運用台帳として最低限ほしいのは、次の項目である。

- 資産番号
- 端末種別
- メーカーと型番
- シリアル番号
- 利用者
- 所有区分
- 管理方式
- 最終接続日
- 現在状態
- 保管場所または返却予定

ここでいう所有区分は、会社所有か、レンタルか、私物利用かである。管理方式は、MDM 管理下か、個別設定か、未管理かである。現在状態は、利用中、予備機、故障中、返却待ち、退役予定などである。

この項目を持っていると、単に何台あるかだけでなく、次の判断ができる。

- 誰の端末が長く接続していないか
- 返却予定なのにまだ回収できていない端末はどれか
- 会社所有なのに管理下へ入っていない端末はないか
- 故障交換時に代替機を出せるか

会計台帳と運用台帳は、役割が違う。会計台帳だけでは足りない理由を分けておくと分かりやすい。

台帳	主に見ること	第10章で足したい列
会計や固定資産の台帳	購入日、金額、減価償却、資産番号	ここでは十分でも、運用には足りない
運用台帳	誰が使うか、どう管理するか、今どの状態か	利用者、管理方式、最終接続日、状態、返却予定

Google Workspace にも、非アクティブな会社所有端末を把握する機能がある。Microsoft Intune にも、長期間接続していない端末記録を cleanup 対象として整理する考え方がある。つまり、公式資料でも「使われていない端末や古い記録を見直す」ことは標準運用として扱われている。

## 配布と初期設定を個人技にしない

標準構成を決めても、それを端末へ安定して入れられなければ意味がない。ここで価値を持つのが、自動登録や MDM である。

Microsoft の Windows Autopilot は、初回配布から再利用や廃棄までを含むデバイスライフサイクルの一部として説明されている。Apple の Automated Device Enrollment も、組織所有端末を初回アクティベーションから MDM 管理下へ確実

に乗せる仕組みである。重要なのは製品名ではなく考え方だ。つまり、端末は「受け取った人が自由に設定するもの」ではなく、「会社の標準設定が自動で入るもの」として扱う方が崩れにくい。

配布方式の違いは、次のように整理すると判断しやすい。

方式	向く場面	できること	注意点
自動登録とMDM連動	組織所有端末の標準配布	初回起動から登録、設定、アプリ配布	事前準備が要る
Apple Automated Device Enrollment	組織所有のApple端末	unenrollment防止、Setup Assistant中の必須設定、FileVault強制	組織所有前提で考える
手動登録	例外端末、試験導入、少量運用	最低限の管理を載せられる	登録漏れや設定漏れが出やすい
BYOD向け登録	私物端末で業務利用する時	会社データ側だけを管理しやすい	端末全体を自由に扱えるとは限らない

配布時に持ちたい工程は、次の通りである。

1. 端末を運用台帳へ登録する
2. 管理対象として登録する
3. 標準構成を適用する
4. 利用者をひも付ける
5. 貸与記録を残す
6. 初回サインインと利用開始を確認する

この順番の中で、貸与記録を軽く見ない方がよい。誰にいつ渡したか、いつ返却予定か、付属品は何か。この記録がないと、故障時や退職時に「その端末は誰のものだったか」が曖昧になる。

また、予備機をどう持つかも決めておきたい。全台ぎりぎり回していると、故障時に復旧が遅れる。小さな会社でも、標準構成済みの予備機を少数持つだけで、障害対応はかなり安定する。

## 利用中管理では更新、権限、休眠端末を見る

配布後の端末管理は、気合いでは続かない。見る項目を絞る必要がある。ひとり情シスが最低限追いたいのは、更新、権限、状態の三つである。

まず、更新である。OS と主要アプリの更新が止まると、脆弱性対応も不具合修正も止まる。標準構成を作る時点で、いつ、どこまで自動適用するかを決めた方がよい。

次に、権限である。端末ローカルの管理者権限が広く残っていると、標準から外れた変更が増え、問題切り分けが難しくなる。必要な例外は記録し、ずっと例外のまま放置しないことが大切である。

最後に、状態である。特に見たいのは、しばらく接続していない端末である。使っていないだけに見えても、実際には次のような可能性がある。

- 返却されたが台帳更新が漏れている
- 退職者がまだ持っている
- 出張用として貸し出されたまま戻っていない
- 故障しているが放置されている
- 管理から外れている

休眠端末は、単なる無駄ではなく、管理抜けの兆候である。月次で一度確認し、利用者不明、長期未接続、退役予定のものを洗い直すだけでも効果がある。

月次で見る項目を絞るなら、次の三つが現実的である。

見る項目	何を見るか	ここで誤解しやすいこと
更新	OSと主要アプリが止まっていないか	更新未適用は気合いで解決しない
例外権限	ローカル管理者や例外設定が残っていないか	一度渡した例外が恒久化しやすい
休眠端末	長期間 check-in がない端末、最終接続日、最終利用者	cleanup で一覧から隠しても、端末そのものは消えない

特に Microsoft Intune の cleanup rule は、古い記録を admin center から隠すだけで、端末には wipe も retire も送らない。Google Workspace の inactive company devices report も、30 日 work data 同期がない company-owned Android を洗い出す仕組みであり、見つけた後の対応は別に必要になる。見つけることと止めることは分けて考えたい。

## スマホと BYOD は管理境界を先に決める

PC と比べると、スマートフォンは私物利用と業務利用が混ざりやすい。ここで曖昧にすると、事故時にもめる。

会社所有スマホであれば、比較的考えやすい。会社管理下へ登録し、必要な設定を配り、紛失時に lock や wipe を実行できる状態を目指せばよい。

難しいのは BYOD である。NIST の BYOD 関連資料でも、組織データ保護と利用者プライバシー保護の両立が課題として明示されている。つまり、BYOD を採るなら、最初に次の境界を決める必要がある。

- 会社データだけ消せるのか
- 端末全体を消せるのか
- どこまで設定を強制するのか

- どこまで端末状態を見えるのか
- 利用者へどう説明するのか

Google Workspace でも、管理レベルによって、会社データだけの wipe と端末全体の wipe が分かれている。つまり、BYOD では「何かあったら全部消す」が当然にはできない。ここを曖昧にしたまま運用すると、事故時に実行できると思っていた操作ができずに詰まる。

ひとり情シスとしては、BYOD を標準にしない方が運用は安定する。もし採るなら、対象業務、対象アプリ、消去範囲、利用者同意、退職時の処理を先に決めてから始めるべきである。

会社所有と BYOD の違いは、この表で見ると実務へ落ちやすい。

方式	主な管理対象	事故時にできること	先に決めること
会社所有スマホ	端末全体	lock、device wipe、在庫管理、回線停止	誰に貸与するか、返却時にどう消すか
BYOD の Android work profile	業務用 profile	会社データだけを外す、個人データは残す	何を work profile に入れるか
BYOD の iOS や Apple の user enrollment 系	業務用アカウントと managed app	managed app と設定の削除、個人側と分離	何が見えるか、何を消せるか

NIST SP 1800-22 も、BYOD はセキュリティ課題だけでなく、従業員プライバシーの課題を持つと整理している。したがって、**管理できるから導入する**ではなく、**どこまで管理するかを説明できる時だけ導入する**と考えた方が安全である。

## 紛失、盗難、故障時の初動を決めておく

端末事故で最も危ういのは、その場で考える運用である。紛失の連絡を受けてから、誰が何を止めるのかを考え始めると遅い。

最低限、初動は次のように固定しておきたい。

1. 連絡を受ける
2. 端末種別と所有区分を確認する
3. 対象アカウントのリスクを判断する
4. 必要に応じてサインインを止める
5. 端末側の lock または wipe を実施する
6. 回線停止や SIM 停止が必要ななら行う
7. 記録を残す
8. 代替機を手配する

ここで大事なのは、アカウント対応と端末対応を分けることである。第9章で見たように、アカウント停止やセッション失効は重要である。だが、それだけでは端末内に残るローカルデータやオフラインファイルまでは扱えない。Microsoft Intune や Google Workspace の資料が遠隔 wipe を独立機能として持っているのは、そのためである。

故障時も同じである。壊れたから交換する、で終わりではない。返ってきた端末のデータ消去、管理状態の更新、代替機の貸与記録が必要になる。故障をきっかけに台帳が崩れることは多い。

事故の種類ごとに、最初の一手をそろえておくと迷いにくい。

事故	まずやること	端末側でやること	追加確認
紛失、盗難	利用者、所有区分、位置情報の有無を確認	lock、必要なら wipe、SIM 停止	アカウント失効、最終同期、代替機
BYOD 紛失	会社アカウント側のリスク確認	account wipe や work profile 削除	個人データには触れない範囲を確認
故障	端末利用継続可否を判断	交換前に回収、必要なら消去	代替機、貸与記録、修理送り
返却遅延	返却予定と連絡先を確認	必要なら sign-out や 管理側ブロック	最終勤務日、付属品、引き継ぎ先

## 回収、再配備、退役を分けて考える

返却された端末は、ただの中古端末ではない。状態を判定して、次の行き先を決める必要がある。

まず回収である。ここでは、端末本体だけでなく、充電器、SIM、周辺機器、貸与記録を確認する。

次に状態確認である。故障の有無、管理下に残っているか、暗号化状態、返却理由を見て、再利用可能かを判断する。

次にデータ消去である。ここを飛ばしてはいけない。Apple の公式資料でも、消去は再設定や再配備前の独立工程として扱われている。Microsoft Intune の wipe も、退役や再利用時に使うことが前提になっている。つまり、「初期化したつもり」ではなく、「標準手順に沿って消去した」と言える状態が必要である。

そのうえで、再配備するなら標準構成を再適用し、新しい利用者へひも付ける。退役するなら、廃棄、返却、売却、保管のどれかを決め、台帳状態を変える。

ここで特に注意したいのが、管理解除である。Apple Business Manager の device release は、組織所有でなくなった端末だけに行うべきで、しかも不可逆である。つまり、初期化と管理解除は同じではない。まだ会社所有で再配備する可能性がある端末を、勢いで管理から外してはいけない。

操作名が似ていても、意味はかなり違う。特に次は混ぜない方がよい。

操作	何をするか	向く場面	注意点
cleanup	古い端末記録を一覧から隠す	stale record 整理	端末には何も起きない
retire	会社データと管理設定を外す	BYOD、組織管理を外す時	個人データは残る
wipe	工場出荷状態へ近い初期化	紛失、盗難、再配備、退役前消去	個人データも消える
delete	管理一覧から削除する	管理記録整理	Intune では Windows、iOS、macOS で retire を呼ぶだけで wipe ではない。Android は enrollment type により retire か wipe が変わる
release from organization	Apple Business Manager などの組織管理から外す	売却、譲渡、完全退役	不可逆で、erase と restore が別途必要

回収後の工程は、最低でも次の四つに分けたい。

- 回収した

- 消去した
- 再配備待ちにした
- 退役または廃棄した

この状態が分かれば、「返ってきたけれど次に渡してよいのか」が曖昧になりにくい。

## 小さな会社で現実的に回るやり方

すべての端末を高度な管理基盤で統一できる会社ばかりではない。小さな会社では、まず運用を細くしすぎないことが重要である。

現実的には、次のくらいから始めると回しやすい。

- 標準機種を PC とスマホで少数に絞る
- 標準構成票を 1 枚作る
- 端末台帳へ利用者、最終接続日、状態欄を入れる
- 配布時と返却時に必ず記録を残す
- 月に一度、休眠端末を確認する
- 紛失時の初動手順を 1 枚にする
- 退職時チェックリストへ端末回収欄を入れる

状態欄は、最初から細かくしすぎない方がよい。最小なら次のくらいで回る。

状態	意味	次にやること
利用中	利用者が使っている	更新、接続、例外権限を見る
予備機	すぐ貸せる状態	定期起動、更新確認
回収待ち	返却されるはずだが未回収	利用者との日程確認

状態	意味	次にやること
消去待ち	回収済みだがまだ次に渡せない	wipe や初期化を実施
再配備待ち	消去済みで次の配布待ち	新利用者の割当て
退役	もう使わない	廃棄、返却、売却、必要なら管理解除

第9章の JML フローとつなげると、端末管理はかなり安定する。入社時は、誰にどの標準端末を渡すかを決める。異動時は、端末を変えるか、設定だけ変えるかを判断する。退職時は、返却、消去、再配備待ちまでを一続きで処理する。人と端末を別々に動かさないことが大切である。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員の入社連絡が一週間前に来る。ひとり情シスは、標準 PC を一台確保し、運用台帳へ登録し、会社管理下へ入れ、標準構成を適用しておく。初日に利用者へ貸与し、貸与記録を残す。退職時は、最終入社日に端末を回収し、状態確認後に消去し、再配備待ちへ移す。スマートフォン紛失時は、あらかじめ決めた手順に沿って、アカウント確認、端末 lock、必要に応じた wipe、回線停止、代替機手配まで進める。結果として、誰が何を持っているか、返ってきた端末が次に使えるかをすぐ判断できる。

崩れた時の例では、端末は調達のたびに機種が違い、設定内容も担当者の記憶頼みである。貸与記録が曖昧なので、退職者の返却 PC がどれかすぐに分からない。スマートフォンを紛失しても、会社所有か私物利用かが即答できず、どこまで消せるか判断できない。返ってきた端末は、とりあえず初期化したつもりで次の人へ回される。結果として、配布、回収、再配備のたびに毎回調べ直しになり、事故が起きた時には止めるべきものが見つからない。

## 最低限ここまではやる

第10章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 標準端末構成を決める
2. 利用者と状態が見える台帳を持つ
3. 紛失時の初動を固定する
4. 回収後の消去と再配備の手順を分ける

この四つがあるだけで、端末管理はかなり崩れにくくなる。端末管理とは、物を配ることではない。会社管理下へ置き続け、必要時に止め、戻し、消し、また配れる状態を保つことである。

## 今日、今週、後でやること

今日やることは、端末台帳を開き、利用者欄と状態欄が埋まっているか確認することである。空欄が多いなら、そこが最初の詰まりどころである。

今週やることは、標準 PC と標準スマホの構成票を 1 枚ずつ作ることである。完璧でなくてよい。OS、必須アプリ、暗号化、画面ロック、管理方法が書かれていれば十分に価値がある。

後でやることは、紛失時フローと返却後フローを、JML の流れとつなげて見直すことである。人が動いた時に端末も動く。この一本線が引けると、第10章の内容は実務へ落ちる。

## 第11章

# ネットワーク、拠点、テレワーク

---

「ネットが遅いです」と言われても、それだけでは何も分からない。Wi-Fi が不安定なのか、回線自体が落ちているのか、特定の会議室だけなのか、在宅勤務者だけなのか。ところが実際の現場では、社内 Wi-Fi とゲスト Wi-Fi が同じ設定のまま、VPN の入口も複数あり、どの回線が主回線でどこへ連絡すればよいかも一枚で説明できないことがある。これでは、障害が起きた時に最初の一手が決まらない。

ネットワークは、専門家だけが触る謎の箱ではない。もちろん細かな設定は専門性が高い。だが、ひとり情シスが押さえるべき本質は別にある。回線がどこから入り、どこで分かれ、社内用と外部向けがどう分かれ、在宅勤務者はどこから入るのか。この構造を説明できることが、運用責任の出発点である。

第10章では、端末を会社管理下へ置き続ける運用を見た。第11章では、その端末をつなぐ基盤を扱う。ここで扱うのは、ルーターの CLI 操作ではない。オフィス、拠点、Wi-Fi、テレワーク接続を、少人数でも崩れにくく回すための考え方である。

## ネットワーク管理の最初の仕事は、図で説明できるようにすること

ネットワークが難しく感じる最大の理由は、見えないまま動いているからである。だから最初にやるべきことは、高度な最適化ではなく、構成を見えるようにすることだ。

CIS Control 12 では、ネットワーク構成図や関連文書を維持し、少なくとも年 1 回、または大きな変更時に見直すべきとしている。つまり、構成図は立派な設計資料ではなく、日常運用の必須資料である。

最初の構成図は細かくなくてよい。最低限、次が分かれば十分に役立つ。

- 回線事業者と回線の入口
- ルーターやファイアウォール
- 主なスイッチ
- Wi-Fi アクセスポイント
- 社内用ネットワーク
- ゲスト用ネットワーク
- VPN や在宅勤務の入口
- 重要な接続先

図に最低限入れたい項目を、役割と一緒に見ると分かりやすい。

項目	なぜ必要か	障害時に何が分かるか
回線事業者と回線入口	外部依存先を明確にする	ISP 側か社内側かを分けやすい
ルーターやファイアウォール	境界を把握する	全社停止時の確認先が決まる
主なスイッチ	どこで分岐しているかを見る	特定フロアや会議室だけの障害を追える
SSID と接続先	社内用、ゲスト用、機器用を分ける	Wi-Fi 障害の影響範囲を狭められる
VPN や在宅勤務の入口	外部からの正規経路を決める	在宅だけ止まる時の確認先が分かる
重要な接続先	DNS、認証、拠点間接続、基幹系を明示する	インターネット障害か社内資源障害か分かる

項目	なぜ必要か	障害時に何が分かるか
外部連絡先	連絡遅延を防ぐ	契約番号や窓口探しから始めなくて済む

この図があると、障害や変更のたびに効く。たとえば、「在宅勤務者だけつながらない」なら VPN 側を見る。「ゲストだけつながらない」なら SSID か DHCP 側を見る。「全社でインターネットが出られない」なら回線や境界機器を見る。図がないと、この切り分けがいつも頭の中だけで行われ、属人化する。

構成図と一緒に持ちたいのが、連絡先の一覧である。回線事業者、保守業者、機器ベンダー、社内窓口。この一覧がないと、障害時にまず電話番号を探すことになる。ネットワーク障害時は、調査力より先に、連絡の速さが効くことが多い。

加えて、CIS Control 12 は、構成図を年 1 回または大きな変更時に見直す考え方を示している。機器のソフトウェアバージョン確認も月次以上で回す前提である。図、連絡先、機器一覧は別々の紙にせず、同じ運用束として更新した方が崩れにくい。

## 社内、ゲスト、機器用を分ける

ネットワーク設計で最初に持ちたい原則は、用途の違うものを混ぜないことである。

NIST の WLAN ガイドでは、内部利用と外部利用でセキュリティプロファイルが違えば、論理的に分離した WLAN を用意すべきだとしている。ゲスト用 WLAN からは、原則として内部ネットワークを通らず、必要最小限だけに絞る考え方である。これは大企業向けの厳格論ではない。小さな会社ほど効果が大きい。

少なくとも、次の区分は分けて考えたい。

- 社員が業務で使うネットワーク
- 来客や一時利用者向けのゲスト Wi-Fi
- 会議室機器やサイネージなどの共用機器
- プリンタ、複合機、IoT 機器

最小なら、次のような役割分離で十分に効く。

区分	主に置くもの	原則	混ぜると起きやすいこと
社内用	社員 PC、管理端末	業務資源へ必要範囲だけ通す	ゲストや古い機器の影響を受ける
ゲスト用	来客端末、一時利用端末	原則インターネットだけ	来客用設定ミスが社内へ波及する
共用機器用	会議室端末、サイネージ	必要な SaaS や配信先だけ通す	共用機器トラブルで社内切り分けが複雑になる
プリンタ、IoT 用	複合機、監視カメラ、IoT	必要な宛先だけ通す	古い機器や既定設定の影響範囲が広がる

全部を一つにすると、問題が起きた時の影響範囲が広がる。来客用 Wi-Fi の設定ミスが社内端末へ波及し、古い会議室機器が同じセグメントにいるせいで切り分けが複雑になる。逆に分かれていれば、「どこまで止まっているか」を狭めやすい。

ここで必要なのは、難しい設計より、最低限の役割分離である。小さな会社なら、まずは社内用とゲスト用を分けるだけでもよい。余裕があれば、共用機器や IoT を別にする。いきなり複雑な VLAN 設計を作って維持できなくなるより、少数の役割分離を守る方が現実的である。

## Wi-Fi は SSID とパスワードだけでは設計したことにならない

Wi-Fi は手軽に見えるぶん、雑に運用されやすい。だが、無線は有線以上に境界を意識した方がよい。

NIST SP 800-153 では、WLAN は設計、導入、保守、監視のライフサイクル全体で考えるべきとしている。つまり、「パスワードを設定したから終わり」ではない。

Wi-Fi で最低限押さえないのは、次の論点である。

- SSID の役割分け
- 暗号化方式
- 認証方式
- 接続先ネットワーク
- クライアント同士の分離
- アクセスポイントやルーターの更新

設計項目を一枚で持つなら、次の表が最小形である。

論点	最低限決めること	ありがちな失敗
SSID の役割	社内用、ゲスト用、機器用のどれか	役割不明の旧 SSID が残る
暗号化方式	WPA3 または WPA2 AES を使う	古い方式を放置する
認証方式	共有パスワードか、802.1X か	どの SSID がどう認証しているか誰も説明できない
接続先	どのネットワークや VLAN へ入るか	ゲストが社内側へ届いてしまう

論点	最低限決めること	ありがちな失敗
client isolation	同一 SSID の端末同士をどう扱うか	ゲスト同士がむやみに届く
AP とルーター更新	いつ確認し、誰が更新するか	古いファームウェアを放置する

暗号化方式は、CIS でも secure protocols の例として 802.1X や WPA2 Enterprise 以上が挙げられている。Cisco Meraki の無線セキュリティ資料でも、WEP は安全ではなく、WPA3 や WPA2 Enterprise がより適した選択肢として整理されている。ひとり情シスとしては、少なくとも古い方式を放置しないこと、どの SSID がどの認証方式を使っているか説明できることが必要である。

また、ゲストや BYOD 向け SSID では、client isolation の考え方が重要になる。Meraki の資料でも、guest や BYOD 向け SSID で、無線クライアント同士の通信を抑える機能が有効だと整理されている。これは高度な追加機能ではなく、「同じ Wi-Fi にいる他人の端末へむやみに届かせない」ための基本である。

Wi-Fi 運用で地味に効くのは、SSID を増やしすぎないことだ。部署ごと、部屋ごと、用途ごとに増やすと、誰がどこにつなぐべきか分からなくなる。基本は少数の役割で分ける。社内用、ゲスト用、必要なら機器用。このくらいから始めた方が崩れにくい。

## テレワークは、入口とアクセス範囲を決める

在宅勤務や出張先からの接続は、「どこでも仕事できて便利」で終わらせると危うい。NIST SP 800-46 は、外部環境は敵対的である前提で、遠隔接続を設計すべきとしている。つまり、自宅回線もカフェ Wi-Fi も、社内ネットワークと同じ前提では扱えない。

ここで最初に決めたいのは、誰が何からどこへ入れるかである。NIST は、端末種別ごとにアクセス範囲を階層化する考え方を示している。たとえば、会社管理 PC は広く許可し、BYOD PC は限定し、スマホは web メール程度に抑える、といった考え方である。

この発想は、ひとり情シスにとってかなり有効である。全部を同じように許可しようとする、運用がすぐ崩れるからだ。最低限、次の区分は持ちたい。

- 会社管理 PC からの接続
- 私物 PC からの接続
- スマートフォンからの接続
- 外部委託先からの接続

それぞれについて、どの接続方式を認めるかを定める。CIS Control 12 でも、遠隔端末は企業管理 VPN と AAA 基盤へ接続してから業務資源へ入る考え方を示している。つまり、「必要なら好きな遠隔ツールでつないでよい」ではなく、会社が決めた入口を通す方が運用しやすい。

この時、接続元ごとの入口と許可範囲を表で持つと迷いにくい。

接続元	標準入口	許可範囲の考え方	追加条件
会社管理 PC	会社指定 VPN や承認済みの遠隔接続	業務に必要な社内資源まで許可	MFA、端末更新、ログ確認
私物 PC	なるべく SaaS や限定入口に絞る	社内 LAN へ広く入れない	利用目的、利用期限、最低限の端末条件
スマートフォン	メール、チャット、承認済みアプリ中心	閲覧や通知中心に抑える	会社データ削除の可否を決める
外部委託先	契約で定めた入口だけ	担当範囲の資源だけに限定	期限、記録、終了時停止

ここで注意したいのが、遠隔接続方式の乱立である。VPN、リモートデスクトップ、各種遠隔操作ツール、SaaS 側の個別ログインが場当たりに増えると、誰がどこから入れるか分からなくなる。CISA の遠隔接続ソフト保護ガイドも、正規ツールが攻撃に悪用される前提で、承認済みツールの利用と監視を求めている。テレワークを理由に接続入口を増やしすぎないことが、ひとり情シスにとって重要な原則となる。

また、在宅勤務ルールはVPN手順だけでは足りない。CISAは自宅Wi-Fiについて、ルーター更新、強いパスワード、WPA3 または WPA2 AES、不要なりモート管理の停止、接続端末の確認を勧めている。会社が自宅ネットワークを全面管理できなくても、「最低限ここは守ってほしい」という基準は持てる。

在宅勤務者へ配る最低条件は、次のくらいに絞ると現実的である。

項目	最低限守ってほしいこと	理由
ルーター更新	firmware を更新する	既知脆弱性を放置しない
Wi-Fi 暗号化	WPA3 または WPA2 AES を使う	古い暗号化を避ける
Wi-Fi パスワード	長く固有なものにする	使い回しを避ける
不要機能停止	不要なりモート管理を止める	外から勝手に入られにくくする
接続端末確認	見覚えのない端末がないか見る	不審利用を早く見つける

## 拠点回線とバックアップ経路も運用対象にする

ネットワーク機器は意識されやすいが、回線は見落とされやすい。だが、実際には主回線、ONU、ルーター、ISP 障害、工事、オフィス移転の方が業務へ大きく効くことがある。

ここで持ちたいのは、単一障害点の視点である。次のどれが止まると、どこまで止まるかを把握したい。

- 主回線
- 予備回線
- ONU やモデム
- 境界ルーターやファイアウォール
- 拠点間接続
- DNS や認証など接続に必要な周辺サービス

Cisco Meraki の MultiWAN 資料でも、複数 ISP やセルラー回線を組み合わせて可用性を上げるユースケースが整理されている。さらに、バックアップ回線有効化のような変更は、通信断を伴うためメンテナンス時間帯に行うことが推奨されている。これは製品固有の話ではなく、回線切替そのものが変更管理対象だという意味で重要である。

小さな会社でも、最低限の暫定手段は持っておきたい。たとえば次のようなものだ。

- スマートフォンのテザリング
- モバイルルーター
- 予備回線
- 一時的に別拠点へ業務を寄せる手順

単一障害点と暫定手段は、次のように見ておくと判断しやすい。

止まるもの	どこまで止まるか	暫定手段	先に決めること
主回線	全社のインターネット接続	予備回線、テザリング、モバイルルーター	切替条件、連絡先、業務継続の優先順

止まるもの	どこまで止まるか	暫定手段	先に決めること
境界ルーター、ファイアウォール	社内外通信全体	予備機、保守交換、設定バックアップ	設定保管先、保守窓口
拠点間接続、VPN	在宅勤務や別拠点接続	一時的な別入口、別拠点業務寄せ	誰にどこまで許可するか
DNSや認証	一見ネットが落ちたように見える障害	予備DNS、認証系確認	依存先を構成図に載せる

ここで大切なのは、完全無停止を目指すことではない。止まった時に、どこまで続けられるかを先に決めることである。受注だけは続けたいのか、メールだけ維持できればよいのか、在宅へ切り替えるのか。この判断があると、障害時の迷いが減る。

また、回線切替やバックアップ回線有効化は、思いつきで平日日中にやらない方がよい。Meraki の MultiWAN 資料が maintenance window を勧めているように、回線変更そのものが変更管理対象である。

## 障害時は、まずどこまで止まっているかを分ける

ネットワーク障害対応で最も重要なのは、最初の切り分けである。ここを外すと、社内で迷い、外部連絡も遅れる。

一次切り分けでは、まず次の順に分けるとよい。

1. 全社か、一部の人だけか
2. 有線も無線も両方か、Wi-Fi だけか
3. 社内だけか、在宅勤務者だけか
4. インターネット全体か、特定の社内サービスだけか
5. 直前に設定変更や工事があったか

この五つだけでも、行き先はかなり絞れる。たとえば、全社かつ有線も無線も両方なら、回線か境界機器を見る。Wi-Fi だけなら SSID、AP、無線側の DHCP や認証を見る。在宅勤務者だけなら VPN、認証、相手側の自宅回線や端末設定を見る。

症状ごとに最初に見る場所を決めておくと、さらに速い。

症状	最初に見る場所	次に考える連絡先
全社で有線も無線も止まる	回線、ONU、境界ルーター、DNS	ISP、保守業者、社内責任者
Wi-Fi だけ止まる	SSID、AP、無線認証、DHCP	無線機器保守、社内設定担当
ゲストだけ止まる	ゲスト SSID、接続先、分離設定	無線機器保守、設定担当
在宅勤務者だけ止まる	VPN、認証、承認済み遠隔接続入口	VPN 保守、認証基盤担当
特定サービスだけ使えない	そのサービス、DNS、経路	サービス管理者、ベンダー

重要なのは、「ネットワークが悪い」という一言で全部をまとめないことだ。問題の層を分けるだけで、連絡先も変わる。ISP へ連絡するのか、保守業者か、社内設定変更の切り戻しか、利用者の自宅 Wi-Fi 見直しか。この順番が決まっていると、ひとり情シスでも慌てにくい。

障害票やメモには、最低限次を残したい。

- 発生時刻
- 影響範囲
- 有線か無線か
- 拠点名
- 直前変更の有無

- 一次確認結果
- 外部連絡先
- 暫定対応

後で振り返る時にも、この情報が効く。同じ会議室だけ繰り返し切れる、同じ時間帯にVPNが遅くなる、といった再発傾向が見えてくるからである。

## 小さな会社で現実的に回るやり方

すべての拠点に高度な冗長化やゼロトラスト基盤を入れられる会社ばかりではない。小さな会社では、まず少ないルールを固定する方が効果が高い。

現実的には、次のくらいから始めると回しやすい。

- 構成図を1枚持つ
- 社内用とゲスト用のSSIDを分ける
- Wi-Fiの暗号化方式と認証方式を確認する
- 在宅勤務の接続方式を一つか二つに絞る
- 回線事業者と保守業者の連絡先を一覧化する
- ISP障害時の暫定手段を決める
- 障害一次切り分け票を1枚にする

ここで意図的に避けたいのは、中途半端に複雑な構成である。VLANもSSIDもVPNも乱立しているのに、誰も全体像を説明できない状態が一番危うい。少数の役割で分け、少数の接続方式に絞り、その代わり構成図と連絡先を最新に保つ方が、ひとり情シスには向いている。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスがネットワーク構成図を一枚持っている。主回線、予備回線、ルーター、主要スイッチ、社内 Wi-Fi、ゲスト Wi-Fi、VPN の入口が載っている。社内用とゲスト用の SSID は分かれ、ゲスト側は内部へ届かない。在宅勤務は会社指定の接続方式へ統一され、私物端末は閲覧系だけに限定されている。ISP 障害時には、まず主回線障害か内部機器障害かを切り分け、必要ならテザリングや予備回線へ切り替える。結果として、「どこで止まっているか」と「次に誰へ連絡するか」がすぐ決まる。

崩れた時の例では、Wi-Fi の SSID は増え続け、どれが社内用でどれが旧設定か分からない。来客も社員も同じ無線へ入り、会議室機器も同じネットワークにぶら下がる。在宅勤務は人によって使う接続方式が違い、VPN も遠隔操作ツールも増えている。回線障害が起きても、ISP の契約番号も保守窓口もすぐ出てこない。結果として、障害そのものより「どこから調べるか」で時間を失う。

## 最低限ここまではやる

第11章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 現在の構成図を 1 枚作る
2. 社内用とゲスト用を分ける
3. テレワークの標準入口を決める
4. 障害一次切り分け票と外部連絡先一覧を作る

この四つがあるだけで、ネットワークはかなり管理しやすくなる。ネットワーク管理とは、全部を自分で設定することではない。どこがどこにつながり、誰がどこまで使え、どこで止まるかを説明できる状態を保つことである。

## 今日、今週、後でやること

今日やることは、紙でもスライドでもよいので、今のネットワーク構成を一枚に書き出すことである。正確でなくてもよい。回線、境界機器、Wi-Fi、VPN の入口が見えるだけで価値がある。

今週やることは、使っている SSID とその接続先を棚卸しすることである。社内用なのか、ゲスト用なのか、誰向けなのか、内部へ届くのか。この整理だけでも、かなりの曖昧さが消える。

後でやることは、在宅勤務ルールと回線障害時の暫定運用を一枚にまとめることである。ネットワークは止まることがある。だからこそ、止まった時の動き方を先に決めておくことが、ひとり情シスの実務では大きい。

## 第12章

## SaaS、クラウド、業務アプリ管理

---

現場が「便利だから」と言って契約した SaaS は一つだけだった。ところが半年後に棚卸してみると、その SaaS は Google アカウントでログインし、Zapier とつながり、ファイルは別のストレージへ送られ、通知はチャットへ流れ、外部委託先とも共有されていた。契約書の枚数は増えていないのに、データの入口と出口は増えていた。

SaaS 管理が難しいのはここにある。問題は、サービス本体だけではない。誰が使っているか、どの連携アプリが権限を持っているか、外部共有がどこまで開いているか、解約する時に何を持ち出せるか。こうした周辺まで見ないと、SaaS はすぐに統制の外へ広がる。

第5章では、調達と契約を設計として扱った。第8章では、認証と権限の原則を見た。第12章では、その先にある実運用を扱う。ここで扱うのは、製品比較ではない。導入後の SaaS とクラウドサービスを、少人数でも崩れにくく管理するための考え方である。

### SaaS 管理は、利用実態を見える化するところから始まる

SaaS を契約一覧だけで見ていると、実態を見失う。契約していることと、使われていることは同じではないからだ。

たとえば、次のようなずれはよく起きる。

- 契約はあるが、ほとんど使われていない
- 契約した担当者が退職し、管理者が曖昧になっている

- 全社利用のつもりが、一部署だけで別運用になっている
- 無料枠や個人契約のまま使われている
- SaaS 本体より、OAuth 連携や API 連携の方が先に増えている

Microsoft Defender for Cloud Apps が cloud discovery を最初の重要項目として置いているのも、どの cloud apps が、誰に、どの端末で使われているかの可視化が先だからである。第12章で最初に持ちたいのは、契約台帳ではなく運用台帳である。

運用台帳の最小形は、次のくらいでよい。

項目	何を書くか	抜けると困ること
サービス名	正式名称、プラン名、利用 URL	同名の別サービスや無料版と混同する
用途	何の業務に使うか	代替可否や停止影響を判断できない
管理者	管理者アカウント、担当部門	退職や異動で管理が空く
利用者範囲	全社、一部部門、委託先利用の有無	棚卸し範囲が曖昧になる
取り扱いデータ	顧客、人事、会計、社内文書など	共有や連携の重さを判断できない
認証方式	SSO、個別 ID、MFA 有無	JML や棚卸しが崩れる
外部共有	共有の有無、相手先、認証条件	公開境界が見えない
主要連携	OAuth、API、ノーコード連携、service account	本体以外のデータ経路を見落とす
export と監査ログ	何を出せるか、どこで見るか	解約や調査で詰まる
最終確認日	いつ見直したか	古い情報のまま運用される

ここで重要なのは、何を契約したかより **今どう使われているか** を書くことだ。たとえば営業支援 SaaS 一つを取っても、顧客データが入るのか、外部委託先へ共有するのか、他サービスへ API 連携しているのかで、管理の重さは変わる。

見える化の入口は、まず手元にある管理画面からでよい。IdP のエンタープライズアプリ一覧、Google Workspace や Microsoft 365 の管理画面、請求一覧、主要部門への聞き取り。この四つだけでも、実態はかなり見え始める。

## 野良 SaaS を見つけ、承認済みの型へ寄せる

現場が先に使い始める SaaS を、完全にゼロにするのは難しい。だから必要なのは、見つけて、分類して、統制下へ寄せることである。

野良 SaaS の問題は、規程違反そのものより、見えていないことである。だから見つかった時の扱いを先に決めておいた方がよい。ひとり情シスとしては、次の三分けが実務で使いやすい。

区分	どう扱うか	最低限の判断軸
承認済み	会社として標準利用する	管理者を置ける、SSO や MFA を使える、共有と連携を制御できる
要審査	条件付きで判断する	取り扱いデータ、外部共有、export 可否、監査ログ、利用部門の必要性
禁止	代替先へ寄せる	機密データを扱えない、権限や共有制御が粗い、終了時の回収が難しい

ここで大切なのは、**見つけたら叱る** で終わらせないことだ。まず用途を確認し、承認済みサービスで代替できるなら寄せる。代替できないなら、要審査として統制条件を満たせるかを見る。この順番の方が実務では回る。

審査時には、少なくとも次を見たい。

- どのデータを入れるか
- 個人契約ではなく会社管理に寄せられるか
- SSO や MFA を使えるか
- 外部共有やゲスト招待を制御できるか
- export と削除の方法があるか
- 主要連携を一覧で見られるか

## OAuth、API、ノーコード連携は別の入口として管理する

SaaS 管理で特に見落とししやすいのが、連携アプリである。ユーザーは **ログイン** しただけ **便利機能を有効にした** だけのつもりでも、実際にはデータへの新しい入口が増えていることがある。

Google Workspace の app access control は、Google-owned、Internal、Third-party の三種類のアプリを区別し、Trusted、Specific Google data、Limited、Blocked という水準で制御できる。requested scopes、verified status、ユーザー数まで見える。Microsoft Entra でも、ユーザー同意を tenant 側で制御でき、verified publisher に絞る考え方が示されている。つまり、SaaS 本体と連携アプリは、分けて管理する前提で作られている。

連携は、少なくとも次のように分けて見たい。

連携種別	最低限見ること	詰まりやすい点
ユーザー同意の OAuth アプリ	誰が同意したか、scope、verified か、今も使うか	利用者本人も許可内容を覚えていない
管理者同意のアプリ	全社影響、管理者名、取り消し経路	一度許可して放置しやすい

連携種別	最低限見ること	詰まりやすい点
API キー連携	発行者、保存場所、用途、 停止手順	退職や引き継ぎで所在不明 になる
ノーコード連携	どの条件でどこへデータを 送るか	便利な自動化が勝手な転送 経路になる
service account	作成者、用途、権限、ロー テーション、停止条件	人にひも付かず棚卸しから 漏れやすい

特に、Zapier のような自動化は便利だが、複数サービスの間新しいデータ経路を作る。どのデータが、どの条件で、どこへ送られるのかを説明できない状態で増やすと危うい。

Microsoft は consent phishing という形で、悪意あるアプリへ同意させる攻撃を明示している。つまり、連携アプリは利便性の問題であると同時に、攻撃経路の問題でもある。だから第12章では、少なくとも次を見たい。

- 誰が許可したか
- 何の権限を要求しているか
- verified publisher か
- 今も使われているか
- 不要になった時、どこで revoke するか

ここで注意したいのは、**verified** だから安全 と言い切らないことだ。verified は重要な条件だが、scope が広すぎる、用途が曖昧、業務上不要、取り消し経路が分からないなら止める方がよい。

## 外部共有は、機能ではなく公開境界である

SaaS やクラウドアプリの共有設定は、単なる便利機能ではない。どこまでを社外へ見せるかという公開境界である。

Google Workspace の Drive では、外部共有は data leaks のリスクを伴うとして、warning や link sharing の制限、組織部門やグループごとの制御、共有専用 shared drive の活用が案内されている。Microsoft SharePoint でも、Anyone、New and existing guests、Existing guests、Only people in your organization というように、外部共有の水準を複数段階で持っている。

ここで重要なのは、共有を **オンかオフか** で終わらせないことだ。実務では、少なくとも次を分けて考えたい。

論点	会社として決めること	典型的な選択肢
誰が共有できるか	全員か、一部管理者か、特定部門だけか	全員、承認済み部門、特定 security group
誰と共有できるか	取引先だけか、任意の外部か	特定ドメイン、既存ゲストのみ、任意外部
認証が必要か	相手の本人確認を求めるか	組織内のみ、認証済みゲストのみ、匿名リンク可
リンク転送で見られるか	認証なし URL を許すか	禁止、限定許可、広く許可
どの領域で共有を許すか	全体か、一部サイトや shared drive だけか	共有専用領域のみ、案件別領域のみ、全体

Microsoft は、特定の security group だけに外部共有権限を持たせる方法も案内している。Authenticated guests only を選べば、外部共有しても相手の認証が必要になる。Google でも、組織部門、グループ、shared drive 単位で共有境界を設計できる。つまり、共有は **全員がどこでも誰にでもできる** 形にしておく必要はない。

また、既定値を放置しない方がよい。SharePoint では site type や OneDrive 側の設定によって、既定の共有範囲が広くなりうる。**特に変えていないから安全だろう**は危うい。第12章では、共有可能者、共有相手、認証要件、共有可能領域を、会社側で先に決める考え方をもちたい。

## クラウドの責任分界を理解する

クラウドでは、提供者が多くの運用を担う。だが、それは責任が消えることと同じではない。

Microsoft Azure は、すべてのクラウド展開形態で、データと ID は顧客が持つ責任だと明示している。Google Cloud も、顧客は自社要件に必要な security controls を自ら設定しなければならないとしている。AWS も、サービス形態に応じて責任分界は変わるが、データ管理、資産分類、IAM による権限付与は顧客側責任だと整理している。

この共通点を、第12章では次のように見たい。

形態	提供者が多く担うこと	自社に残ること	第12章で最低確認したい点
SaaS	基盤運用、アプリ提供、可用性の多く	データ分類、共有、管理者権限、連携アプリ、退職時停止	誰が管理者か、共有をどこまで許すか、出口があるか

形態	提供者が多く担うこと	自社に残ること	第12章で最低確認したい点
PaaS	OSや基盤の一部運用、実行環境	アプリ設定、秘密情報、アクセス制御、ログ確認	何を設定し、誰が見直すか
IaaS	物理設備、仮想化基盤	OS、ネットワーク設定、IAM、保存データ、監査	構成、権限、ログ、バックアップを誰が持つか

ひとり情シスの実務では、ここで全部を厳密に分類することが目的ではない。大事なのは、**ベンダーに聞けば何とかなる** と考えないことだ。SaaS でも、共有設定、管理者権限、OAuth 連携、データ出口、退職時の無効化までは自社責任であることが多い。

## 利用終了時の出口を先に決める

SaaS は入れる時より、終わらせる時に詰まりやすい。請求停止だけで終わらないからである。

利用終了時には、少なくとも次を見たい。

見る点	先に決めること	詰まりやすい点
export 対象	何のデータを持ち出すか	添付ファイルや共有領域が漏れる
保存先	どこへ保管するか	受け皿が決まっておらず一時保管が散らばる
形式	CSV、ZIP、原形式、監査ログなど	欲しい形式で出せない
所要時間	いつ開始し、いつ完了見込みか	退職日や解約日に間に合わない

見る点	先に決めること	詰まりやすい点
所有権移転	メール、ファイル、共有領域の引き継ぎ先	共有ドライブや共同所有データが残る
連携停止	OAuth、API、ノーコード連携をいつ止めるか	旧 SaaS へデータが流れ続ける
削除確認	いつ削除し、何を証跡にするか	停止だけしてデータが残る

Google Workspace の Data Export tool では、export は最短でも 48 時間後、通常 72 時間、場合によっては 14 日かかる。Google 提供バケットに出した場合は開始から 60 日で自動削除される。しかも、個別ユーザー export では shared drives のような **個人が所有しないデータ** は含まれず、shared drive data は time range で絞れない。つまり、**退職日までに急いで全部ダウンロードしておこう** という発想は危うい。

この考え方は Google Workspace に限らない。どの SaaS でも、導入前に次を確認した方がよい。

- export 機能があるか
- 監査ログを出せるか
- 共有データの所有権を移せるか
- 解約後の保持期間はどうか
- API 連携先をどう止めるか

第12章では、**導入前に出口を見る** という癖を持たせたい。出口を考えずに始めると、移行や解約のたびに人が張り付く。

## 小さな会社で現実的に回るやり方

すべての SaaS を高度な CASB や SaaS セキュリティ 製品で管理できる会社ばかりではない。小さな会社では、まず少数のルールを固定する方が効く。

現実的には、次のくらいから始めると回しやすい。

- 主要 SaaS を運用台帳にする
- 承認済み、要審査、禁止を分ける
- 重要 SaaS の OAuth 連携を棚卸しする
- 外部共有設定を一つずつ確認する
- 解約前チェック欄を作る

回し方も、月次と四半期で分けると重くなりにくい。

頻度	やること
月次	新しく増えた SaaS、主要連携、管理者変更の確認
四半期	重要 SaaS の共有設定、管理者権限、不要連携、出口情報の見直し

台帳に最初から全部を入れなくてもよい。まずは、顧客データ、人事データ、会計データ、社内ファイルに触るものから始める。そこに管理者、認証方式、主要連携、外部共有、export 可否の欄を足すだけでも、かなり実務が軽くなる。

また、例外を減らすことも大切である。部門ごとに別ツール、担当者ごとに別連携、管理者ごとに別ルールが増えると、SaaS の数以上に運用が膨らむ。小さな会社ほど、使うサービスと接続方式を絞った方がよい。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスが主要 SaaS の運用台帳を持っている。そこには、用途、管理者、認証方式、取り扱いデータ、主要連携、外部共有、出口欄が入っている。現場が新しい SaaS を使いたいと言ったら、まず承認済みサービスで代替できないかを見る。必要なら要審査として、データ種別、OAuth 連携、export 可否、共有境界を確認する。Google Workspace や Microsoft 365 の連携アプリは定期的に見直し、不要な同意は外す。結果として、使うサービスは増えても、何がどこへ届くかを説明できる。

崩れた時の例では、契約一覧はあるが、利用実態が分からない。現場は勝手に SaaS を試し、Google や Microsoft アカウントでログインして、便利そうな連携にそのまま同意する。共有設定は既定値のまま、誰でも外部共有できる。いざ退職者対応や解約の話になると、どのアプリを止めるべきか、どこにデータがあるか、何が export できるかが分からない。結果として、SaaS が増えるほど、会社側の把握が追いつかなくなる。

## 最低限ここまではやる

第12章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 主要 SaaS の運用台帳を持つ
2. 承認済み、要審査、禁止の三分けを作る
3. OAuth や API 連携を別管理する
4. 解約前にデータ出口を確認する

この四つがあるだけで、SaaS 管理はかなり崩れにくくなる。SaaS 管理とは、何を契約したかを覚えることではない。誰が使い、何と連携し、どこへ共有し、どう終わらせるかを会社として把握し続けることである。

## 今日、今週、後でやること

今日やることは、主要 SaaS を五つだけ選び、用途、管理者、認証方式、主要連携、外部共有を書き出すことである。空欄が多いなら、そこが最初の詰まりどころである。

今週やることは、その中で一つの Google Workspace 連携か Microsoft 365 連携を見て、どのアプリが何の権限を持っているか確認することである。ここを見るだけでも、連携アプリ管理の感覚が変わる。

後でやることは、主要 SaaS の外部共有設定と export 可否を順に確認することである。入口と出口が見えると、SaaS は初めて会社の管理対象になる。

## 第13章

## メール、チャット、ファイル共有の運用

---

会社の問い合わせ窓口は一応ある。sales@ も support@ も存在する。だが実態は、どちらも特定の担当者の個人メールへ転送されている。日中の相談はチャットで流れ、取引先へ送る最新版資料は添付で往復し、誰が何を確定したのかは後から追にくい。担当者が休むと返事が止まり、退職すると過去経緯の所在が分からなくなる。

メール、チャット、ファイル共有は、いまやどの会社にもある。だからこそ整備されているように見えやすい。だが実際には、最も日常的に使う基盤ほど、慣習だけで回りやすく、個人依存しやすい。第13章で扱うのは、ここをどう崩れにくくするかである。

第12章では SaaS やクラウドの統制を扱った。第13章では、その上で毎日の連絡と共有をどう運用するかを見る。論点は単純である。どの手段を何に使うか。共有窓口をどう持つか。社外とどこまでつなぐか。誤送信をどう減らすか。この四つが曖昧だと、どれほど高価なツールを入れても運用品質は上がらない。

### メール、チャット、ファイル共有の役割を分ける

最初に決めたいのは、どの道具を何に使うかである。ここが曖昧だと、連絡は速くなくても、後から探せない、代理対応できない、最新版が分からない、という問題が残る。

最小限の役割分担は、次のように整理しやすい。

手段	向く用途	決まった後に戻す場所	避けたい使い方
メール	社外への正式連絡、依頼、承認、記録として残したいやりとり	共有受信箱、案件記録	最新版ファイルを添付で往復し続ける
チャット	短い相談、状況確認、即時連携、軽い調整	決定事項メモ、チケット、案件記録	重要な決定をチャットだけで閉じる
ファイル共有	最新版の保管、共同編集、閲覧提供	共有ドライブ、チームサイト、案件フォルダ	正本が分からないまま添付とリンクを併用する

ひとり情シスとしては、厳密な規程をいきなり作る必要はない。まずは、次の三点だけでも共有しておくとうい。

1. 社外への正式連絡はメールを基本とする
2. 最新版を残すものは添付でなく共有先を基準にする
3. チャットで決まった重要事項は、後から見つけられる場所へ戻す

この三つがあるだけで、連絡基盤の散らかり方はかなり変わる。

## 代表アドレスと共有対応は個人メールで回さない

第13章で最も強く言いたいのはここである。問い合わせ、採用、請求、総務連絡のような会社窓口を、個人メールへの転送で回してはいけない。

理由は単純である。個人メールに寄せると、その人の在席が運用の前提になるからだ。休暇、退職、異動、委託先交代のたびに、転送設定、代理送信、過去経緯の確認で詰まる。第9章で退職時のメール処理を扱ったが、本来はその前に、会社窓口を個人から切り離しておくべきである。

実務では、次のように使い分けると整理しやすい。

方式	向く場面	最低限決めること	避けたいこと
Shared mailbox	問い合わせ、請求、採用のような恒常窓口	管理者、閲覧者、Send As / Send on Behalf、対应当番	対応用アカウントへ直接サインインさせる
Gmail delegation	少人数での代理対応、個人宛てだが代理が必要な時	誰を delegate にするか、いつ外すか	パスワード共有で代用する
Collaborative inbox	問い合わせを複数人で割り当て、状態管理したい時	担当割り当て、未対応の確認方法、完了基準	ただのメーリングリストとして放置する
個人メール転送	暫定措置に限る	終了日、移行先、暫定であることの明示	恒久運用にする

Microsoft 365 の shared mailbox は、共通アドレスで受信し、複数人で返信するための仕組みである。しかも、見る権限と送る権限は別で、Full Access だけでは足りず、Send As か Send on Behalf を別に与える必要がある。Google の delegation でも同じで、delegate はパスワード共有なしに読み書きできるが、password 変更や chat はできない。つまり、共有窓口は **みんなで同じアカウントを使う** のでなく、**各自のアカウントに権限を配る** 形に寄せる方がよい。

shared mailbox では、作成時に対応する user account もできるため、その sign-in を block しておく考え方も重要である。共有窓口のために作ったアカウントへ直接ログインさせると、結局 **共通 ID をみんなで使う** 状態へ戻りやすいからだ。

Google の collaborative inbox が usefulなのは、ただ複数人で読めるだけでなく、担当割り当て、未対応の抽出、完了や duplicate の状態管理まで持てる点である。support@ のような窓口で、誰が拾い、どこまで終わったかを見たいなら、単純な転送よりこちらの方が向いている。

共有窓口では、少なくとも次は決めたい。

- どのアドレスを共有窓口にするか
- 誰が見られるか
- 誰が送れるか
- 誰が管理者か
- 不在時は誰が一次対応するか

問い合わせ窓口が個人依存から外れるだけで、会社の連絡基盤はかなり強くなる。

## チャットの外部協業は、接続モデルを決めてから開く

社外とチャットできる機能は便利である。だが便利だからといって、同じ感覚で外部接続を増やすと、どこまで見えているのかが曖昧になる。

ここで大事なのは、**相手と話したいだけか 相手を内部空間へ入れたいのか** を分けることだ。実務では、次のように整理すると迷いにくい。

接続モデル	何ができるか	向く場面	終了時にやること
メールへ戻す	記録を残した正式連絡	単発依頼、合意、契約連絡	スレッド保管、案件記録へ戻す
外部チャット	相手と話すか、内部空間へは入れない	短い調整、日程連絡、一次確認	外部連絡先の見直し
外部参加スペース、チーム、チャンネル	会話だけでなく、ファイルや作業空間も共有する	案件期間中の共同作業	外部参加者除去、共有ファイル見直し
恒常接続	保守会社、顧問、委託先との継続連携	定例運用、障害一次連絡	半期ごとの接続先棚卸し

たとえば Teams では、external access は主に chat と meetings の相互接続であり、teams や sites、その他 Microsoft 365 resources への access は与えない。一方で guest access は、teams、documents in channels、resources、chats、applications まで含む協業モデルである。同じ 外部と Teams でやりとりするでも、見せる範囲は大きく違う。

Teams の external access は、組織設定だけでなく user policy 側の条件も要る。したがって、設定は on のはずなのにつながらないという時は、tenant 全体だけでなく対象ユーザー側の policy も見た方がよい。

Google Chat でも同様で、外部ユーザーと 1:1 で話すのか、external members を含む space を作るのかで運用の重さが変わる。しかも、Google Chat は外部ユーザーを group message へ追加できず、1:1 か external space で考える必要がある。さらに、外部ユーザーを含む space かどうかは作成時に決める必要があり、後から切り替えられない。つまり、案件開始時に接続モデルを決めずに作ると、後から詰まりやすい。

さらに重要なのは、閉じ方である。Google Chat は、external chat を off にしても、既存の external messages、conversations、spaces は自動で消えない。space に共有した Drive files や tasks も別扱いで、space から外しただけでは片付かないことがある。Slack Connect でも、channel から外すだけでは communication 全体は終わらず、必要なら disconnect まで考える必要がある。

社外チャットを開く前に、少なくとも次を決めたい。

- これは 話すだけ か 内部空間へ入れる のか
- 誰が承認するか
- どの案件、どの期間で使うか
- 終了時に誰が外部参加者を外すか
- 共有ファイルをどこまで見直すか

## ファイル共有は、リンクより境界と期限を決める

ファイル共有でもっとも避けたいのは、最新版が添付とリンクの両方に散らばることである。見積書、提案書、手順書のように更新されるものは、基本的には共有先を正本にした方がよい。

ただし、共有リンクなら何でもよいわけではない。第13章で重視したいのは、共有のしやすさより、誰がどこまでいつまで見られるかである。最低限、次の四点は意識したい。

- 対象者は誰か
- 相手の認証は必要か
- 期限を付けるか
- 再共有を許すか

共有方法は、次のように分けると運用しやすい。

場面	共有方法	最低条件	避けたい形
社内共同編集	社内限定の共有先	正本の置き場を固定する	個人フォルダを正本にする
取引先レビュー	認証ありの個別共有	相手、権限、期限を決める	Anyone with the linkで渡す
案件協業	案件用の共有領域	外部参加者の範囲、終了日、再共有可否を決める	全社共通領域へ外部を入れる
公開配布	公開前提の専用導線	公開対象かを確認する	機密資料と同じ置き場から公開する

Google Drive では、eligible work or school accounts なら Viewer・Commenter 権限の共有相手へ expiration date を付けられる（Editor 権限には設定できない）。owner は、editors の再共有や権限変更、viewers と commenters の download / print / copy を抑止できる。逆に、Anyone with the link はサインインなしで見られる。つまり、リンカー一つでも、認証、期限、再共有可否の重さはかなり違う。

もう一つ見落とししやすいのが、チャットとファイル共有のつながりである。Google Drive では、Chat space に grant access したファイルは、後からその space に参加した人にも見える。つまり、この人へ渡したのではなく、この場に入る人へ見える状態を作ったことになる。Teams や Slack でも似たことが起きる。ファイルを置く場所と会話の場所が結びついているからだ。

だから、社外と共有するファイルは、案件用、委託先用、社内限定用のように領域を分けた方がよい。誰にでも同じ共有ドライブや同じチームサイトを使わせると、境界が曖昧になる。

## 誤送信と誤共有は、送る前に防ぐ

誤送信対策というと、送ってしまった後に取り消せるか を考えたくなる。だが実務では、そこに期待しない方がよい。Microsoft の cloud-based message recall は、同一組織内に限られ、他組織やインターネット越しの宛先には頼れない。最大 24 時間試行されるとはいえ、社外誤送信の救済策として考えるのは危うい。

送信前に効く対策は、派手ではないが実用的である。

確認点	なぜ見るか	最低限の運用
外部宛先	社外へ出ると回収しにくい	社外宛ては一呼吸置く
添付かリンクか	添付は版ずれしやすい	更新物は共有先を渡す

確認点	なぜ見るか	最低限の運用
共有範囲	リンク先が広すぎることがある	送る前に認証条件を確認する
宛先数	配布ミスが起きやすい	大量宛先や配布先は再確認する

MailTips のような事前警告が使える環境なら活用した方がよい。ただし、External Recipients MailTip は既定で off であり、有効化が必要である。つまり、**機能があるはず**ではなく、実際に有効かを確認しなければいけない。

小さな会社なら、送信前の確認は次の三つで十分である。

1. 外部宛先は含まれていないか
2. 添付やリンクの中身は合っているか
3. この相手に本当にこの範囲まで見せてよいか

## 代理対応、引き継ぎ、不在時運用に崩れない設計

連絡基盤の品質は、平常時より、崩れた時に分かる。担当者が休む。退職する。委託先が変わる。災害や障害で普段の連絡先へ届かない。そういう時に止まる設計は、最初から脆い。

崩れにくくするためには、日常運用の時点で次を共有化しておきたい。

共有しておくもの	最低限の形	抜けると何が起きるか
代表アドレス	共有窓口、管理者、一次対応者	休暇や退職で窓口が止まる
主要チャネル	案件名、参加者、外部参加有無	誰がどこで話しているか分からない
正本ファイル置き場	最新版を置く共有先	添付とリンクが混ざる
決定事項の記録先	議事メモ、チケット、案件記録	後から経緯を追えない

ここで言う共有化は、全員が何でも見られるようにすることではない。担当を明確にしつつ、個人一人がいなくなっても止まらない状態にすることである。たとえば、営業問い合わせは shared mailbox へ入る。一次返信担当は営業リーダーと情シス補助の二名にする。見積 TEMPLATE や最新資料は共有フォルダへ置く。チャットで決まったことは案件メモへ戻す。こうしておけば、誰か一人が不在でも業務は続く。

第9章では退職時の停止、失効、引き継ぎを扱った。第13章で伝えたいのは、その作業を軽くする鍵は、通常時から共有窓口へ寄せることだという点である。退職者メールボックスから過去経緯を掘り出す運用は、あくまで救済策であって標準ではない。

## 小さな会社で現実的に回るやり方

すべての会社が、高度なメール保護や外部協業管理製品を入れられるわけではない。小さな会社では、まず例外を減らす方が効く。

現実的には、次のくらいから始めるとよい。

- 代表アドレスを三つだけ共有運用にする
- 取引先との恒常窓口は個人メールへ転送しない
- 社外チャットは案件単位で許可し、終了時に見直す
- 共有ファイルは、社外向け領域を分け、期限付き共有を基本にする
- 重要事項はチャットだけで閉じず、後から見つかる場所へ残す

月次と案件終了時だけでも、次を見直すとかなり崩れにくい。

頻度	やること
月次	代表アドレスの担当者、主要チャネルの外部参加者、共有リンクの広さを確認する

頻度	やること
案件終了時	外部参加者除去、共有リンク無効化、正本ファイルの移管、決定事項の保管を行う

この章のポイントは、完璧な統制ではない。迷った時に戻る基準を持つことである。たとえば、判断に迷うなら次のように考えるとよい。

- 社外との正式な依頼や合意ならメールへ戻す
- 最新版を参照してほしいなら添付でなく共有先を渡す
- 外部参加者を内部空間へ入れる必要が弱いなら、まずは外部チャットかメールで始める

単純な基準がある方が、現場は回る。

## 通常時の例と、崩れた時の例

通常時の例では、問い合わせ窓口、採用窓口、請求窓口が shared mailbox が collaborative inbox で管理されている。担当者は複数名おり、送信名義も決まっている。社外との案件チャンネルは、案件単位で作られ、終了時に外部参加者を見直す。最新版資料は共有フォルダか共有ドライブに置き、メールではリンクを送る。社外共有は認証必須を基本にし、必要に応じて期限を付ける。誤送信防止は、外部宛先、添付、共有範囲の三点確認で回している。結果として、担当者が休んでも、顧客対応と案件進行は止まらない。

崩れた時の例では、support@ は担当者一人の個人メールへ転送されている。顧客とのやりとりは個人受信箱に散らばり、最新資料は添付で往復している。社外とのやりとりは個別のチャットで進み、誰が外部参加者として残っているか分からない。案件が終わっても、外部スペース、共有リンク、共有ファイルが残り

続ける。退職や引き継ぎのたびに、メール転送、共有リンク、過去ファイルの所在確認が必要になる。結果として、道具はそろっているのに、業務は人について回る。

## 最低限ここまではやる

第13章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 代表アドレスを個人メール依存から外す
2. メール、チャット、ファイル共有の役割を決める
3. 社外チャットの接続モデルを分ける
4. 誤送信前の三点確認を決める

この四つがあるだけで、日常運用の事故と引き継ぎ負荷はかなり減る。メール、チャット、ファイル共有の運用とは、便利に連絡できることではない。誰が不在でも、どこまで社外とつながっても、業務が止まらず、後から追える状態を保つことである。

## 今日、今週、後でやること

今日やることは、会社窓口として使っているメールアドレスを洗い出し、そのうち個人メールへ転送されているものを確認することである。一つでも見つければ、そこが最初の改善点である。

今週やることは、代表アドレスを一つだけ共有運用へ移すことである。shared mailbox、delegation、collaborative inbox のどれでもよい。とにかく、個人依存を一つ減らす。

後でやることは、社外チャットと社外共有リンクの運用を見直し、案件終了時の見直し手順と、共有期限の標準を決めることである。ここまでできると、連絡基盤は単なる便利ツールから業務基盤へ変わる。

## 第14章

# サーバー、Web、ドメイン、DNS、証明書

---

会社の Web サイトは普通に見えている。採用ページも、お問い合わせフォームも、一応動いている。だから問題はないように見える。ところが、ドメイン更新日を聞くと誰も答えられない。DNS は制作会社が管理しているらしいが、どこのサービスか分からない。証明書更新は「たぶん自動です」と言われるが、何がどう自動なのかを社内ですべて説明できる人がいない。こういう状態は珍しくない。

公開資産は、普段静かである。社内端末のように毎日触るわけでもなく、ネットワークのように不調が目立つわけでもない。だからこそ、責任者不明のまま放置されやすい。だが、ドメインが切れればサイトもメールも止まる。DNS を誤れば閲覧先が変わる。証明書が切れれば利用者は不安になる。公開サーバーの更新が止まれば侵入口になる。第14章で扱うのは、この静かな資産をどう見失わないかである。

第11章では接続基盤、第12章ではクラウド統制、第13章では日常の連絡基盤を扱った。第14章では、その外側にある公開基盤を見る。論点は、サーバーを持つかどうかより先に、公開資産の棚卸し、責任分界、期限管理、変更手順である。

この章では、製品名より先に次の四原則を押さえたい。

- 公開資産は、まず一覧化する
- ドメイン、DNS、公開先の責任分界を分けて持つ
- 期限管理は、人の記憶でなく通知と台帳で回す
- 変更は、控えと切り戻し先を持ってから行う

## 公開資産の管理は、サーバーの前に一覧化から始める

第14章で最初に持ちたいのは、公開資産台帳である。サーバーがある会社だけでなく、外注サイトや SaaS 型 CMS を使っている会社でも必要だ。ここで言う公開資産とは、Web サーバーだけではない。ドメイン、DNS、ホスティング、CDN、証明書、CMS、通知先、管理者アカウントまで含む。

まずは、台帳の列に意味を持たせたい。

項目	何を書くか	抜けると何が起きるか
ドメイン、サブドメイン	example.co.jp、www、recruit、form など	使っている公開面が分からず、古い資産が残る
主用途	会社サイト、採用、問い合わせ、LP、ポータルなど	何が止まると困るか判断できない
レジストラ	どこでドメイン契約を持っているか	更新、移管、請求の窓口が分からない
DNS ホスト	どこで DNS レコードを持っているか	どこへログインすれば切替できるか分からない
公開先	hosting、CDN、SaaS CMS、VPS など	障害時の連絡先と責任分界が曖昧になる
Web アプリ、CMS	WordPress、静的サイト、独自アプリなど	更新対象と攻撃面が見えない
証明書方式	CDN 側、ACME、自動、手動など	期限切れと失敗通知の見落としが起きる
管理者	社内責任者、委託先、保守窓口	問い合わせと判断の持ち主が分からない
通知先	更新通知、失敗通知、請求通知の宛先	退職者や個人メール依存が残る

項目	何を書くか	抜けると何が起きるか
次の期限	契約更新日、有効期限、見直し日	期限が人の記憶依存になる

最初の公開資産台帳は、次の形で十分である。

公開資産	主用途	レジストラ	DNS ホスト	公開先	CMS、アプリ	証明書	管理者	通知先	次
example.co.jp	会社サイト、メール	取得先名	DNS 提供先名	managed hosting	WordPress	自動更新	情シス	infra@	20
recruit.example.co.jp	採用ページ	同上	同上	SaaS 型 CMS	なし	CDN 側で自動更新	採用責任者	corp-admin@	20
form.example.co.jp	問い合わせフォーム	同上	Cloudflare	SaaS フォーム基盤	なし	SaaS 側	総務	support@	20

ここで重要なのは、**会社サイト** という一項目で終わらせないことである。実際には、本体サイト以外にも、採用サイト、キャンペーン用 LP、外注フォーム、旧製品サイト、委託先向けポータル、検証用のまま残ったサブドメインがぶら下がっていることが多い。

公開資産の事故は、主力サイトより、こうした周辺から起きやすい。理由は単純で、誰も見ていないからだ。だから一覧化では、重要資産だけでなく、古いもの、止めたつもりのもの、担当不明のものを拾う必要がある。

## オンプレとクラウドの使い分けは、維持管理責任で決める

公開基盤を考える時、つい「オンプレかクラウドか」という言い方をしたくなる。だが、ひとり情シスにとって本当に重要なのは、誰が更新責任を持つかである。

NIST のサーバー保護資料が示す通り、サーバーを持つということは、選定、実装、更新、監視、バックアップまで継続して持つということである。OS、Webサーバー、ランタイム、ミドルウェア、CMS、ログ、証明書を見続けられないなら、自前の公開サーバーは重い。

小さな会社では、次の順で考える方が現実的である。

方式	提供者が主に持つもの	自社に残るもの	向く場面
static hosting	配信基盤、証明書自動化、CDN	コンテンツ更新、ドメイン、DNS、公開判断	会社案内、採用、LP
managed hosting	OS や Web 基盤の一部保守、証明書、バックアップ機能の一部	CMS、plugin、公開内容、委託先管理	WordPress などの定番 CMS
SaaS 型 CMS、ノーコード公開基盤	アプリ基盤、証明書、可用性の多く	ドメイン接続、権限、公開範囲、契約管理	制作速度を優先する時
VPS、自前サーバー	ほぼ何も自動で持たない	OS、Web、ミドルウェア、アプリ、バックアップ、監視、証明書の全部	独自アプリ、特殊要件が強い時

自前サーバーが必要になるのは、独自アプリ、特殊なミドルウェア、細かな制御、社内要件など、持つ理由が明確な時である。逆に、会社案内、採用ページ、問い合わせ窓口程度なら、まずは運用負荷の少ない形を優先した方がよい。

ここでの結論は単純だ。公開サーバーは、作れるかではなく、維持できるかで決める。持つ理由が弱いのにサーバーだけ増えると、後で更新漏れの温床になる。

## ドメインと DNS は、契約先と責任分界を分けて管理する

公開資産で特に混乱しやすいのが、ドメインと DNS を同じものだと思ってしまうことである。実際には、次は別かもしれない。

- ドメインを登録しているレジストラ
- DNS レコードを持っている DNS ホスト
- Web サイトやアプリを置いているホスティング先
- 証明書を発行、終端している CDN やサーバー

この四つが別でも普通である。だからこそ、どこにログインすれば何を変えられるかを分けて把握しておきたい。責任分界は、次の形で整理しやすい。

論点	どこを見るか	事故時に何が止まるか	社内で持つべきもの
ドメイン契約	レジストラ	失効すると Web、メール、各種認証が止まる	更新日、支払い方法、通知先
DNS レコード	DNS ホスト	接続先誤り、切替失敗、メール不達が起きる	レコード一覧、変更権限、切替手順
公開先	hosting、CDN、SaaS	サイト表示、アプリ接続が止まる	障害窓口、責任範囲、バックアップ先

論点	どこを見るか	事故時に何が止まるか	社内で持つべきもの
CMS、アプリ	WordPress、独自アプリ	改ざん、脆弱化、更新停止が起きる	更新責任者、保守契約、手順
証明書	CDN、Web サーバー、ACME	ブラウザ警告、信頼低下が起きる	更新方式、通知先、依存 DNS

ドメイン管理でまず押さえてほしいのは、更新漏れを防ぐことだ。ICANN は、レジストラが有効期限の約1か月前と約1週間前に更新通知を送ること、expiration 後5日以内にも notice を送ることを求めている。通知は official contact information に入っている registrant email address へ送られる。つまり、ドメイン運用の基本は次の四つである。

1. auto-renew を有効にする
2. 支払い情報を最新にする
3. 登録連絡先を共有管理できるものにする
4. 次の期限を台帳側でも持つ

ここで注意したいのは、auto-renew があるから安心とは限らない点である。カード失効、登録メール未着、担当者退職で止まることは普通にある。さらに、失効後の Auto Renew Grace Period はレジストラ次第で1日から45日ありうるが、その間も Web やメールが止まることもある。したがって、救済策として考えすぎない方がよい。

DNS 側では、少なくとも主要レコードの意味を説明できる状態にしておきたい。

レコード	主な用途	触ると何に影響するか
A	IPv4の接続先を示す	Webやアプリの接続先が変わる
AAAA	IPv6の接続先を示す	IPv6経由の接続先が変わる

レコード	主な用途	触ると何に影響するか
CNAME	別名参照を示す	CDN、SaaS 接続先、検証先が変わる
MX	メール配送先を示す	メール受信が止まることがある
TXT	SPF、DKIM、検証文字列など	メール認証、ドメイン検証が壊れる
NS	どのネームサーバーが見つかるかを示す	参照先そのものが変わる
DS	DNSSEC で親ゾーンに置く情報	DNSSEC 利用時の名前解決に影響する

細かな設計を全部覚える必要はない。だが、どのレコードがどの用途かわからないまま触ると、Web だけでなくメールや認証まで巻き込む。第14章では、DNS を専門家の黒箱にしないことが重要である。

## DNS 切替は、事前準備と待ち時間がある変更作業である

DNS 変更でありがちなのは、「レコードを書き換えたのにすぐ変わらない」「戻したのに一部だけおかしい」という混乱である。これは DNS の仕組みというより、事前準備不足で起きることが多い。

Cloudflare の資料でも、TTL は DNS resolver がどれだけ長くキャッシュするかを制御する項目だと整理されている。AWS Route 53 の移行手順では、現在設定の取得、TTL の事前調整、古い TTL の期限切れ待ち、NS 切替、監視、必要なら切り戻し、という順番が明示されている。つまり、DNS 切替はその場の設定変更ではなく、変更管理そのものである。

製品名や管理画面は変わっても、ここで変わりにくい原則は同じである。現行設定を控える TTL を事前に調整する 確認項目を持つ 切り戻し先を決める の四つは、どの DNS サービスでも外しにくい。

実務では、DNS 変更前に次を確認したい。

見る点	先にやること	詰まりやすい点
現行設定	現在のレコード一覧や zone file を控える	元設定が分からず戻せない
変更対象	どのレコードを何に変えるか決める	関係ない MX、TXT まで巻き込む
TTL	事前に短くし、古い TTL が抜ける時間を待つ	切替直前に短くしてもすぐ効かない
変更日時	業務影響の少ない時間帯を選ぶ	問い合わせが集中する時間にぶつける
確認項目	Web 表示、証明書、フォーム、メールを確認する	Web だけ見てメール障害を見落とす
切り戻し	前の NS、前のレコード控えを残す	戻し先を持たず片道変更になる
DNSSEC	利用中か確認し、必要なら DS レコードも扱う	DNS だけ移せば終わると思いつい込む

ここで特に重要なのは、TTL を事前に下げるという発想である。切替直前に TTL を短くしても、すでに古い TTL でキャッシュされている利用者にはすぐ効かない。だから、早めに準備し、古い TTL が抜ける時間を見込む必要がある。

また、DNSSEC を使っている場合は、DS レコードの扱いも別論点になる。すべての会社で必須ではないが、使っているなら「DNS を移すだけ」で済まない。高度な設定ほど、移行手順を軽く見ない方がよい。

## Web サイトと公開アプリは、公開後の運用責任が本体である

Web サイトや公開アプリは、作る時にだけ注目されやすい。だが、実際に危ういのは公開後である。古い特設サイト、更新されないフォーム、止まった採用ページ、誰も入らない管理画面が残ると、公開資産は一気に危うくなる。

ここで持ちたい視点は、公開物を案件でなく運用で見ることだ。たとえば WordPress のような公開 CMS を置くなら、見るべきものは次の表に落とし込める。

項目	最低限見ること	放置時のリスク
WordPress core	最新版へ更新できているか	古い版は security updates の対象外になる
plugin	使用中 plugin の更新日と保守状態	脆弱 plugin が侵入口になる
theme	更新有無と導入元	改ざんや不整合が残る
backup	更新前に current backup を取れるか	更新失敗時に戻せない
不要 plugin、theme	使っていないものを残していないか	攻撃面だけが增える
導入元	trusted sources に絞れているか	出所不明の拡張が混入する
運用責任	host 側とサイト owner 側の役割が分かっているか	保守契約があるはずで止まる

WordPress 公式資料は、常に最新バージョンへ更新すること、古い版はセキュリティ更新対象外であること、plugin 更新前には current backup を持つべきことを明確に示している。さらに、host responsibility と website owner responsibility は別である。つまり、公開 CMS は作成時より、更新運用の方が本体である。

ここで外注と内製の別はあまり関係ない。外注先が作ったサイトでも、公開後の責任がどこにあるかを社内で説明できなければ危うい。ベンダー保守契約があるなら、その範囲、緊急連絡先、更新手順を持つ。ないなら、自社で更新できる体制か、そもそもその置き方が妥当かを見直す必要がある。

## 証明書更新と期限管理は、自動化を前提にする

証明書運用で一番避けたいのは、担当者の記憶に依存することである。証明書は切れた瞬間に利用者が不安になり、社内では障害扱いになる。だから、手動更新を平常運転にしない方がよい。

第14章では、証明書運用で最低限次を記録したい。

項目	記録すること	抜けると困ること
対象	どのドメイン、サブドメインに使うか	どこが期限切れか分からない
発行方式	CDN 側、ACME、サーバー手動など	どこで更新するか分からない
現在の有効期間	何日有効か、次回更新目安はいつか	更新周期を誤る
更新方式	自動か手動か	手順が個人依存になる
通知先	失敗時に誰へ届くか	失敗しても誰も気づかない
依存設定	DNS-01、HTTP-01、ロードバランサ、CDN など	更新時に別系統で詰まる
切り分け先	CDN、サーバー、委託先のどこを見るか	障害時に初動が遅れる

2026年3月21日時点で、Let's Encrypt の default certificates は 90 日有効で、90 日証明書は 60 日ごとの renewal を勧めている。一方で、Let's Encrypt は 2025年12月2日に、2026年5月13日から opt-in profile を 45 日へ、2027年2月10日に classic profile を 64 日へ、2028年2月16日に classic profile を 45 日へ移行予定だと公表している。つまり、短寿命化の方向はすでに明確であり、手動更新を前提にした運用はさらに現実的でなくなる。

もし年に一回、担当者が思い出したら更新するという運用が残っているなら、それは技術的負債である。すぐに置き換えられなくても、少なくとも期限監視と引き継ぎ可能な手順は必要だ。

証明書では、更新できることだけでなく、更新失敗に気づけることも重要である。自動更新ジョブがあっても、失敗通知が誰にも届かないなら意味が薄い。自動化と監視はセットで考えたい。

## 公開システムで注意すべき運用

公開系では、技術そのものより、管理の置き方で事故が起きやすい。特に注意したいのは次の点である。

- 個人アカウント依存を避ける
- 外注任せにしすぎない
- 変更記録を残す
- 不要サブドメインを残さない
- ドメイン移管を期限直前にやらない

個人依存の典型は、昔の担当者の個人メールがレジストラ連絡先になっている状態である。これは第9章、第13章と同じ問題で、公開基盤でも強く起きる。更新通知、請求、証明書失敗通知が個人へ飛ぶ設計は避けたい。

また、外注任せにしすぎるのも危うい。制作会社や保守会社が実務を担うこと自体はよい。だが、どのドメインがどこにあり、誰が最終責任者が を社内で説明できない状態は避けるべきである。

変更記録も大切である。DNS をいつ変えたか、証明書をどの方式へ切り替えたか、Web の公開先をどこへ移したか。これを残しておかないと、次の変更で毎回調べ直すことになる。最低限、変更日、変更者、変更理由、元設定、切り戻し先、変更後確認の結果は残したい。

## 小さな会社で現実的に回るやり方

すべての公開資産に高度な監視や二重化を入れられる会社ばかりではない。小さな会社では、まず見える化と例外削減の方が効く。

現実的には、次のくらいから始めるとよい。

- 公開資産台帳を作る
- 主要ドメインの auto-renew を確認する
- 更新通知先を共有メールへ寄せる
- 主要サイトの証明書方式と失敗通知を月1回確認する
- WordPress サイトの更新日を残す
- DNS 変更前に控えと切り戻し先を持つ

ここで意図的に避けたいのは、公開資産の種類を増やしすぎることである。古い LP、検証用サブドメイン、用途の薄い独立サイトが増えるほど、期限管理と更新責任が広がる。小さな会社ほど、公開先を絞る方が運用しやすい。

## 通常時の例と、崩れた時の例

通常時の例では、会社は公開資産台帳を持っている。そこには、本体ドメイン、採用サブドメイン、問い合わせフォーム、DNS ホスト、レジストラ、証明書方式、管理者、更新通知先が入っている。主要ドメインは auto-renew で、通知は共有アドレスに届く。DNS 変更時には事前にレコード控えを取り、TTL を調整

し、変更後確認と切り戻し先を準備する。WordPress サイトは core と plugin の更新日が残っており、更新前バックアップも確認している。結果として、公開基盤の変更や引き継ぎが人の記憶に依存しない。

崩れた時の例では、ドメインは昔の制作担当者が取得したままで、レジストラも DNS ホストも分からない。証明書は「たぶん自動」と言われるが、方式も通知先も不明である。古いキャンペーンサイトや使っていないサブドメインが残り、WordPress plugin 更新も止まっている。DNS 切替はその場で行われ、元設定の控えもなく、何か起きると戻せない。さらに、MX や TXT を巻き込んでメールや認証も崩す。結果として、公開資産は見えているのに管理されていない。

## 最低限ここまではやる

第14章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 公開資産台帳を作る
2. 主要ドメインの更新通知を共有管理する
3. DNS 変更前の控えと切り戻し先を持つ
4. 証明書更新方式と期限監視を確認する

この四つがあるだけで、公開資産の事故はかなり減る。公開資産の管理とは、Web サイトが見えることではない。どのドメインが、どの DNS で、どの証明書で、誰の責任で動いているかを会社として把握し続けることである。

## 今日、今週、後でやること

今日やることは、会社が使っている公開ドメインとサブドメインを洗い出すことである。本体サイトだけでなく、採用、問い合わせ、古い LP、検証用まで含める。

今週やることは、その中で重要なドメインを三つ選び、レジストラ、DNS ホスト、証明書更新方式、通知先を確認することである。空欄が多いなら、そこが公開基盤の弱点である。

後でやることは、WordPress を含む公開サイトの更新体制と、DNS 変更時の標準手順を一枚にまとめることである。ここまできると、公開資産は「昔から動いているもの」ではなく、会社が管理している資産へ変わる。

## 第15章

# データ保護、バックアップ、ログ、保持

---

バックアップは取ってある。そう聞くと少し安心する。ところが実際に「昨日の状態へ戻せますか」「この共有フォルダだけ復元できますか」「半年前の管理者操作ログを追えますか」と聞くと、急に答えがあいまいになることがある。NAS にコピーはある。SaaS に retention はある。監査ログもあるはずだ。だが、戻したことはない。保持期間は知らない。検索方法も決まっていない。これは珍しい状態ではない。

データ保護が難しいのは、似た言葉が多いからである。バックアップ、保持、ログ、アーカイブ、削除保留。どれも「残す」話に見える。だが、目的は同じではない。バックアップは復旧のためにある。保持は保存と削除のルールのためにある。ログは追跡と調査のためにある。この違いを曖昧にしたまま運用すると、平時は静かでも、有事に弱い。

第15章で扱うのは、ここを分けて考えるための型である。何を守るのか。どこまで戻せればよいのか。何のログをどれだけ残すのか。保存と削除をどう設計するのか。そして、本当に戻せることをどう確認するのか。この順番で見えていく。

## データ保護は、何を守るかを分けるところから始まる

最初に決めたいのは、何を守るかである。ストレージを丸ごと守る発想から入ると、重要度の違うものが全部同じ扱いになり、運用が重くなる。

まずは、次のように分けて考えると整理しやすい。

データ区分	例	止まる業務	どこまで巻き戻せる必要があるか	どれくらいで戻したいか
基幹業務データ	受発注、売上、顧客対応履歴	売上計上、受注処理、顧客対応	数時間前まで	当日中、できれば数時間以内
機微データ	人事、会計、契約、個人情報	給与、支払、監査、法務対応	前日まで、場合によりもっと細かく	当日中
共有ファイル	手順書、提案書、議事録、共有ドライブ	日常業務、引き継ぎ、営業資料参照	前日まで	半日から1日以内
構成情報	設定ファイル、スクリプト、IaC、証明書設定	再構築、設定復旧、切り戻し	直前の変更前まで	数時間から当日中
ログ	監査ログ、操作ログ、障害ログ	調査、監査、事故切り分け	戻す話ではなく保持が主	必要時にすぐ検索できること

CISA も、バックアップは何を守るかの inventory から始めるべきだとしている。つまり、バックアップの出発点は保存先ではなく、業務影響である。

ここで強く意識したいのは、復旧優先順位である。全部が同じ重さではない。たとえば、受発注データは当日中に戻したいが、過去の広報画像は翌週でもよいかもしれない。管理者操作ログは量が多くても、事故調査では重要かもしれない。だから、「何を残すか」と「どの順で戻すか」を一緒に決めた方がよい。

難しい言葉を使えば、**どこまで巻き戻せればよいか** と **どれくらいで戻したいか** である。だが、ひとり情シスの現場では、まず日本語で十分だ。昨日まで戻せればよいのか。数時間前まで必要か。半日止まってもよいのか。1 時間以内に戻したいのか。この粒度で考えるだけでも、設計はかなり変わる。

## バックアップは、復旧目標から設計する

バックアップを「毎日取っています」で終わらせると危うい。重要なのは、その頻度と方式で本当に必要な復旧ができるかどうかである。

たとえば、日次バックアップだけでは、午前中の入力を午後に見失うかもしれない。逆に、毎時間バックアップを取っていても、戻すのに丸一日かかるなら、実務では困るかもしれない。だから先に見たいのは、次の点である。

- どこまで前の状態へ戻せればよいか
- どれくらいの時間で戻したいか
- ファイル単位で戻したいのか
- システム単位で戻したいのか

NIST の contingency planning は、復旧を計画、手順、技術的措置を含む coordinated strategy として扱っている。これは実務的である。バックアップも同じで、媒体やソフトだけでなく、優先順位、手順、代替手段まで含めて初めて意味を持つ。

最初の設計表は、次の形で十分である。

対象	頻度	世代	保管先	分離	暗号化	戻し方
受発注 DB	1時間ごと	7日分+月次	本番と別ストレージ	別アカウント、別ネットワーク	する	DB単位で復元
共有ファイル	日次	30日分	NASと別クラウド	利用者から直接見えない	する	フォルダ、ファイル単位で復元

対象	頻度	世代	保管先	分離	暗号化	戻し方
設定ファイル、スクリプト	変更のたび	履歴保持	Git、保守領域	本番権限と分ける	必要ならする	変更単位で戻す
重要端末、サーバー土台	月次見直し	直近数世代	golden image 保管先	通常運用と分離	する	再展開して再構築

ここでよくある誤解は、「全部守ろう」とすることである。実際には、重要度に応じて差をつけた方が回る。たとえば次のような分け方がある。

- 基幹データは高頻度で取り、優先復旧対象にする
- 共有ファイルは日次と世代管理を基本にする
- 構成情報やスクリプトは更新のたびに残す
- 一時ファイルや再生成可能なデータは軽く扱う

重要なのは、守る範囲を絞ることではなく、重さを分けることである。

## 世代管理、別媒体、別ネットワーク保管を組み合わせる

バックアップで一番避けたいのは、同じ事故で本体とバックアップが一緒にやられることである。ランサムウェア、誤削除、設定ミス、物理故障。原因は違っても、一か所依存は弱い。

CISA は、offline かつ encrypted backups を維持し、regular basis で availability と integrity を test することを勧めている。さらに golden images の維持も挙げている。つまり、データ保護では次の組み合わせが重要になる。

考え方	何をするか	ないと何が起きるか
複数世代	最新だけでなく、一つ前、二つ前も残す	改ざんや誤削除が最新世代に反映される

考え方	何をするか	ないと何が起きるか
別媒体	本体と別ストレージへ置く	同じ故障で本体も控えも失う
別ネットワーク、別権限	本番と別アカウント、別管理面へ置く	侵害時にまとめて削除される
オフライン	常時接続しない控えを持つ	ランサムウェアで同時暗号化される
暗号化	保管中の backup を暗号化する	持ち出しや誤送付で漏えいする
golden image	再構築用の土台を別に持つ	データが戻っても再展開に時間がかかる

ここで言う世代管理とは、最新一個だけを持つことではない。最新版が壊れても、一つ前、二つ前へ戻れるようにする考え方である。誤削除や改ざんは、気づいた時にはすでに最新バックアップにも反映されていることがあるからだ。

また、分離も重要である。同じ認証情報、同じネットワーク、同じ管理画面で本体とバックアップがつながっていると、侵害時にまとめて侵害される。小さな会社でも、少なくとも「バックアップ先は普段の利用者から直接見えない」「削除や暗号化にすぐ届かない」状態は作りたい。

CISA は 3-2-1 rule も示している。これは、三つのコピー、二種類の媒体、一つは offsite という考え方である。第15章では、これを厳密な宗教にする必要はないが、**一か所依存を避ける** ための基準として持っておきたい。

## ファイルサーバーと SaaS は同じ守り方をしない

ここはひとり情シスが迷いやすい点である。ファイルサーバーの感覚で SaaS を見たり、逆に SaaS の retention をそのままバックアップだと思ってしまったりする。

実務では、次のように分けると整理しやすい。

対象	native の保護機能	強い点	限界、注意点
ファイルサーバー、NAS	snapshot、世代管理、複製	ファイル、フォルダ単位で戻しやすい	同一筐体、同一権限だと同時被害を受ける
Microsoft 365 の OneDrive、SharePoint	recycle bin、Files Restore、audit、retention	利用者に近い粒度で戻せる機能がある	restore 期間や粒度に制約がある
Google Workspace の Drive	trash、管理者復元、Vault、audit	削除直後や保持対象の検索、export ができる	管理者復元の期間と粒度に制約がある

Microsoft 365 側で押さえない具体知識は、次である。

- OneDrive 全体復元は last 30 days の範囲で使える
- Recycle Bin から permanently deleted された files は戻らない
- restore point より後に作られた files は Recycle Bin へ送られる
- SharePoint の deleted items は 93 days retained である
- SharePoint Online の backend backup は actual deletion 後 14 additional days あるが、site collection 単位であり、specific files や lists には使えない

Google Workspace 側で押さえない具体知識は、次である。

- Drive の管理者復元は last 25 days の deleted data が対象である
- individual files や folders ではなく、selected date range 全体を restore する
- one restoration project at a time の制約がある
- user's trash に残っている data や permanently deleted more than 25 days の data は戻せない

- Vault は deleted data を searchable and exportable に保てるが、サービス内へそのまま即時復元する機能とは別である

ここでの実務的な整理は次である。

1. ファイルサーバーは backup 方式を明示する
2. SaaS は native restore、recycle bin、retention、export、audit の範囲を確認する
3. その上で、足りない復旧要件があるなら別対策を考える

この「足りない復旧要件」という考え方が重要である。すべての SaaS に追加バックアップ製品が必要とは限らない。だが、期間、粒度、検索性、別コピーの有無が足りないなら、そこで初めて補強を考える方が実務的である。

## ログ取得、保持、検索の設計

ログは、残っているだけでは足りない。NIST SP 800-92 が示すように、ログ管理は generate、transmit、store、analyze、dispose まで含む。つまり、次の四つを決めないと、有事に使いにくい。

- 何を取るか
- どこへ集めるか
- どれだけ残すか
- 誰が見られるか

ここでログを大きく分けると、少なくとも次がある。

ログ種別	主目的	例	最低限決めること
監査ログ	だれが何をしたか追う	管理者操作、権限変更、共有設定変更	保持期間、検索手順、閲覧権限

ログ種別	主目的	例	最低限決めること
認証ログ	サインインや失敗の確認	login、MFA、blocked sign-in	どこで見るか、アラート条件
障害ログ	不具合切り分け	app error、job failure、API error	集約先、保持期間、担当者
セキュリティ関連ログ	侵害兆候や異常把握	malware、EDR、network events	保持、通知、エスカレーション

ベンダー標準の保持期間も見落としやすい。第15章では、少なくとも次は知っておきたい。

サービス	代表的なログ、機能	2026年3月21日時点の保持例	実務上の意味
Microsoft Purview Audit(Standard)	監査ログ	180 days	半年前の操作確認が限界になることがある
Microsoft Purview Audit(Premium)	Entra、Exchange、OneDrive、SharePoint の audit records	1 year	年単位の調査に向くが無期限ではない
Google Workspace reports、log events	管理、Drive、User など多くの logs	generally 6 months	後から確認しようと思うと消えていることがある
Google Email log search	メール配送調査	30 days	古い配送問題は追えない
Google Vault log events	Vault 操作	indefinite	法務、監査の追跡に使いやすい

つまり、「必要になったら後で見ればよい」は危うい。必要な調査期間が半年を超えるなら、標準保持のままで足りるかを先に見ておく方がよい。

ログ設計で特に大切なのは、検索導線である。検索画面がどこにあるか、どの権限で見られるか、CSV に出せるか、どこまで遡れるか。これが決まっていないと、ログが存在しても調査は遅れる。

## 保持は、保存と削除をセットで考える

保持設定は安全そうに見えるが、同時に削除設定でもある。ここを軽く見ると危うい。

Microsoft Learn は、retain-only、delete-only、retain and then delete の三つを明確に分けている。さらに retention always takes precedence over permanent deletion と longest retention period wins の原則を示している。Google Vault でも、indefinite retention は deleted data を searchable and exportable に保てる一方で、retention period を持つ rule を submit すると、期間を超えた data が purge されうる。

保持の基本形は、次のように整理しやすい。

型	何が起きるか	向く場面	注意点
retain-only	一定期間は残す	契約、法務、監査対応で消せないもの	残り続けるので検索対象が増える
delete-only	一定期間後に消す	古い通知、不要データの整理	消してよい範囲を誤ると事故になる
retain and then delete	一定期間残し、その後消す	一定保存と整理を両立したい時	期間設定を誤ると古い data が想定外に消える

保持ルールを変える前には、少なくとも次を確認したい。

- 何に適用するか
- いま残っている古い data に何が起きるか

- 法務や監査の例外があるか
- まず少数対象で試せるか
- 削除後に戻せるのか、戻せないのか

Google Vault は、test new rules on a small set of accounts before applying them to the entire organization と勧めている。これはとても実務的である。保持設定は、全社へ一気にかける前に小さく試した方がよい。

## 復旧を前提にした運用確認

復元テストをしていないバックアップは、まだ完成していない。これは第15章の中心メッセージの一つである。

CISA も、バックアップは regular basis で availability と integrity を test すべきとしている。理由は単純だ。取れていると思っていたが失敗していた、復元に必要な手順が欠けていた、権限が足りず戻せなかった、ということが普通に起きるからである。

復元テストは大掛かりでなくてもよい。小さな会社なら、まずは次のくらいから始められる。

試すこと	頻度	成功条件	詰まりやすい点
共有フォルダの sample file 復元	月次	指定ファイルを元場所か別場所へ戻せる	権限不足、世代不足
主要 SaaS の export 実行	月次か隔月	必要データを検索し export できる	検索場所不明、対象期間切れ
管理者操作ログの検索	月次	特定操作を時刻つきで特定できる	権限不足、保持切れ
端末、サーバー土台の再展開確認	四半期か半期	golden image から再構築手順を追える	イメージ古い、手順欠落

試すこと	頻度	成功条件	詰まりやすい点
保持ルール影響確認	ルール変更時	何が残り、何が消えるか説明できる	本番へ先に適用してしまう

ここで大切なのは、本番障害が起きてから初めて触らないことである。戻す練習を一度でもしておく、必要な権限、時間、詰まりどころが見える。逆に、練習していないと、有事に「残っているか」だけでなく「どう戻すか」から調べることになる。

また、復旧確認は年一回の儀式にしない方がよい。すべてを毎月試す必要はないが、対象を回しながら小さく継続した方が運用しやすい。

## 小さな会社で現実的に回るやり方

すべてのログを長期保管し、すべてのデータを多重化し、頻繁に大規模復旧訓練をするのは現実的ではない会社も多い。小さな会社では、まず重要なものに絞る方が効く。

現実的には、次のくらいから始めるとよい。

- 重要データ一覧を作る
- バックアップ対象、頻度、世代、保管先、戻し方を書く
- オフラインまたは分離保管を一つ持つ
- 主要 SaaS の native restore とログ保持期間を確認する
- 保持ルールの一覧を持つ
- 月に一つだけ復元や検索を試す

ここで意図的に避けたいのは、何でも残すことと、何も試さないことである。前者は管理不能になり、後者は有事に役立たない。小さな会社ほど、「重要なものを、必要な期間だけ、戻せる形で持つ」という単純な方針の方が強い。

## 通常時の例と、崩れた時の例

通常時の例では、会社が重要データ一覧を持っている。受発注データ、人事データ、共有ファイル、主要 SaaS、管理者監査ログが載っており、それぞれに復旧優先順位、バックアップ方式、保持期間、担当者が書かれている。ファイルサーバーは世代付きで分離保管され、主要 SaaS は native restore と retention、export 可否が確認済みで、監査ログ保持期間も把握している。月に一度は何か一つ復元や検索を試し、手順を更新している。結果として、「何をどこまで守れているか」を説明できる。

崩れた時の例では、バックアップはあると言われるが、対象も世代も保管先も説明できない。SaaS は retention があるから大丈夫だと思っているが、どのデータがどれくらい残るか、何日まで戻せるかは知らない。監査ログも「たぶん残る」と考えていたが、必要になった時には保持期間を過ぎている。保持ルールを変えたら古いメールや Drive data が消え、復元手順も分からない。問題は機能不足ではなく、目的を分けずに運用していたことである。

## 最低限ここまではやる

第15章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 重要データと復旧優先順位を決める
2. バックアップ対象、頻度、世代、保管先、戻し方を書く
3. 主要ログの保持期間を把握する

#### 4. 小さくても復元テストを実施する

この四つがあるだけで、データ保護はかなり実務的になる。データ保護とは、たくさん保存することではない。何をどれだけ残し、どこまで追え、どの順で戻せるかを会社として決めておくことである。

### 今日、今週、後でやること

今日やることは、守るべきデータを五つだけ書き出し、それぞれに「止まると何が困るか」と「いつまで戻せればよいか」を一言添えることである。これだけで、優先順位の感覚が出てくる。

今週やることは、その中から一つを選び、実際のバックアップ対象、保持期間、保管先、復元方法を書き出すことである。同時に、主要 SaaS の native restore 期間と監査ログ保持期間も一つ確認したい。

後でやることは、月次の小さな復元テストと、保持ルール一覧の見直しを定例にすることである。ここまでできると、バックアップは「あるらしいもの」から「説明できる運用」へ変わる。

## 第16章

## ヘルプデスク、依頼管理、ナレッジ整備

---

問い合わせには毎日答えている。パスワード再設定も、ソフト導入依頼も、PC不調も、その場その場では処理している。だが一か月後を振り返ると、同じ質問にまた答えている。ソフト導入依頼では毎回同じことを聞き返している。FAQは一応あるが古く、誰も見ない。チケットは件数を数えるだけで、何が繰り返されているかは見えていない。こういう状態は珍しくない。

ヘルプデスク運用が重くなるのは、依頼が多いからだけではない。もっと本質的には、依頼が標準化されず、自己解決へ寄せられず、対応の中で知識が残らないからである。すると、処理はしているのに、来月も同じことをする。

第2章では、窓口、記録、台帳、手順、優先順位という最小の土台を整えた。第16章では、その土台の上で、日々の依頼対応をどう軽くしていくかを見る。論点は、チケット化、依頼種別、申請フォーム、self-service、ナレッジである。

### 問い合わせは、必ず一つの記録へ乗せる

第16章で最初に確認したいのは、すべての依頼が、最終的に一つの記録へ乗る状態になっているかである。ここで大切なのは、最初から完璧なITSMツールを入れることではない。共有窓口でも、簡易チケットでも、スプレッドシートでもよい。重要なのは、口頭、チャット、メールで来た依頼が、そのまま流れないことである。

AtlassianもServiceNowも、service deskの価値を、単なる受付でなく、標準化、tracking、improvementの基盤として説明している。つまり、チケット化の目的は監視ではない。後から追えることと、後で直せることにある。

最低限、次は一つの場所で見たい。

項目	何を書くか	抜けると何が起きるか
受付日時	いつ来た依頼か	滞留や優先順位が見えない
依頼者	だれの依頼か	確認先が分からない
依頼種別	障害、依頼、変更など	同じ箱で混線する
優先度	急ぐか、影響が大きいか	先に何を見るべきか分からない
状態	新規、対応中、承認待ち、完了など	何が止まっているか見えない
担当	だれが次に動くか	放置と重複対応が起きる
次のアクション	何を待っているか、何をやるか	進め方が人の記憶依存になる
完了日	いつ終わったか	処理時間や滞留を振り返れない

第2章では入口を一つにすることを強調した。第16章で一段進めたいのは、**受けた依頼が、例外なく記録へ乗る** という運用である。チャットで直接頼まれても、その場で自分が代筆してよい。とにかく、後から「何が残っているか」が見えることが先である。

一次回答も、短い型を持つと楽になる。

一次回答で返すこと	目的
受付したこと	依頼者を待たせない
追加で必要な情報	聞き返しを一回にまとめる
次にやること	対応の見通しを出す
次回連絡の目安	放置感を減らす

## 依頼種別と優先度を分ける

依頼管理が重くなる大きな理由の一つは、障害と依頼と変更が同じ流れで処理されることである。これでは、緊急障害に引っ張られて定型依頼が溜まり、定型依頼に時間を取られて重要障害の初動が遅れる。

Atlassian は、service request を separate workstream として扱うことを勧めている。service request とは、パスワード再設定、ソフトウェアライセンス、アクセス付与、新しい端末の手配、情報照会のような、繰り返し発生しやすい依頼である。これらは incident と違い、標準化しやすい。

少なくとも、次は分けたい。

種別	代表例	先に見ること	向く処理
障害	メール不達、PCが起動しない、VPN障害	影響範囲、復旧優先度、暫定対応	初動優先、切り分け優先
依頼	パスワード再設定、ソフト導入、権限追加	必要情報が揃っているか、承認要否	標準手順、定型処理
変更	設定変更、ネットワーク変更、公開設定変更	影響、承認、切り戻し	変更管理、承認付き処理

この三つを分けるだけでも、見方が変わる。障害は復旧を急ぐ。依頼は必要情報を揃えて標準処理へ寄せる。変更は影響確認と承認が重要になる。全部を一つの箱に入れると、どの順で見るべきかが曖昧になる。

queue の考え方もここで効く。Atlassian は、queue を specific criteria で filtered した focused view として説明している。つまり、queue の本体は **一覧を増やすこと** ではなく **今見るべきものだけを見ること** にある。

最初の queue は、次のくらいで十分である。

queue 名	入れるもの	目的
新着	新しく来た依頼全部	まず受ける
障害	incident と判断したもの	復旧を優先する
情報不足	追加情報待ちの依頼	放置を見える化する
承認待ち	manager 承認や予算待ち	情シス側で止まっていないものを分ける
長期滞留	数日以上止まっているもの	詰まりを見つける

## 申請フローは、確認事項を前倒しする仕組みである

依頼フォームや申請フローを嫌がられることがある。「手間が増える」「気軽に頼めない」という反応である。だが実際には逆で、フォームの価値は、担当者が後で聞き返す手間を前に寄せることにある。

Atlassian の request forms は、conditional sections や custom layouts を持つ。これは見栄えの話ではない。依頼内容に応じて、必要な質問だけを出せるという意味である。たとえば、代表的な request type では、次の項目を最初から取りたい。

request type	最初に取り情報	先に取り理由
ソフト導入依頼	利用目的、利用者、利用端末、予算、承認者、希望時期	毎回の聞き返しを減らす
アクセス権依頼	対象システム、必要権限、業務理由、承認者、必要終了日	権限過剰や期限なしを防ぐ

request type	最初に取る情報	先にとる理由
端末手配依頼	新規か交換か、利用者、勤務地、必要時期、標準外要件	手配漏れと例外対応を減らす
備品依頼	何が必要か、用途、数量、承認者	不足情報の往復を減らす

ここが抜けると、ひとり情シスは毎回同じ質問を返すことになる。結果として、依頼者も待ち、担当者も疲れる。

申請フローで重要なのは、すべてを厳しくすることではない。むしろ、低リスクの定型依頼は簡単にし、高リスクや例外だけ重くする方がよい。たとえば、標準ソフトの追加インストールは簡単な申請で済ませ、未承認 SaaS や高権限付与だけ別承認にする。この分け方がないと、全部が面倒になり、結局裏口依頼が増える。

ここで一つ注意したいのは、受付チャンネルと必須入力分けて考えるべきことだ。Atlassian の公式資料では、forms は portal では使えるが、2026年3月21日時点で email、chat、virtual agent、widget では使えない。つまり、メールやチャットでも依頼は受けられるが、必要情報を強制しにくい。だから、どのチャンネルから来ても最終的に記録へ乗せ、不足情報はあとで補記する運用が必要である。

フォームは利用者を管理する道具ではない。確認漏れを減らし、処理を速くする道具である。

## self-service へ寄せられるものを見極める

ヘルプデスク運用を軽くするうえで、self-service は重要である。ただし、何でも自己解決に寄せればよいわけではない。向いているものと向いていないものを分けた方がよい。

ServiceNow は、common issues を self-service や automation で処理し、IT が complex requests に集中できるようにする考え方を示している。ここで代表例になるのが、パスワード再設定である。Microsoft Entra の SSPR (Self-Service Password Reset) も、None、Selected、All users から選べ、selected group で試してから広げるやり方を案内している。つまり、self-service は全社一斉でなく、限定導入から始めてよい。

self-service に向きやすいものと、人が見た方がよいものは、次のように分けやすい。

向き	代表例	理由
self-service に向く	パスワード再設定、Wi-Fi や VPN の基本案内、申請状況確認、標準ソフト申請方法	繰り返し多く、標準化しやすい
人が見た方がよい	高権限付与、例外端末、機密データ関連、状況判断が必要な障害	個別判断と確認が重い

self-service を導入する時に避けたいのは、「記事を読んでから来てください」で終わることである。記事が古く、見つけにくく、手順も複雑なら、利用者は余計に困る。self-service は、見つけやすく、短く、今の環境で使えることが前提である。

Microsoft Entra の SSPR で実務的に参考になるのは、次の点である。

- selected group で試してから広げられる
- notification を users や admins へ送れる
- custom helpdesk email or URL を設定できる

この最後の点が重要である。self-service は、helpdesk を不要にする仕組みではない。困った時に、どこへ戻ればよいかを見せる仕組みでもある。

## ナレッジは、対応の副産物として育てる

ナレッジが増えない最大の理由は、別仕事として積み上がるからである。問い合わせに答えた後で、落ち着いたら記事を書く。多くの場合、その「後」は来ない。

Atlassian が紹介する KCS (Knowledge Centered Service) は、knowledge を capture、structure、reuse、improve していく loop を示している。ここで重要なのは、articles are created and updated as a by-product of the problem-solving process としている点である。これはひとり情シスに特に向いている。なぜなら、対応しながら少しずつ残す方が、後でまとめて書くより現実的だからだ。

実務では、次の流れにすると続きやすい。

動き	何をするか	ねらい
search	まず既存記事を探す	同じ説明を一から書かない
capture	なければ短く残す	次の一件を楽にする
structure	テンプレートで整える	読みやすさと一貫性を出す
reuse	次回から記事を案内する	回答を再利用する
improve	足りない所だけ直す	古い記事を使える状態に保つ

ここで重要なのは、最初から立派な記事にしないことである。短い FAQ でもよい。見出しと手順だけでもよい。大切なのは、次の一件を楽にできることである。

最小の記事テンプレートは、次のくらいで十分である。

項目	何を書くか
タイトル	利用者が検索しそうな言い方

項目	何を書くか
症状、依頼内容	何に困った時の記事か
手順	3から7手順程度
注意点	よく詰まる所、対象外
更新日	古い記事を見分けるため

ナレッジ化に向いている題材は、同じ説明を二回以上したものだ。たとえば、VPN 接続手順、社内プリンタの使い方、標準ソフトの申請方法、共有フォルダ申請時の注意点。このあたりは、一本記事があるだけでかなり軽くなる。

## チケットは、件数より傾向を見るために使う

チケット管理を件数報告だけで終わらせると、改善に結びつきにくい。第16章で見たいたのは、どの依頼が繰り返されているか、どこで滞留しているか、どの申請で差し戻しが多いかである。

ServiceNow も、tracking provides valuable data that can be analyzed to identify patterns、trends、areas for improvement としている。つまり、チケットの価値は、何件処理したかより、何を減らせるかを見ることにある。

月に一度でもよいので、次を眺めたい。

見る項目	何が分かるか	次の手
繰り返し多い依頼	self-service や FAQ 候補	記事化、self-service 化
差し戻しが多い申請	request form の不足	入力項目の見直し
情報不足で止まる依頼	一次回答やフォームが弱い	必須項目の追加
長期滞留案件	承認待ち、担当不明、優先度ミス	queue や承認経路の見直し

見る項目	何が分かるか	次の手
記事が使われない依頼	検索しにくい、古い、見つからない	タイトル、置き場、内容の修正

ここを見ると、改善点が見える。パスワード再設定が多いなら self-service を考える。ソフト導入依頼の差し戻しが多いならフォームを直す。同じ PC 設定の質問が多いなら初期案内を見直す。こうして、依頼対応が改善活動へつながる。

## 小さな会社で現実的に回るやり方

すべての会社が、本格的な service catalog や高度な virtual agent を入れられるわけではない。小さな会社では、最小構成でも十分効果がある。

現実的には、次のくらいから始めるとよい。

- 共有窓口を一つにする
- 依頼をチケットまたは一覧表へ乗せる
- request type を三つか四つに絞る
- よくある依頼のフォームを一つ作る
- FAQ を五本だけ作る
- パスワード再設定を self-service 候補にする

ここで意図的に避けたいのは、依頼分類を細かくしすぎることと、ナレッジを完璧にしようとするることである。分類が増えすぎると誰も正しく選べない。記事を完璧にしようすると誰も書かない。小さな会社ほど、単純な型の方が強い。

## 通常時の例と、崩れた時の例

通常時の例では、IT 窓口へ入った依頼は必ず一つの記録へ乗る。依頼種別は、障害、依頼、変更に分かれている。標準ソフト導入や権限追加は request form で必要情報を先に集める。portal 以外のチャンネルから来た依頼も、記録へ転記して不足情報を補う。パスワード再設定は selected group から self-service を始めており、FAQ と合わせて単純問い合わせを減らしている。一次回答で使った説明は短いナレッジ記事へ残し、次回から記事を案内する。月に一度、繰り返し多い依頼を見て、フォームや FAQ を直す。結果として、問い合わせ件数はあっても、同じ説明を何度もゼロから繰り返さなくなる。

崩れた時の例では、依頼はチャット、口頭、メールに散っている。障害と依頼が同じ箱で扱われ、急ぎではない定型依頼が割り込む。ソフト導入依頼では毎回「誰が使うのか」「何のためか」を聞き返す。パスワード再設定は全部人手で受け、FAQ は古く、誰も見ない。チケットは件数だけ数えて終わり、繰り返し傾向は見ていない。結果として、毎日忙しいのに、来月も同じことで忙しい。

## 最低限ここまではやる

第16章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 依頼が必ず記録へ乗る状態を作る
2. service request と incident を分ける
3. よくある依頼の request form を一つ作る
4. 繰り返し問い合わせを一件ナレッジ化する

この四つがあるだけで、ヘルプデスク運用はかなり軽くなる。ヘルプデスク運用とは、来た依頼に答えることではない。依頼を標準化し、単純なものを減らし、残った重要案件へ時間を使える状態を作ることである。

## 今日、今週、後でやること

今日やることは、直近一か月で多かった依頼を五つ書き出すことである。そこに、同じ説明を繰り返したものがあれば、ナレッジ候補である。

今週やることは、その中から一つを選び、request form が FAQ のどちらかへ落とすことである。たとえばソフト導入依頼ならフォーム、VPN 接続案内なら FAQ が向く。

後でやることは、月次で recurring requests を見直し、self-service 候補と記事更新候補を一つずつ決めることである。ここまでできると、ヘルプデスクは受け身の仕事から、軽くなる仕事へ変わる。

第IV部

# セキュリティ、法務、監査

事故を減らし、説明できる運用にする。

第17章～第20章

## 第17章

# セキュリティ基礎対策

セキュリティ事故というと、高度な標的型攻撃や巧妙なゼロデイを思い浮かべやすい。だが、ひとり情シスや中小規模組織の現場では、もっと基本的な抜けから事故が始まることが多い。管理者アカウントに MFA がない。古い認証方式が残っている。怪しいメールが来ても報告先が曖昧で、そのまま開かれる。社内 Wi-Fi とゲスト Wi-Fi が同じで、複合機や私物端末も同じ場所にいる。送金依頼や口座変更依頼を、メール一本で確定してしまう。こうしたことは、珍しい失敗ではない。

第17章で扱うのは、ここを先に整えるための基礎対策である。高度な製品比較ではない。小さな会社でも優先しやすく、効果が出やすい最低線を扱う。中心になるのは、認証、メール、端末、ネットワーク、そして業務手順である。

## セキュリティ対策の全体像は、入口、成立、拡大の三段で考える

セキュリティ対策を考える時に有効なのは、個別製品名から入らないことである。まずは、どこから入られるか、入られた時に成功しやすいか、成功した後にどこまで広がるかで整理した方が分かりやすい。

第17章では、基礎対策を次の三段で考えたい。

段	何を減らすか	代表例	まず効く対策
侵入口を減らす	入られるきっかけ	legacy authentication、怪しいメール、未管理端末	MFA、旧式認証停止、anti-phishing、端末管理

段	何を減らすか	代表例	まず効く対策
攻撃を成立しにくくする	侵入後の成功率	password only、管理者常用、保護なし 端末	標準ユーザー運用、 built-in AV、 firewall、最小権限
被害面積を広げにくくする	侵入後の横展開、実害	guest と社内未分離、送金メール一本承認	guest Wi-Fi 分離、管理者分離、二経路確認

この順番が重要である。ログの保持や復旧は後続章で詳しく扱うが、第17章ではまず、**そもそも起こりにくくすること**へ集中したい。

## マルウェア、フィッシング、BEC を分けて考える

セキュリティ対策がばやけるのは、脅威を全部ひとまとめにしてしまうからである。少なくとも、マルウェア、フィッシング、BEC は分けて考えた方がよい。

脅威	主な入口	狙われるもの	効く対策
マルウェア	添付ファイル、悪意あるリンク、脆弱ソフト、USB	端末、サーバー、共有データ	端末保護、更新、実行制御、標準ユーザー
フィッシング	メール、SMS、チャット、偽ログイン画面	認証情報、承認操作	MFA、メール保護、利用者教育、報告導線
BEC	なりすましメール、乗っ取られた正規メール	送金、口座変更、請求先変更、機微情報	送信認証、受信保護、二経路確認、承認手順

マルウェアは、端末やサーバーで不正な動作をさせるものだ。添付ファイル、悪意あるリンク、脆弱なソフト、USB 媒体など入口は複数ある。ここでは、端末保護、更新、実行制御、権限管理が効く。

フィッシングは、利用者をだまして認証情報や承認操作を引き出す攻撃である。メール、SMS、チャット、音声通話の形を取ることもある。ここでは、メール保護、MFA、利用者教育、報告導線が効く。

BEC は business email compromise の略で、メールアカウントの乗っ取りやなりすましを通じて、送金、口座変更、請求先変更、機微情報送付をだまし取る詐欺である。FBI IC3 の 2024年9月11日の PSA では、2013年10月から 2023年12月までの exposed dollar loss が 550 億ドル超とされている。ここでは、メール保護だけでなく、業務手順が重要になる。

この三つは似て見えるが、対策の置き場所が違う。だから、**セキュリティ教育**をやるだけでは足りないし、逆に **製品を入れる** だけでも足りない。

## 端末の基礎防御を切らさない

端末は最も身近な入口である。ここでの基礎対策は、豪華な機能を盛ることではなく、最低線を切らさないことだ。

最低限、次はそろえたい。

項目	最低線	抜けると何が起きるか
OSと主要ソフト	サポート中のものだけを使う	既知の弱点が放置される
ウイルス対策	built-in または管理された保護を有効に保つ	実行時検知が弱くなる
ローカル firewall	切らない	不要な通信が通りやすくなる
標準ユーザー運用	日常利用で管理者権限を常用しない	侵害時の被害が大きくなる
画面ロック	短時間で自動ロック	離席時にそのまま使われる

項目	最低線	抜けると何が起きるか
管理対象	管理されていない端末を増やさない	守れていない端末が入口になる

Microsoft は、Microsoft Defender Antivirus を Windows に built in の保護機能として位置づけており、disabled や uninstalled は一般に推奨していない。ここから見ても、EDR までに入れられない組織であっても、少なくとも組み込みのウイルス対策を無効化せず、更新され、状態確認できることが最低線になる。

ひとり情シスの現場では、**専用製品がないから十分に守れない** と考えやすい。だが実際には、何もないより、組み込み保護を正常に保ち、標準ユーザーで使い、怪しい挙動を報告できる方がはるかに強い。

ここで避けたいのは、例外端末の放置である。古い業務ソフトのために更新が止まった PC、個人判断でウイルス対策を切った PC、ローカル管理者のまま使われている PC。こうした端末は、全体の防御線を静かに崩す。詳細なパッチ運用は第18章で扱うが、第17章の段階でも、**守れていない端末を放置しない** という姿勢は必要である。

## メールと認証の基礎防御を整える

最も効果が高い基礎対策の一つが、認証の近代化である。古い認証経路や password only の抜け道を残したまま MFA だけ議論しても、守りは弱い。

Microsoft Entra の security defaults は、全ユーザーの MFA 登録、管理者の MFA、legacy authentication protocols の block を基礎線としている。2024年7月29日以降は、MFA 登録の 14 日猶予も廃止されている。Google Workspace でも、2025年5月1日から username と password だけで接続する less secure apps をサポートしなくなった。両者に共通するのは、**古い抜け道を残さない** という考え方である。

認証とメールで、最低限見たい項目は次である。

項目	最低限確認すること	ねらい
管理者 MFA	まず管理者へ強制する	重要アカウント侵害を減らす
一般利用者 MFA	段階導入計画を持つ	password only を減らす
legacy authentication	使っている経路を洗い出し止める	MFA回避経路を消す
less secure apps	Google Workspace 側の古い接続を残さない	username/password only を止める
anti-phishing	既定保護を確認する	怪しいメールを通しにくくする
anti-spoofing	spooftelligenceなどを有効に保つ	なりすましを見抜きやすくする
SPF、DKIM、DMARC	自社ドメインの送信認証状態を見る	自社なりすましを減らす

実務では、次の順で進めるとよい。

1. まず管理者アカウントに MFA を強制する
2. 次に一般利用者へ広げる
3. legacy authentication や less secure apps を止める
4. 例外が必要な機器や古いクライアントを洗い出す

ここで大切なのは、MFA を入れることと、MFA を避けられる旧式認証を止めることを一体で見ることである。片方だけだと抜ける。

メール保護も同じで、教育だけでは不十分である。Microsoft 365 では、all cloud mailboxes に basic anti-phishing features があり、spooftelligence や unauthenticated sender indicators のような保護がある。こうした既定の保護設定を確認せず、利用者の注意力だけに依存するのは危うい。

また、自社が受けるメールだけでなく、自社ドメインから出るメールのなりすまし対策も重要である。Google は DKIM に加えて SPF と DMARC の設定を推奨しており、DMARC はいきなり厳格にせず、まず monitoring してから段階的に強める流れを案内している。ここから分かるのは、送信ドメイン認証も基礎対策の一部だということだ。

## ネットワークと境界は、社内だから安全と思わない

ゼロトラストという言葉は難しく見えるが、第17章の段階では単純に考えてよい。NIST SP 800-207 が言う通り、network location に基づく implicit trust を置かないことが本質である。つまり、社内 LAN にいるだけで安全だと思わないことである。

小さな会社でまず効くのは、次のような対策だ。

項目	最低線	抜けると何が起きるか
guest Wi-Fi	社内用と分ける	来客、私物端末、業務機器が混ざる
機器配置	複合機、カメラ、会議室機器を何でも同じ場所に置かない	弱い機器が横移動の足場になる
初期パスワード	ルーター、AP、UTM で残さない	初期値のまま侵入される
管理画面公開	インターネットへさらさない	外から直接狙われる
VPN、管理画面 MFA	管理系経路へ MFA をかける	管理権限を奪われやすい

第11章でネットワーク構成は詳しく扱ったが、第17章での要点は、**境界を信用しすぎない** ことである。社内だから自由に見える、社内だから管理画面を開けてよい、社内だから高権限で使ってよい。こうした前提を少しずつ崩すだけで、防御はかなり安定する。

## 最小権限と業務手順で被害を小さくする

基礎対策というと、設定の話に寄りがちだ。だが、実害を小さくするには、設定だけでなく業務手順が必要である。特に BEC はその典型だ。

FBI IC3 は、口座情報変更の依頼に対して secondary channels や 2FA で verify することを prevention tips に入れている。これはとても実務的である。送金依頼や口座変更依頼がメールで来ても、そのメールだけで確定しない。既知の電話番号へかけ直す。別チャンネルで確認する。承認者を一段増やす。このような確認統制が必要である。

BEC 対策の最低線は、次のように整理しやすい。

場面	メールだけで決めないもの	最低限の確認
送金依頼	振込実行、金額変更	既知の番号へ折り返し、別承認者確認
口座変更依頼	振込先口座の差し替え	取引先へ別チャンネルで確認
請求先変更依頼	請求書送付先、担当変更	既存連絡先で確認
機微情報送付依頼	人事、契約、顧客情報送付	依頼者確認、承認経路確認

最小権限も、ここで効く。もしアカウントが侵害されても、何でもできる権限でなければ被害面積は小さくなる。第8章で詳しく扱った通り、管理者権限の分離、役割ごとの権限、共有アカウントの抑制は、セキュリティ基礎対策でも中心にある。

もう一つ大切なのが、報告導線である。怪しいメールを受けた時、クリックしてしまった時、見慣れない承認通知が来た時、どこへ連絡すればよいか曖昧だと、初動は遅れる。ここでは完璧な CSIRT は要らない。この窓口へ連絡する この件名で送る まずネットワークを切る のような最低線があるだけでよい。

利用者教育も、この文脈で書く方がよい。教育の目的は、全員をセキュリティ専門家にするのではない。怪しいものに気付き、止まり、報告できるようにすることである。責める文化より、早く知らせる文化の方が強い。

## 小規模組織で優先すべき対策

時間も予算も限られるなら、全部を同時にやろうとしない方がよい。第17章での現実的な優先順位は、次のようになる。

優先順	やること	先にやる理由
1	管理者アカウントに MFA を強制する	最も危険な権限を先に守る
2	一般利用者へ MFA を広げる	password only の侵入を減らす
3	legacy authentication や less secure apps を止める	MFA の抜け道を消す
4	既定の anti-phishing と anti-spoofing を確認する	もっとも多い入口を減らす
5	built-in antivirus と firewall が有効な端末だけを使う	端末起点の侵入を減らす
6	guest Wi-Fi と社内ネットワークを分ける	被害の広がりを抑える
7	送金、口座変更、請求先変更は二経路確認にする	BEC の実害を止める

この順序には理由がある。最初の三つは、認証突破の入口をかなり減らす。次の二つは、よくあるメール起点と端末起点の侵入口を減らす。最後の二つは、侵入やなりすましがあっても被害が広がりにくくする。

ここで意図的に避けたいのは、**高度な仕組みがないから後回し**という考え方である。EDR、CASB、ZTNA のような製品があれば強い場面はある。だが、その前に MFA も guest 分離も二経路確認もないなら、優先順が逆になる。

日本の小規模組織では、IPA の SECURITY ACTION 二つ星のような自社診断と基本方針から始める導線もある。第17章で伝えたいのは、完璧な制度を先に作ることでなく、最低線を順にそろえることだ。

## 通常時の例と、崩れた時の例

通常時の例では、管理者アカウントに先に MFA がかかっており、一般利用者も段階的に MFA へ移行している。古い認証方式は棚卸しされ、Google Workspace や Microsoft 365 のメール保護設定も確認済みである。自社ドメインには SPF、DKIM、DMARC の考え方が入り、まず監視しながら段階導入している。Windows 端末では built-in のウイルス対策と firewall が有効で、ローカル管理者での常用は避けている。guest Wi-Fi は社内と分かれ、送金や口座変更依頼は既知の電話番号で折り返し確認する。怪しいメールを見た時の報告先も決まっている。結果として、攻撃が絶対になくなるわけではないが、よくある入口で事故が起きにくい。

崩れた時の例では、管理者アカウントに MFA がなく、一般利用者にも password only の経路が残っている。古いメールクライアントや機器のために旧式認証が放置され、怪しいメール対策は利用者注意だけで済ませている。自社ドメインの送信認証は整っておらず、guest Wi-Fi と社内ネットワークも同じである。ある日、経理担当に役員名義の送金依頼メールが届き、そのまま処理され

る。後で調べると、差出人表示は似ていたが別ドメインで、しかも送金依頼を確認する二経路ルールもなかった。問題は、一つの製品が足りなかったことではなく、基礎対策の最低線がなかったことである。

## 最低限ここまではやる

第17章の内容を一度にすべて整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 管理者アカウントに MFA を強制する
2. legacy authentication や less secure apps の残りを確認する
3. anti-phishing と anti-spoofing の既定保護を確認する
4. built-in antivirus と firewall を無効化していない端末だけを使う
5. 送金や口座変更はメール一本で確定しない

この五つがあるだけで、セキュリティ基礎対策はかなり前進する。セキュリティ基礎対策とは、高価な仕組みを足すことではない。よくある侵入口を減らし、認証を近代化し、被害が広がりにくい最低線を作ることである。

## 今日、今週、後でやること

今日やることは、管理者アカウントに MFA が強制されているかを確認し、同時に旧式認証や password only の例外が残っていないかを一つ洗うことである。

今週やることは、メール保護設定を見直し、受信側の anti-phishing 設定と、自社送信ドメインの SPF、DKIM、DMARC の現状を一覧にすることである。あわせて、送金や口座変更の二経路確認ルールを文書化したい。

後でやることは、guest Wi-Fi 分離、管理画面の保護、報告導線の明確化のような **社内だから安全** 前提を崩す整備を順に進めることである。ここまでできると、セキュリティは不安の話から、優先順位を持って管理する実務へ変わる。

## 第18章

## 脆弱性管理、パッチ管理、構成管理

脆弱性情報は毎日のように出る。OS、ブラウザ、VPN 装置、業務アプリ、ネットワーク機器。重大と書かれた記事を見るたびに不安になる。だが、実務で本当に必要なのは、ニュースを多く追うことではない。その情報が自社に関係あるかを判断し、優先順位を付け、当て、確認し、当てられないものには別の守りを置くことである。ここができていないと、毎回騒いで終わる。

第18章で扱うのは、この定常運用である。NIST SP 800-40 Rev.4 は patching を preventive maintenance for technology と位置づけ、enterprise patch management を identifying、prioritizing、acquiring、installing、verifying の流れで整理している。つまり、パッチ管理は臨時対応ではなく、保守運用である。さらに、脆弱性管理、パッチ管理、構成管理は別々の仕事に見えて、実際には一つにつながっている。何を持っているかを知り、どこに危険があるかを知り、どこから直し、当てられない時はどう困り、どこを標準へ戻すかを定める仕事だからである。

### 脆弱性管理、パッチ管理、構成管理は一つの循環である

第18章では、まず骨格を固定したい。脆弱性対応は、次の循環で考えるとぶれにくい。

段	何をするか	最低限残すもの
収集	vendor advisory、JVN / JVN iPedia、KEV などを見る	情報源、確認日

段	何をするか	最低限残すもの
影響判断	自社で使っているか、どの版か、公開有無を確認する	対象資産、製品名、版
優先順位付け	どれを急ぐか、通常月次へ回すかを定める	優先度、判断理由
適用または代替策	patch、upgrade、設定変更、公開停止などを実施する	実施内容、実施日
確認	当たったか、失敗したか、再起動や業務影響はどうかを見る	成否、失敗端末、再試行予定
例外管理	当てられないものを保留で終わらせず、代替策と期限を持つ	保留理由、代替策、期限、担当

この中で特に抜けやすいのが、**影響判断 確認 例外管理** である。影響判断がないと、重大ニュースに引きずられやすい。確認がないと、配ったつもりで終わる。例外管理がないと、古い機器がそのまま残る。脆弱性管理とは、CVE を見た回数ではなく、この循環が回っているかで決まる。

## 情報収集元は、役割を分けて持つ

脆弱性対応が崩れやすいのは、情報源が定まっていないからである。検索して見つける運用では遅い。少なくとも、見る場所の役割を分けておきたい。

情報源	主な役割	ここで見ること
vendor advisory	一次情報	affected versions、fixed versions、workaround、再起動要否
JVN / JVN iPedia	日本語での集約	国内外の脆弱性対策情報の把握

情報源	主な役割	ここで見ること
MyJVN	収集支援と版確認支援	情報収集の効率化、PC上の製品版確認
CISA KEV	active exploitationの優先判断	exploited in the wildかどうか
NVD	severityの補助線	CVSS、CVEメタデータ

最優先になるのは、製品ベンダーの advisory である。affected versions、solution、workaround、回避策、再起動要否のような一次情報は、たいていここにある。自社が使っている VPN、OS、ブラウザ、認証基盤、主要 SaaS の advisory 受信先は、最初に決めておいた方がよい。

日本語の集約源として有用なのが、JVN と JVN iPedia である。JVN iPedia は、JVN に加え、国内外で日々公開される脆弱性対策情報を集約するデータベースとして案内されている。さらに英語説明では、JVN、国内製品開発者、NVD の情報を集約するとされている。MyJVN は、脆弱性対策情報の効率的収集や、利用者 PC 上のソフトウェア製品バージョン確認を支援する仕組みとして案内されている。ひとり情シスにとっては、英語の vendor advisory だけで追うより現実的である。

そのうえで、active exploitation の優先判断に役立つのが CISA の Known Exploited Vulnerabilities、いわゆる KEV である。CISA は KEV Catalog を、exploited in the wild であることが確認された脆弱性の authoritative source として維持し、vulnerability management prioritization framework の input として使うよう勧めている。つまり、理論上危険 と すでに狙われている は分けて扱うべきだということである。

ここで忘れてはいけないのが、自社の製品名とバージョンの一覧である。何を見ればよいか決まっても、自社のどこにその製品があるか分からなければ判断できない。脆弱性管理は、情報収集だけで成立しない。資産と版が結び付いて初めて動く。

## 優先順位は CVSS だけで決めない

NVD は、CVSS について **severity** を測る仕組みであり、**risk** そのものではないと明示している。ここは第18章の重要点である。CVSS 9.8 だから即日対応、6.5 だから来月でよい、と機械的には決められない。

優先順位付けでは、少なくとも次を重ねたい。

観点	何を見るか	優先度が上がりやすい条件
該当性	自社がその製品と版を使っているか	使っている版が affected versions に入る
公開性	外から触れるか	インターネット公開、VPN、公開 Web、メール入口
入口性	認証、VPN、管理画面など入口資産か	認証基盤、管理系、メール、公開資産
業務影響	止まる業務や扱うデータは何か	基幹業務、機微情報、全社利用
攻撃状況	すでに狙われているか	KEV 掲載、vendor が exploitation に言及
代替策	workaround や回避策があるか	patch 前に被害面を減らせるか
適用影響	今すぐ当てた時の停止影響はどうか	先に業務調整や段階展開が必要

重要なのは、**severity** が高い と **自社にとって今すぐ危ない** を分けて考えることである。たとえば、社内だけで使う端末向けの中程度脆弱性と、インターネット公開中の VPN 装置にある高めの脆弱性では、後者を先に見る方が自然である。さらに、その脆弱性が KEV に載っているなら、優先度はもう一段上がる。

逆に、CVSS が高くても、自社でその機能を使っていない、公開もしていない、代替策もすでに入っているなら、急ぎ方は変わる。大切なのは、**点数を見る**ではなく **自社へ当て**ることである。

## 通常更新と緊急更新を分ける

NIST SP 1800-31 は、routine and emergency patching situations を分けて扱っている。これはそのまま実務に使える。通常更新と緊急更新は、同じレーンで回さない方がよい。

項目	通常更新	緊急更新
起動条件	月次、週次の定常保守	KEV 掲載、active exploitation、公開入口資産
展開方法	test、pilot、production の段階展開	影響確認後に短縮経路で展開
調整事項	再起動時間、業務時間、失敗端末追跡	公開停止、機能停止、workaround 先行
確認	適用率、失敗率、再試行	当日中の影響確認、露出低減、適用確認
戻し方	Pause、Resume、Uninstall の手順を用意	戻すより先に被害面を減らす判断を優先

通常更新では、月次または週次の更新日を決め、test、pilot、production の順で段階展開する。Microsoft Intune の update rings も、deployment stages として test、pilot、production を作る考え方を取っている。deferral periods、restart

settings、deadlines を調整でき、Pause、Resume、Uninstall もできる。製品名はさておき、考え方としては、小さな会社でも参考になる。重要なのは、一斉配布一発勝負にしないことだ。

一方、緊急更新は別である。active exploitation が確認され、通常の deferral を待つ余裕がない時は、緊急レーンへ乗せる必要がある。Microsoft の expedited updates も、out-of-band security update for a zero-day flaw を急ぐ用途を示しており、deferrals and other settings を一時的に上書きして早く入れる仕組みとして整理されている。ここから分かるのは、緊急更新は例外ではなく、運用として事前に用意しておくべきだということである。

ブラウザのような日常利用ソフトでも同じである。Chrome は 2025年3月25日の stable update で、CVE-2025-2783 について **exploit exists in the wild** と明示した。こういう時は、通常月次を待つより、影響確認と更新可否確認を先に回す方がよい。

## 更新できない資産には代替策を置く

ここは非常に重要である。更新できない資産は必ず出る。古い業務ソフト、依存関係が深い装置、停止できないサーバー、ベンダー都合で fix が遅い機器。問題は、当てられないこと自体より、当てられないまま何も足さないことだ。

NIST SP 1800-31 は、patching の代替として isolation methods や emergency mitigations を挙げている。つまり、patch 不可は **終わり** ではなく **別の守りへ切り替える合図** である。

現実的な代替策は次のようなものだ。

代替策	何をするか	向いている場面
公開停止	外部公開を止める	公開資産の緊急時

代替策	何をするか	向いている場面
機能停止	問題機能を無効化する	一部機能だけ危ない時
workaround 適用	vendor の回避策を入れる	advisory に暫定策がある時
到達制限	接続元や管理経路を絞る	すぐ止められない時
分離	別セグメントや別経路へ逃がす	古い機器や停止困難機器
監視強化	関連ログや異常検知を厚くする	暫定で時間を稼ぐ時
置換期限設定	いつまでに入れ替えるか決める	恒久 fix が出ない時

Fortinet の 2026年2月10日の PSIRT では、Agentless VPN / FSSO の認証回避脆弱性に対し、FortiOS 7.6.0 through 7.6.4 を affected、7.6.5 or above を solution とし、Disable unauthenticated bind on the LDAP server を workaround として示している。こうした advisory は実務的である。すぐ更新できないなら、公開を止めるか、回避策を入れるか、到達経路を絞るかを決めなければならない。

ここで避けたいのは、今は当てられないので保留 で止めることだ。保留するなら、最低でも次は残したい。

項目	何を書くか
対象資産	機器名、ホスト名、担当
対象脆弱性	advisory 名、CVE、確認日
保留理由	停止不可、依存あり、fix 未提供など
代替策	公開停止、設定変更、分離など
見直し期限	次回確認日
恒久対応	upgrade、置換、廃止予定

## 構成管理は、標準と逸脱を管理する

脆弱性管理と構成管理は別物に見えるが、実際には近い。標準構成から逸脱していると、同じ脆弱性でも被害が大きくなりやすいし、更新も当たりにくくなる。

NIST の National Checklist Program は、security configuration checklist を lockdown、hardening guide、benchmark を含むものとして位置づけ、configured properly の確認や unauthorized changes の特定にも使えるとしている。Microsoft の security baselines も、Microsoft-recommended configuration settings をまとめたもので、well-known and well-tested な baseline を使うことを勧めている。つまり、構成管理とは **最初に固める** だけでなく **今もその状態を見る** ことでもある。

第18章で扱いたい構成管理は、次の四区分である。

区分	何を持つか	例
標準構成	基本の設定値	firewall 有効、標準ユーザー、TLS 方針
例外	標準から外す正当理由	旧式業務アプリのため一部設定緩和
逸脱	許可なく外れている状態	ある端末だけ firewall 無効
是正	標準へ戻すか、例外化するか	設定修正、例外台帳登録、廃止判断

ここで重要なのは、標準をゼロから作り込まないことである。小さな会社ほど、既存の baseline や checklist を起点にした方が速く、ぶれにくい。逆に、何も基準がないと、ある PC だけ firewall が切られている、あるサーバーだけ古い TLS 設定が残っている、ある端末だけローカル管理者のまま使われている、といった逸脱が静かに積み上がる。これをそのままにすると、脆弱性管理をいくら回しても、足元から崩れる。

## 小さな会社で現実的に回るやり方

全部の製品を同じ密度で追うのは現実的ではない。小さな会社では、重要資産から回す方がよい。

現実的には、次のくらいから始めるとよい。

頻度	やること
日次	vendor advisory、JVN / JVN iPedia、KEV をざっと確認する
週次	重要資産への該当有無、更新失敗端末、例外期限を確認する
月次	通常更新を回し、失敗端末再試行、例外見直し、baseline 逸脱確認をする

対象資産は、まず上位 10 件から 20 件でよい。OS、ブラウザ、VPN 装置、認証基盤、公開サーバー、主要業務アプリ、ネットワーク機器など、止まると困るものから始める。その一覧には、最低でも **製品名 版 公開有無 担当** を持ちたい。

緊急更新条件も、最初は単純でよい。たとえば次のように置ける。

- KEV 掲載
- active exploitation が確認されている
- インターネット公開資産が対象
- 認証、VPN、メール、公開 Web の入口に関わる

この条件に当たるものは、通常月次を待たずに同日中に影響確認を始める。そこまで大げさでなくても、公開停止判断をする、workaround を検討する、更新可否を確認する、といった最低線は持ちたい。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスが主要資産一覧を持っている。OS、ブラウザ、VPN 装置、認証基盤、公開サーバー、主要業務アプリの製品名と版が分かる。vendor advisory と JVN / JVN iPedia を普段から見ており、KEV 掲載の有無も確認する。CVSS は参考にするが、そのまま順位にはせず、公開有無、入口性、業務影響を重ねて判断する。月次更新は test、pilot、production で段階展開し、失敗端末は一覧で追う。active exploitation が確認された時だけ緊急レーンへ切り替え、必要なら公開停止や workaround を先に入れる。標準構成から外れた端末や機器も定期的に見直し、標準へ戻す。結果として、ニュースに反応するのではなく、運用として回る。

崩れた時の例では、脆弱性情報は X やニュースで知るだけで、どの製品を自社が使っているかはすぐ出てこない。CVSS が高いものだけを追っているが、インターネット公開中の認証機器は後回しになる。月次更新は一斉配布で、失敗端末は把握していない。fix が出ない機器もそのまま公開し続ける。標準設定からの逸脱も見していない。ある日、VPN 装置の advisory が出るが、影響確認が遅れ、暫定対策も打たないまま数日が過ぎる。問題はニュースの見落としではなく、定常運用がなかったことである。

## 最低限ここまではやる

第18章の内容を一度にすべて整えなくてもよい。だが、次の六つは早めに持ちたい。

1. 重要資産の製品名と版を一覧にする
2. vendor advisory、JVN / JVN iPedia、KEV の見る先を決める
3. CVSS 以外の優先判断軸を持つ

4. 通常更新と緊急更新のレーンを分ける
5. 更新できない資産の代替策と期限を記録する
6. 標準構成、例外、逸脱の三区分を持つ

この六つがあるだけで、脆弱性管理はかなり実務的になる。脆弱性管理とは、CVE を眺めることではない。自社に関係あるものを見極め、優先順位を付け、当て、当てられないものは代替策で囲い、標準構成へ戻し続ける運用である。

## 今日、今週、後でやること

今日やることは、公開資産と認証系資産を中心に、重要資産を 10 件だけ書き出し、製品名と版が分かるかを確認することである。分からないものは、それだけで脆弱性管理上の宿題になる。

今週やることは、その一覧に対して advisory の受信先を決め、通常更新日と緊急更新条件を一枚にまとめることである。あわせて、更新不能資産があれば、保留理由と代替策を書きたい。

後でやることは、標準構成を一つ決め、逸脱一覧を持ち、更新失敗端末や例外資産を月次で見直すことである。ここまでできると、脆弱性対応は不安ベースの作業から、優先順位を持って回る保守運用へ変わる。

## 第19章

## 個人情報保護、社内規程、監査対応

個人情報保護の話になると、多くの会社はすぐに規程や同意文の話を始め  
る。もちろんそれもある必要である。だが、ひとり情シスの実務で本当に詰まる  
のはそこではない。どの業務が個人情報を扱っているのか、どこに保存され  
ているのか、誰が触れるのか、外へどう出ていくのか、いま残っている記録  
で説明できるのか。ここが曖昧なままでは、規程を何本作っても回らない。

個人情報保護委員会のガイドラインや Q&A は、法律の解釈だけでなく、かな  
り実務的なヒントを含んでいる。責任者を置くこと、取扱規律を決めること、ア  
クセス制御や教育を行うこと、取扱状況を確認できる記録を残すこと、委託と第  
三者提供と共同利用を切り分けること。第19章では、これらを **日常運用へどう落  
とすか** に絞って整理する。

### 個人情報、個人データ、保有個人データを分けて考える

最初に押さえたいのは、似た言葉を混ぜないことである。個人情報保護委員会の  
通則編は、**個人情報 個人データ 保有個人データ** を明確に使い分けている。ここが  
曖昧だと、必要な義務も曖昧になる。

実務へ落とすなら、次の表で整理しやすい。

用語	実務で見ているもの	主に気にすること
個人情報	名刺、メール本文、応募書 類、問い合わせ内容、録音 記録	どこに散らばっているか

用語	実務で見ているもの	主に気にすること
個人データ	顧客管理、人事台帳、勤怠、給与、問い合わせ管理のように検索して管理しているもの	誰が見られるか、何に使うか、外へ出るか
保有個人データ	本人からの開示、訂正、利用停止等に自社が応じる権限を持つもの	周知事項、請求手続、保存期間、削除方法が決まっているか

つまり、個人情報保護の第一歩は、用語を覚えることではなく、次を言えるようにすることである。

- どの業務が個人情報を扱っているか
- どのシステムや保存場所にあるか
- どの項目を持っているか
- 利用目的は何か
- 誰がアクセスできるか
- 外部へ出る時の区分は何か
- いつまで持ち、どう消すか

通則編は、取扱状況を確認する手段として、少なくとも **個人情報データベース等の種類・名称 個人データの項目 責任者・取扱部署 利用目的 アクセス権を有する者** を明確にすることを考えている。最初の管理表は、そのままこの形でよい。

項目	何を書くか
データベース等の種類・名称	人事 SaaS、給与 SaaS、問い合わせフォーム、共有フォルダなど
個人データの項目	氏名、住所、電話、評価、応募書類、問い合わせ本文など
責任者・取扱部署	人事責任者、営業責任者、情シス担当など

項目	何を書くか
利用目的	採用、給与支給、顧客対応、契約管理など
アクセス権を有する者	人事担当、経理担当、営業責任者など
外部に出る区分	社内のみ、委託、第三者提供、共同利用
クラウド利用	利用あり / なし、保存国、事業者が取扱うか
保存期間	1年、2年、在職中、契約終了後何年など
削除方法	管理画面削除、保管期限到来後削除、返却、廃棄証明など

この洗い出しがないと、個人情報保護はいつまでたっても **総論** のままで終わる。監査で困るのも、削除で困るのも、漏えい時に困るのも、たいていはここが曖昧だからである。

なお、給与や社会保険の実務では、マイナンバーのように本章よりさらに強いルールがかかる情報もある。第19章は個人情報保護全般の土台を扱う章であり、そうした個別制度の詳細には深入りしないが、**同じ個人情報でも追加ルールがある場合がある** ことは意識しておきたい。

## 安全管理措置を、日常運用へ落とす

個人情報保護の話が空中戦になりやすい理由は、**安全管理措置を講じる** という言葉が抽象的だからである。通則編は、単に強く守れと言っているのではない。基本方針、取扱規律、組織的、人的、物理的、技術的な措置へ落としている。なお、外国で個人データを取り扱う場合は、これらに加えて外的環境の把握も求められる（通則編の別添では計7区分）。ひとり情シスにとって大事なのは、この分け方をそのまま運用へ持ち込むことだ。

実務では、次の表で見ると整理しやすい。

区分	最低限決めること	既存運用とのつながり	残る証跡
基本方針	事業者名、法令遵守、窓口、安全管理の考え方	プライバシーポリシー、社内方針	公開文書、改定履歴
取扱規律	取得、利用、保存、提供、削除・廃棄の手順	申請フロー、削除手順、外部送付承認	規程、申請テンプレート
組織的措置	責任者、報告連絡体制、点検、監査	月次確認、四半期点検	点検記録、監査記録、是正記録
人的措置	教育、秘密保持、異動・退職時の見直し	入社説明、権限棚卸し、JML	教育記録、誓約、棚卸し記録
物理的措置	紙、媒体、端末、廃棄、持ち出し管理	施錠、持出申請、端末回収	持出記録、返却記録、廃棄証明
技術的措置	アクセス制御、識別認証、監視、更新	IAM、ログ確認、パッチ適用、端末保護	アクセスログ、設定記録、更新記録

ここで重要なのは、**安全管理措置** を新しい仕事にしないことである。第8章の権限管理、第9章の異動・退職処理、第10章の端末回収、第15章の削除や廃棄、第17章の基礎対策、第18章のパッチ管理は、そのまま個人情報保護法上の安全管理措置にもなる。

特に、組織的安全管理措置では **利用状況等を記録すること** と **取扱状況を確認する手段を整備すること** が重要である。技術的安全管理措置も、アクセス制御だけではない。Q10-20 は、アクセス状況の監視、異常記録の定期確認、対策ソフトの更新確認、修正ソフトウェアの適用、未許可ソフト導入防止まで挙げている。つまり、ログ確認やパッチ管理は、セキュリティ運用であると同時に個人情報保護の運用でもある。

## 委託、第三者提供、共同利用を分け、クラウドは別軸で見る

第19章で最も実務的な論点がここである。外部へ個人データが出る場面は多い。給与計算、配送、サポート、クラウド、共催、親会社連携、業務提携。ところが、この全部を **委託** と呼んだり、逆に全部を **第三者提供** と呼んだりすると、同意、監督、記録の判断を誤る。

迷った時は、次の表で見ると判断しやすい。

区分	何で見分けるか	同意・周知の考え方	最低限残すもの
委託	相手は自社の指示範囲だけで処理するか	利用目的達成に必要な範囲内なら第三者に該当せず、本人同意不要	契約書、委託範囲、監督方法、削除・返却条件
第三者提供	相手が自社の目的で利用するか	原則として本人同意や他の法的根拠を要確認	本人への説明、承認記録、提供記録
共同利用	特定の者と共同して使い、必要事項を通知又は容易知得状態に置くか	第三者に該当しないが、共同利用の事項を周知する必要がある	共同利用の通知、範囲、利用目的、責任者
クラウド利用	法的区分ではなく、クラウド事業者が個人データを取り扱うこととなっているかを見る視点	委託、第三者提供、どちらでもない場合があり、別途安全管理措置と保存場所を確認する	契約条件、取扱可否、保存場所、アクセス制御、ログ

つまり、**委託** **第三者提供** **共同利用** が法的な整理であり、**クラウド利用** はその整理をする時に追加で確認する実装形態だと考えると混乱しにくい。

**委託** は、利用目的の達成に必要な範囲内で、個人データの取扱いを外部へ委ねる場合である。給与計算、配送、データ入力、運用代行などが典型である。この場合、提供先は第三者に該当しない。ただし、委託先は委託された業務の範囲内でのみ取り扱える。委託先が自社分析や営業に使い始めたら、もう委託の話ではなくなる。

**第三者提供** は、相手が自社の判断で利用する場合である。業務提携先へ顧客一覧を渡す、共催先へ参加者名簿を渡す、親会社や関連会社へ各社の判断で使える形で渡す、という場面はここに近い。ここを **委託です** と呼んで済ませると危ない。

**共同利用** も、第三者に該当しない場面である。ただし、何でも共同利用と呼ばばよいわけではない。共同利用の旨、共同して利用される項目、利用者の範囲、利用目的、管理責任者の情報等を、あらかじめ本人へ通知するか、本人が容易に知り得る状態に置く必要がある。実務では、共同利用と書けば自由に渡せる、ではなく、必要事項込みで説明できるかを見るべきである。

もう一つ誤解が多いのがクラウド利用である。Q7-53 は、クラウド利用が第三者提供や委託に当たるかを、**クラウド事業者が個人データを取り扱うこととなっているか** で判断するとしている。つまり、クラウドだから自動的に第三者提供、クラウドだから自動的に委託、ではない。

たとえば、契約上、クラウド事業者が保存データを取り扱わないこととなっており、適切なアクセス制御もされている場合には、第三者提供でも委託でもない整理があり得る。逆に、事業者が保守や運用のために個人データを取り扱うなら、委託として見る必要が出てくる。

ただし、ここで止めてはいけない。Q7-54 は、クラウド利用が法第27条の **提供** に当たらない場合でも、利用事業者は自ら果たすべき安全管理措置の一環として適切な措置を講じる必要があるとしている。つまり、**法的には提供ではない** と **何もしなくてよい** は別である。

さらに、外国にあるクラウドサービスを使う場合は、Q10-25 の論点もある。外国クラウドで個人データを保存するなら、当該外国の制度等を把握した上で安全管理措置を講じる必要があり、保有個人データについては、クラウド事業者所在国や保存サーバ所在国、把握した制度等を踏まえて講じた措置の内容を、本人の知り得る状態に置く必要がある場合がある。ひとり情シスは、**クラウドだから便利** の前に、保存場所と説明責任を確認した方がよい。

## 社内規程と本人向け周知は、短く具体的に整える

規程は多ければよいわけではない。読まれず、守られず、監査でも説明に使えない規程は、実務上ほとんど価値がない。ひとり情シスが目指すべきなのは、短くても判断に使えるルールである。

最低限、次の文書や表示は整理しておきたい。

文書・表示	最低限入れること
基本方針	事業者名、法令遵守、安全管理、質問・苦情窓口
取扱規程	取得、利用、保存、提供、削除・廃棄、責任者、担当者、承認フロー
本人向け周知	利用目的、苦情窓口、開示等請求手続、安全管理措置の概要
開示等請求手順	受付方法、本人確認、回答方法、手数料、期限

特に **保有個人データ** では、本人の知り得る状態に置くべき事項がある。最低限、次は明確にしたい。

項目	何を示すか
事業者情報	氏名又は名称、住所、法人なら代表者氏名
利用目的	全ての保有個人データの利用目的
請求手続	開示、訂正、利用停止等の受付方法
手数料	定めているならその額
苦情窓口	申出先、連絡先
安全管理措置の概要	どのような考え方で保護しているか

ここで大事なのは、抽象語を減らすことだ。たとえば次の二つを比べると差が分かる。

- 悪い例
  - 個人情報は適切に管理する
- 良い例
  - 人事システムの閲覧権限は人事責任者承認で付与し、四半期ごとに棚卸しする

前者はきれいだが運用できない。後者は地味だが、申請、承認、棚卸し、証跡までつながる。ひとり情シスの規程は、こういう具体性を持っていた方が強い。

また、ルールは **個人情報保護だけの文書** に閉じ込めない方がよい。入社、異動、退職のフロー、端末配布ルール、共有フォルダ権限、SaaS 導入申請、外部送付承認。これら既存運用に個人情報保護の観点を埋め込む方が、別紙を増やすより実効性が高い。

## 監査で見られる証跡を、平時から残す

監査対応でありがちな失敗は、規程ファイルだけを揃えて安心することだ。監査で本当に見られるのは、**そのルールが実際に回っているか**である。言い換えると、監査は書類の有無より、運用の痕跡を見る。

個人情報保護の監査で役に立つ証跡は、たとえば次のようなものである。

見られる論点	出せる証跡
どの業務がどの個人データを持つか	業務一覧、台帳、システム一覧
誰がアクセスできるか	権限申請、承認、棚卸し、アクセスログ
外部へどう出しているか	契約書、共同利用通知、提供承認、転送ログ
委託先をどう見ているか	委託契約、再委託条件、点検記録
いつ消したか	削除記録、返却記録、廃棄証明
教育や点検をしているか	研修記録、点検記録、監査記録、是正記録

ここで実務的に重要なのが、**専用の巨大台帳を新しく作ることだけが答えではない**という点である。Q13-25 は、既存の契約書などで記録事項を満たしていれば記録として認められるとしている。Q13-26 は、伝送日時や伝送先のログを記録と認めている。Q13-27 は、継続的・反復的授受では、基本契約書と付帯資料をあわせて記録とする方法を認めている。

つまり、申請フォーム、承認記録、契約書、付帯資料、転送ログ、削除証明を、後で引けるように保存しておけばよい。監査前に慌てて新しい一覧を作るより、日常運用の中で自然に残る形へした方が強い。

また、証拠は **ある** だけでは弱い。どこにあるか、誰が取り出せるか、いつまで残すかが決まっていないと、監査当日に見つからない。証拠の保存場所を一つか二つに絞り、命名と保存期間を決めておきたい。

## 小さな会社で現実的に回るやり方

すべての個人情報を一気に洗い出し、全ルールを一斉更新し、全システムの証拠を整理するのは現実的ではない。小さな会社ほど、重要な業務から順に固めた方がよい。

現実的には、次の五つくらいから始めると回しやすい。

- 人事
- 勤怠
- 給与
- 顧客管理
- 問い合わせ対応

最初の管理表は、たとえば次の形でよい。

業務名	保存場所	主な項目	利用目的	アクセス役割	外部区分	クラウド利用	保存期間	削除方法
採用	採用SaaS、共有フォルダ	氏名、連絡先、職務経歴	採用選考	人事責任者、人事担当	委託	あり	1年	期限到来後に削除

業務名	保存場所	主な項目	利用目的	アクセス役割	外部区分	クラウド利用	保存期間	削除方法
問い合わせ	問い合わせフォーム、メール	氏名、会社名、連絡先、本文	顧客対応	営業責任者、営業担当	委託	あり	2年	保持期間後に削除
顧客管理	CRM	氏名、所属、連絡先、商談履歴	営業活動、契約管理	営業責任者、営業担当	共同利用	あり	契約終了後3年	期限到来後に削除

この表ができるだけで、かなりの論点が見える。共有フォルダに昔の応募書類が残っている。給与データの委託契約が古い。問い合わせフォームの転送先が個人メールである。退職者のフォルダが削除されていない。どれも、表にしないと見えにくい。

次に、確認サイクルだけ決める。

頻度	やること
月次	アクセス権変更、外部送付承認、削除予定確認
四半期	主要業務の棚卸し、委託先確認、証跡保存場所の点検
年次	規程と本人向け周知の見直し、教育、主要委託契約の再確認

ここで大事なのは、完璧な制度を先に作らないことである。五業務の一覧化、外部区分の分類、証跡保存場所の決定。この三つだけでも、個人情報保護の実務はかなり前に進む。

## 通常時の例と、崩れた時の例

通常時の例では、人事、勤怠、給与、顧客管理、問い合わせ対応の五業務が一覧化されている。給与計算は外部委託として整理され、契約書、委託範囲、削除や返却条件が確認されている。顧客管理システムはアクセス権が役割ごとに分かれ、退職や異動のたびに見直される。問い合わせデータを外部ベンダーへ渡す場合は、委託が第三者提供か共同利用かを事前に判断し、承認記録と転送記録を残す。プライバシー案内には、利用目的、苦情窓口、開示等請求手続、安全管理措置の概要が整理されている。監査で聞かれた時も、規程だけでなく、申請、承認、棚卸し、ログを出せる。

崩れた時の例では、個人情報保護規程はあるが、どの業務がどの個人データを持っているか一覧がない。給与委託先へ渡しているデータの範囲も曖昧で、契約書もすぐ出てこない。営業が提携先へ顧客一覧を渡しているが、委託と呼んで済ませており、実際には相手が自社目的で利用している。共同利用と書いているが、利用者の範囲や責任者の表示がない。問い合わせフォームのデータは個人メールへ自動転送され、アクセス権棚卸しもしていない。監査で **誰がアクセスできるか どの承認で外へ出したか 削除した証跡はあるか** と聞かれると答えに詰まる。問題は、法律を知らないことより、運用と証跡がないことである。

## 最低限ここまではやる

第19章の内容を一度に全部整えなくてもよい。だが、次の六つは早めに持ちたい。

1. 個人データを扱う主要業務とシステムを一覧にする
2. 各業務について、利用目的、アクセス権、外部区分、保存期間、削除方法を定める

3. 外部へ出る場面を、委託、第三者提供、共同利用に分けて整理し、クラウド利用は保存国や取扱有無を別欄で確認する
4. 保有個人データに関する周知事項と社内ルールを、短く具体的に整える
5. 契約書、承認記録、転送ログ、棚卸し記録、削除証明を、後で引ける形で保存する
6. 外国クラウドを使うなら、保存国と説明事項を確認する

この六つがあるだけで、個人情報保護はかなり実務になる。個人情報保護とは、規程を置くことではない。どの業務でどの個人データを扱い、誰が触れ、どこへ渡し、どんな記録を残すかを説明できる状態を作ることである。

## 今日、今週、後でやること

今日やることは、人事、給与、勤怠、顧客管理、問い合わせ対応の五業務だけを取り上げ、どのシステムや保存場所に個人データがあるかを書き出すことである。ここで空欄になる項目が、そのままリスクになる。

今週やることは、その一覧に **利用目的** **アクセスできる役割** **外部区分** **クラウド利用** **保存期間** **削除方法** を足すことである。あわせて、外部に出る場面を委託、第三者提供、共同利用に分け、クラウドは保存国や取扱有無を別欄で確認しながら、承認経路と記録の残し方を決めたい。

後でやることは、本人向け周知事項を見直し、権限棚卸し、削除証跡、点検記録の残し方を整えることである。ここまでできると、個人情報保護は **規程を持っている会社** から **運用を説明できる会社** へ変わる。

## 第20章

# 教育、訓練、委託先を含むセキュリティ運用

セキュリティ事故は、設定ミスだけで起きるわけではない。怪しいメールを開いてしまう。口座変更依頼をメール一本で信じてしまう。委託先が共有アカウントで接続している。取引先から届いたファイルを、いつもの共有手順を通さず開いてしまう。どれも、技術対策だけでは防ぎきれない。だから、セキュリティを本当に回すには、人と外部接点を含めた運用が必要になる。

第20章で扱うのは、その運用である。年1回の研修を実施したかどうかではない。怪しい時に止まれるか、報告できるか、経理や管理者が自分向けのリスクを理解しているか、委託先にも同じ最低限ルールが届いているか。ここまでできて初めて、セキュリティ教育は効き始める。

## 教育は、イベントではなく運用である

セキュリティ教育が形骸化する最大の理由は、**受講したかどうか** をゴールにしてしまうことだ。NIST SP 800-50r1 は、learning program を life cycle approach で運用し、behavior change と security culture につなげる考え方を示している。しかも対象は大企業だけでなく、小規模組織も含む。つまり、教育は単発イベントではなく、繰り返し改善する運用として持つ方がよい。

第20章では、教育運用を次の五つに分けると回しやすい。

要素	目的	具体例	最低限残すもの
注意喚起	直近の手口や変化に気づかせる	月次メール、社内チャット告知、ログオン画面の注意表示	配信記録、本文

要素	目的	具体例	最低限残すもの
基本学習	ルールと判断基準をそろえる	入社時説明、5分教材、年次学習	受講記録、教材
役割別学習	職務ごとの誤判断を減らす	経理向け BEC、人事向け個人情報、管理者向け特権運用	対象者一覧、教材
実践訓練	実際に動けるか試す	phishing 訓練、tabletop、連絡訓練	訓練計画、結果、振り返り
振り返り	次の改善へつなぐ	訓練後レビュー、実事例共有、手順修正	改善項目、改定履歴

NIST は、awareness activities を年間を通じて継続実施すること、practical exercises と training を分けることを明示している。CISA も、training の合間に最新手口を共有するよう勧めている。つまり、**年1回の研修** と **月次の注意喚起** と **四半期の訓練** は別物であり、全部あった方が強い。

教育の価値は、知識量より行動変化で見るべきである。NIST SP 800-50r1 も、measurement は compliance を超えて capabilities、attitude、behavioral changes を見るべきだとしている。小さな会社では複雑な指標は要らないが、最低限でも次は見たい。

見る項目	何を見るか	使い道
受講完了	誰が基本学習を受けたか	未受講者の補講
報告件数	怪しいメールや依頼の報告が来ているか	報告文化の有無確認
訓練報告率	訓練メールを報告できたか	早期検知の力を見る
訓練クリック率	リンクや添付に反応したか	題材や対象者の見直し
行動変化	MFA 利用、ルール順守、既知経路確認が増えたか	教育内容の有効性確認

見る項目	何を見るか	使い道
更新頻度	注意喚起や教材更新が止まっているか	教材の陳腐化防止

受講完了率だけで終わらせると、**分かったつもり** を量産しやすい。報告件数、訓練報告率、行動変化まで見ると、教育が運用へつながっているかが見えてくる。

## 標的型メール訓練と注意喚起を分けて回す

フィッシングや標的型メールへの対策は、訓練だけでは足りない。CISA は repeated training at regular intervals を勧める一方で、between trainings に最新手口を共有するよう勧めている。ここから分かるのは、訓練と日常の注意喚起は別物であり、両方必要だということだ。

項目	日常の注意喚起	標的型メール訓練
目的	新しい手口やルール変更を知らせる	実際に止まり、報告できるか試す
頻度	月次、必要時	四半期、半期、変更時
内容	最近多い手口、自社事例、報告先、確認方法	phishing、BEC、なりすまし、口座変更依頼
見るもの	読まれたか、共有されたか	難易度、クリック、報告、振り返り
残すもの	配信記録、掲示内容	訓練シナリオ、結果、改善点

CISA は、怪しい連絡を受けた時には、返信したり本文中の電話番号を使ったりせず、検索した電話番号や既知の連絡手段で確認するよう勧めている。ひとり情シスの現場では、メールの真偽を完全判定しようとするより、**怪しい時は既知の経路で確認する**を徹底した方が強い。

標的型メール訓練でよくある失敗は、クリック率だけを見て終わることである。NIST の Phish Scale は、訓練メールの human phishing detection difficulty を評価し、その文脈で click rate と report rate を読む考え方を示している。つまり、難しい題材でクリック率が高かったという事実だけでは、十分な評価にならない。

訓練では、最低限でも次を持ちたい。

項目	何を見るか	理由
題材	請求書、口座変更、パスワード期限切れ、配送通知など	自社に近い脅威を扱うため
対象者	全社員、経理、人事、管理者など	premise alignment を合わせるため
難易度	見抜きやすいか、見抜きにくい	結果の解釈を誤らないため
クリック	開封、リンク、添付反応	誘導に引っかかる傾向を見るため
報告	どれだけ早く報告されたか	被害最小化の力を見るため
直後フォロー	何を見逃したか、次にどうするか	懲罰で終わらせないため

NIST Phish Scale は、難易度を **email cues** と **premise alignment** で見ている。前者はメール自体にどれだけ不審な手掛かりがあるか、後者はその題材が受信者の職務や責任にどれだけ合っているかである。たとえば、経理向けの **振込先変更** は **premise alignment** が強くなりやすく、一般社員向けの雑な偽配送通知より難しい訓練になりやすい。

また、訓練題材は怪しい添付ファイルだけでは足りない。IPA の BEC 事例集には、偽口座送金、口座変更依頼、社長なりすまし、取引先アカウント乗っ取りが並んでいる。つまり、訓練は **リンクを踏むか** だけでなく、**送金や口座変更をどう確認するか** まで広げた方が実務に効く。

## 管理者向け教育と現場向け教育を分ける

全社員へ同じ内容を一斉配信するだけでは、必要なことが抜ける。NIST SP 800-50r1 は、**all users**、**privileged access account holders**、**significant responsibilities** を分けている。IPA の **5分でできる！情報セキュリティポイント学習** も、**経営者・管理者向け** と **従業員向け** を分けている。つまり、役割別教育は大企業だけの話ではない。

小さな会社では、最低限でも次のように分けたい。

対象	まず教えること	事故になりやすい誤判断
一般社員	不審メール、外部共有、パスワード、MFA、報告先	怪しい依頼に返信してしまう
新入社員	使ってよいツール、持ち出し、相談先、共有ルール	個人クラウドや個人メールを使う
管理職	例外承認、相談を受けた時の判断、事故時連絡	急ぎ案件を独断で通してしまう
経理	口座変更、送金依頼、請求書差し替え、既知経路確認	メール一本で支払先を変える

対象	まず教えること	事故になりやすい誤判断
人事	個人情報、委託先共有、退職者データ、権限見直し	退職後もデータや権限を残す
管理者権限保有者	特権アカウント、リモート接続、ログ、設定変更	共有 ID を使う、権限を戻さない
役員	なりすまし、急ぎ案件、機密資料、権限委任	社長案件 を口実に手順を飛ばす

ここで重要なのは、教育テーマを増やしすぎないことだ。一般社員に管理者向けの深い話をしても定着しないし、経理に一般的な phishing の話だけをしても BEC 対策には足りない。ひとり情シスは、誰が狙われやすく、どの判断を誤ると痛いかで内容を切る方がよい。

たとえば経理なら、怪しいリンクより先に 振込先口座の変更依頼をメール一本で確定しない を徹底した方が効果が高い。管理者なら、共有アカウントで作業しない 作業後に権限を戻す 外部ベンダーへ管理者権限を渡しっぱなしにしない が重要になる。教育は平等である必要はない。必要に応じて違ってよい。

## 委託先、協力会社、取引先も、セキュリティ運用の対象である

自社社員だけ教育しても、委託先が弱ければ入口は残る。委託先が自社アカウントを使う、共有フォルダへ入る、ファイル授受をする、リモート接続するなら、その相手はセキュリティ運用の対象である。

ここで大事なものは、契約条項だけで終わらせないことだ。第6章では責任分担や契約の話をしたが、第20章で扱いたいのは、日常運用として何を守ってもらうかである。最低限、次は一枚で決めたい。

項目	最低限決めること	残すもの
外部アカウント	個人単位で発行し、共有しない	発行記録、棚卸し
MFA	必須にし、例外は期限付き	設定記録、例外承認
接続条件	接続元、利用時間、接続先を必要範囲に絞る	接続ルール、承認記録
ファイル共有	承認済みツールだけを使う	共有ルール、授受記録
作業連絡	作業内容、連絡先、緊急時連絡先を明確にする	連絡先一覧、運用票
注意喚起	直近の偽ログイン、手口変更、報告先を共有する	配信記録
契約終了時	停止、返却、削除、アクセス廃止を実施する	停止記録、削除確認

これらは難しい統制ではない。だが、曖昧なままだと事故が起きやすい。委託先が自分の私用クラウドでファイルを受け取る。誰が接続したか分からない共有IDを使う。障害時の連絡先が営業担当しか分からない。こうした状態では、技術的に守っていても崩れやすい。

また、委託先が自社システムを触るなら、自社社員と同じ注意喚起が必要な場面もある。外部だから一切共有しないのではなく、必要なルールと連絡は渡す必要がある。

## サプライチェーン全体で考える防御

取引先やベンダーのリスクを見ると、大企業向けの重い監査を想像しやすい。だが、NIST SP 1305もCISAのSMB向けVendor SCRM資料も、もっと実務的である。中心は **supplier requirements** を定義し、伝え、確認することだ。

まず、ベンダーを同じ重さで見ない方がよい。NIST SP 1305 は、supplier criticality を次で見ている。

重要度を見る観点	何を見るか
業務重要性	その製品やサービスが止まると、どの業務が止まるか
データ機微性	個人情報、契約情報、機密資料を扱うか
システムアクセス	自社システムへの程度アクセスするか

この三つで **高 / 中 / 低** を決めるだけでも、要求水準を変えやすい。高重要度の委託先に、低重要度の SaaS と同じ確認しかしないのは危ない。

次に、重要度に応じて求めることを決める。

要求項目	最低限確認すること
契約条項	守るべきセキュリティ要求が契約へ入っているか
情報共有	障害、脆弱性、事故時に誰へどう連絡するか
SLA	監視、対応、通知の水準が決まっているか
脆弱性開示	製品やサービスの脆弱性情報を通知するか
構成把握	重要製品なら component inventory を持てるか
従業員管理	担当者の vetting や権限管理をしているか
証拠	自己宣誓、標準準拠、認証、点検結果などを出せるか
再委託と終了時	再委託条件、終了時の返却や削除が決まっているか

NIST SP 1305 は、supplier requirements を default contractual language や SLA へ落とし込むこと、情報共有ルールや reporting process を決めること、必要に応じて vulnerability disclosure、component inventory、employee vetting、security evidence を求めることを挙げている。CISA の SMB 向け template も、yes / no /

partial で答える質問票として使える。つまり、サプライチェーン防御は **有名ベンダーだから安心** でも **契約書があるから安心** でもない。要求、確認、連絡経路の三つが必要である。

また、日本でも取引先からのセキュリティ要求は現実のものになっている。IPA の 2025年5月27日公表の調査では、**新たな脅威や攻撃の手口を知り対策を社内共有する仕組み** は 37.9% にとどまり、**1割強の企業が取引先から情報セキュリティ対策の要請を受けている** とされている。要請内容では **秘密保持のための措置** が 79.6% と多い。第20章は社内教育の章であると同時に、取引の信頼を守る章でもある。

## 小さな会社で現実的に回るやり方

小さな会社で最初から大規模な教育プログラムを作る必要はない。むしろ、長い研修ほど続かない。IPA の **5分でできる！情報セキュリティポイント学習** のような短い無料教材を土台にして、自社で必要な話だけ足す方が現実的である。

最小構成なら、次のくらいから始めるとよい。

タイミング	やること
入社時	基本ルール、報告先、使ってよいツール、共有方法を説明する
月次	5分で読める注意喚起を流し、最近の手口を共有する
四半期	短い phishing / BEC 訓練を行い、結果を振り返る
半期	経理、人事、管理職、管理者向けの役割別教育を足す
契約前	委託先の重要度を決め、最低要求と連絡経路を確認する
契約中	外部アカウント、MFA、共有方法、緊急連絡先を棚卸しする
契約終了時	停止、返却、削除、共有解除を確認する
年次	教材、ルール、連絡先、訓練題材を見直す

また、教育は **やった** で終わらせず、実際に何が起きたかへつなげたい。怪しいメールが来た。取引先から突然共有リンクが届いた。ベンダーが新しい接続方法を求めてきた。こうした出来事を材料にして短く振り返る方が、抽象的な研修より定着しやすい。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員は入社時に基本ルールと報告先を教わる。一般社員には毎月短い注意喚起が流れ、四半期ごとに標的型メール訓練を行う。訓練結果はクリック率だけでなく難易度と報告率も見て、次回の題材を調整する。経理には口座変更依頼と送金依頼の確認ルールを別に周知し、管理者権限保有者には特権アカウント運用の注意を出す。委託先には個別アカウントと MFA を必須にし、ファイル授受と緊急連絡先を一枚にまとめて渡す。ベンダー重要度も **業務重要性 / データ機微性 / システムアクセス** で分類され、重要な委託先には報告ルールと SLA まで確認する。こうすると、教育、注意喚起、委託先ルールが一本線につながる。

崩れた時の例では、年1回の eラーニングは実施しているが、報告先を誰も覚えていない。標的型メール訓練はクリック者の一覧を出して終わり、題材の難易度も報告率も見えていない。経理も人事も一般社員と同じ教材だけを受けている。委託先は共用アカウントで接続し、ファイルは各自のやりやすい方法で受け渡している。ある日、取引先担当者になりすました口座変更依頼メールが届き、経理がそのまま処理する。後で見ると、確認電話もせず、報告窓口も使われていない。問題は **教育をしていなかった** ことではなく、教育が運用へつながっていなかったことである。

## 最低限ここまではやる

第20章の内容を一度に全部整えなくてもよい。だが、次の六つは早めに持ちたい。

1. 怪しいメールや不審な依頼の報告先と確認経路を明文化する
2. 年1回以外の短い注意喚起を定期化する
3. 標的型メール訓練では、難易度、クリック、報告をあわせて見る
4. 経理、人事、管理職、管理者向けの教育内容を分ける
5. 委託先向けに、アカウント、接続、共有、報告、終了時対応の最低限ルールを一枚にする
6. 重要なベンダーには、契約条項、報告ルール、SLA、証拠確認まで求める

この六つがあるだけで、セキュリティ教育はかなり実務になる。セキュリティ教育とは、研修を実施することではない。社員も委託先も怪しい時に止まり、既知の経路で確認し、すぐ報告できる運用を作ることである。

## 今日、今週、後でやること

今日やることは、不審メール、不審な送金依頼、不審な外部接続依頼が来た時の報告先と確認経路を、一枚で書き出すことである。ここが曖昧なら、教育しても動けない。

今週やることは、対象者を一般社員、経理、人事、管理職、管理者権限保有者、役員に分け、それぞれに何を教えるかを定めることである。あわせて、委託先向けの接続、共有、報告ルールを短く文書化したい。

後でやることは、標的型メール訓練の題材と結果の見方を整え、報告率や振り返りまで含めて回すことである。さらに、重要な委託先やベンダーについて、重要度と最低要求を整理したい。ここまでできると、セキュリティ教育は **受講した**かの管理から、**事故が起きにくくなる**運用へ変わる。

# 障害、インシデント、事業継続

有事の初動、連絡、復旧、継続を設計する。

第21章～第23章

## 第21章

## 障害対応とインシデント管理

---

障害が起きた時、最初に失われやすいのはサーバーや回線だけではない。状況把握もすぐ失われる。誰かは調べている。誰かは利用者から問い合わせを受けている。誰かはベンダーへ連絡している。だが、全体としては **何が起きているか 誰が見ているか どこまで分かっているか** が見えない。これが、ひとり情シスや小規模組織で最もつらい状態である。

第21章で扱うのは、この混乱を減らすための型である。深い攻撃対応は第22章で扱う。長時間停止や災害対応は第23章で扱う。第21章の中心は、その手前にある共通の incident management だ。障害かもしれないし、セキュリティ事故かもしれない。まだ原因は分からない。そういう時に、どう立ち上げ、どう連絡し、どう戻すかを整理する。

### 障害とセキュリティインシデントを切り分ける

第21章で最初に押さえないのは、incident の意味である。ServiceNow は ITIL の incident を、IT service の **unplanned interruption** または **reduction in the quality** と説明している。NIST SP 800-61r3 は **cybersecurity incident** を、機密性、完全性、可用性を実際に又は差し迫って損なう事象や、法令、security policy、procedure、acceptable use policy の violation 又は imminent threat と整理している。つまり、incident とは単なる問い合わせや不具合の別名ではない。通常運用を外れ、緊急対応として立ち上げるべき状態である。

ここで第16章との違いが出る。service request は、申請すれば進められる標準作業である。パスワード再設定、ソフト導入依頼、権限追加依頼はそちらに入る。incident は違う。メールが使えない、社内 Wi-Fi がつながらない、勤怠システムが遅い、SaaS が一部落ちている。このように、業務へ影響が出ており、急いで状況整理と復旧が必要なものが incident である。

最初の切り分けは、次の表で考えると分かりやすい。

区分	何か	例	まずやること
request	標準手順で処理できる依頼	パスワード再設定、権限追加、ソフト導入	通常キューで処理する
service incident	サービス停止や品質低下	メール停止、VPN 不安定、SaaS 遅延	起票して影響確認、復旧を優先する
security incident 寄り	悪性の可能性が残る incident	不審ログイン、設定改ざん疑い、data loss の可能性	第22章寄りへ寄せ、証跡保全を意識する

ただし、ここで難しいのが、障害とセキュリティインシデントの境界である。最初は単なる障害に見えても、実は不正アクセスや設定改ざんが原因かもしれない。CISA は Incident Response Playbook の対象を、confirmed malicious cyber activity だけでなく、悪性の可能性がまだ reasonably ruled out されていない状態も含めている。これは実務的である。原因未確定なら、早く **ただの障害** と決めつけない方がよい。

特に次のどれかがあるなら、security incident 寄りで扱いたい。

- 不審な認証や権限変更が見える
- data loss や情報改ざんの可能性がある
- 原因が直前変更では説明しにくい
- 取引先や外部から不正利用の連絡が来ている

この切り分けがあるだけで、初動の迷いはかなり減る。

## 初動は、認知、起票、指揮、影響確認、初報の順で行う

障害対応で崩れやすいのは、原因究明から入ってしまうことである。もちろん原因は知りたい。だが、最初の 10 分から 30 分で先にやるべきことは別にある。Atlassian の incident response flow でも、detect の後に raise a new incident、open comms、assess、send initial comms を置いている。ここから分かるのは、原因究明より前に、incident を立ち上げて人を集める必要があるということだ。

初動では、次の順で考えるとよい。

1. 認知する
2. incident を起票する
3. 指揮役を決める
4. 影響範囲と代替手段を確認する
5. 初報を出す

起票はとても重要である。豪華な仕組みは要らないが、一つの incident ticket は必要だ。Atlassian は every incident で Summary、Description、Severity、Faulty service、Affected products を埋め、issue key を全連絡の起点にしている。起票がないと、後から時系列がたどれず、誰がどこまで確認したかも分からなくなる。

incident ticket には、最低限これだけは欲しい。

項目	何を書くか
要約	何が起きているか。緊急度が分かる短い題名
影響	利用者や業務に何が起きているか

項目	何を書くか
severity	高 / 中 / 低 の仮置き
対象サービス	どのサービスやシステムが怪しいか
影響対象	全社、一部部門、一部顧客など
起点時刻	いつから起きているか
security 可能性	不審ログイン、改ざん、data loss の疑い有無
報告者連絡先	追加確認が必要な時の連絡先

指揮役も早く決めたい。NIST SP 800-61r3 は roles and responsibilities を文書化し、必要な authority を事前に与えるべきとしている。ひとり情シスの組織でも、直す人と全体を見る人を頭の中で分けるだけで違う。自分一人しかいないなら、自分がその二役を切り替えることになる。それでも、今は 調査モード なのか社内告知モード なのかを意識して分けた方が混乱しにくい。

影響確認では、Atlassian の assess の質問がそのまま使いやすい。少なくとも次を早く押さえたい。

確認項目	何を見るか
何が使えないか	どの機能や業務が止まっているか
何が見えているか	利用者は何を見ているか。エラーか、遅延か、無反応か
何人に影響するか	全社か、一部か、顧客影響があるか
いつからか	障害開始時刻、直前変更の有無
代替手段はあるか	別経路、手作業、別システムで回せるか
security / data loss の可能性	不正利用、改ざん、漏えい、削除の疑いがあるか

この段階で完全な正解は要らない。仮説でよい。大切なのは、何も見えていないまま黙らないことである。

severityも難しくしなくてよい。Atlassianはseverity matrixをcustomer impact基準で文書化し、チーム間で合意するべきとしている。小規模組織なら次の三段階で十分である。

severity	判断基準	初動の目安
高	全社停止、顧客影響大、代替手段なし	すぐ起票、すぐ初報、関係者招集
中	一部停止、代替手段あり、業務継続は可能	起票、影響確認、定期更新
低	限定影響、通常時間内対応でよい	通常時間内に調査、必要時だけ共有

## 重大障害では、一つの source of truth を作る

重大障害で混乱が増える最大の理由は、情報が散ることだ。メール、チャット、口頭、電話、ベンダー窓口。どこにも部分情報はあるのに、全体像がない。これを防ぐには、一つの source of truth を持つのが最も効く。

Atlassian は、incident communication で dedicated status page を primary communication solution とし、clear source of truth を持つべきだとしている。これは status page 製品の話だけではない。ひとり情シスの現場では、incident ticket、共有チャット、更新メモのどれか一つを **ここを見れば今の状況が分かる場所** と決めるだけでも効果がある。

重大障害では、少なくとも次を一か所に集めたい。

集めるもの	何を書くか
何が起きているか	いまの仮説と現象

集めるもの	何を書くか
影響範囲	誰が困っているか、どの業務が止まっているか
現在ステータス	investigating、mitigating、restoring など
次の更新予定	次回更新時刻
担当	誰が指揮し、誰が技術調査し、誰が連絡するか
バンダー対応	問い合わせ番号、回答待ち、次アクション

NIST SP 800-61r3 は、incident communication を四つに分けている。これを分けて考えると、連絡先が混ざりにくい。

連絡の種類	誰向けか	何を含めるか
coordination	incident 対応者、社内関係者、委託先	いま何をしているか、誰が担当か、次の作業
notification	顧客、従業員、取引先、規制当局など	incident の発生、影響、必要な行動、法令や契約に基づく通知
public communication	一般公開、メディア、外部向け案内	公表してよい範囲の状況説明
information sharing	ISAC 等、関係組織	threat 情報、観測した TTP、被害拡大防止に役立つ情報

連絡も早く出した方がよい。Atlassian Support は **communicate early** **communicate often stay consistent across channels** を勧めている。NIST も **what must be reported to whom and at what times** を事前手順へ含めるべきとしている。つまり、まだ調査中でも、起きていることと影響の仮説を短く伝え、次の更新時刻を示した方がよい。

初報と更新は、たとえば次の型で十分である。

文面	最低限入れること
初報	何の障害か、誰に影響があるか、調査中か 回避策があるか、次回更新時刻
更新	1-2文の要約、Current Status 2-4項目、Next Steps 2-4項目、次回更新時刻

ここで避けたいのは、詳細が分かるまで黙ることだ。利用者は **まだ直っていないこと** より **何も分からないこと** に強い不安を感じる。初報は安心させるためというより、混乱を広げないために必要である。

## 復旧は、原因究明より先に業務影響を下げる

incident management の goal は、ServiceNow が説明する通り、normal service operation をできるだけ早く回復し business impact を最小化することである。つまり、最初の目的は美しい原因分析ではなく、業務影響を下げることだ。

ここでは、次の違いを分けて考えると判断しやすい。

手段	何をするか	向いている場面	注意点
workaround	完全には直っていませんが業務を回す	代替経路、手作業、別手段で継続できる時	恒久対策と混同しない
rollback	直前変更を元へ戻す	障害開始と change の時刻が近い時	security incident 可能性がある時は慎重に
failover	別系統や予備系へ切り替える	冗長構成や代替環境がある時	切替後の整合性を確認する
restoration	通常状態へ戻す	原因対処後、通常運用へ復帰する時	一度戻っただけで close しない

この順番が大切である。たとえば、社内メールが止まった時に、根本原因がまだ分からなくても、チャットや電話で代替連絡を案内すれば業務影響は下がる。SaaS 障害なら、ベンダー障害情報を確認しつつ、代替手段を先に出せる。原因は後で詰めればよい。

また、Atlassian は incident の start time が change と対応しているかを見て、必要なら rollback を検討している。つまり、**直前変更の有無** は初動確認項目でもある。

一方で、原因未確定のまま危険な変更を重ねるのは避けたい。特に、悪性の可能性が消えていない時はなおさらである。**とりあえず再起動** **とりあえず消す** は、場合によっては状況を悪化させる。第22章へ渡す条件に触れそうなら、むやみに証跡を壊さない方がよい。

## 運用に戻すまでの判断を急がない

一度つながった、一度ログインできた、それだけで close しない方がよい。incident がつらいのは、復旧したと思ったら再発することだ。だから、運用に戻すまでにはもう一段階必要である。

Atlassian も、incident is resolved を **affected service resumes functioning in its usual way** としている。つまり、単に一瞬戻っただけでなく、通常の状態へ戻ったかを見る必要がある。

close 前には、少なくとも次を確認したい。

確認項目	何を見るか
影響収束	全社か一部か、影響範囲は本当に縮んだか
監視安定	監視値、エラー件数、利用者報告は落ち着いたか
代替手段解除	workaround や failover を戻してよいか

確認項目	何を見るか
未解決課題	恒久対策、監視改善、ベンダー回答待ちが残っていないか
security可能性	不正利用、改ざん、data lossの可能性は消えたか
次アクション	review、problem、changeへ何を送るか

また、close の前には、何を保留して次の課題へ送るかを明確にした方がよい。恒久対策がまだ、監視改善がまだ、ベンダー回答待ち、レビュー未実施。こうしたものを曖昧にして close すると、後で何も残らない。

## 事後レビューは、責任追及でなく改善に使う

incident が終わると、そのまま次の仕事へ流れやすい。だが、振り返りをしないと、次も同じ混乱を繰り返す。Atlassian の postmortem は、impact、actions taken、root cause、follow-up actions を含む written record と整理している。さらに、責任追及を目的にしない postmortem を強調し、timeline、causal chain、mitigations を個人ではなく system と process で捉えるとしている。NIST SP 800-61r3 も after-action report に、incident 自体、response / recovery actions、lessons learned を含めるとしている。

ここは第21章の重要点である。振り返りは犯人探しではない。誰がミスしたかより、なぜそのミスが起きても止まらなかったか なぜ気づくのが遅れたか なぜ連絡が遅れたかを見る方が価値がある。

レビューでは、最低限次を残したい。

項目	何を書くか
何が起きたか	incidentの要約
impact	誰にどんな影響があったか
timeline	いつ何をしたか

項目	何を書くか
response / recovery	実施した workaround、rollback、restore
root cause / contributing factors	根本原因と寄与要因
lessons learned	何を学び、何を変えるか
follow-up actions	何を直すか、誰がいつまでにやるか

ここで時系列があると強い。Atlassian は、incident 中の dedicated chatroom へ observations、changes、decisions をその場で書いておくことが、後の timeline と postmortem に役立つとしている。障害中にメモしておけば、振り返りの質がかなり上がる。

また、振り返りの出力は改善へつながらなければ意味がない。監視追加、権限見直し、切り戻し手順改善、ベンダー連絡先整理、初報テンプレート修正。こうした改善項目を担当者と期限付きで残したい。

## 小さな会社で現実的に回るやり方

小さな会社では、NOC や専任の incident commander はいないことが多い。だが、それでも最低限の型は作れる。現実的には、次の六つで十分始められる。

持っておくもの	役割
共有 ticket	起票、時系列、担当、判断の起点にする
一つのチャット部屋	連絡とメモを散らさない
初報テンプレート	何を最初に伝えるかを固定する
更新テンプレート	続報を同じ型で出す
review テンプレート	impact、timeline、改善項目を残す
外部相談先一覧	自力で抱え込まないための連絡先にする

これだけでも、かなり違う。ツールの名前は何でもよい。大切なのは、**毎回ゼロから考えない** ことだ。

また、小規模組織では外部相談先を持っておく価値が高い。IPA の **サイバーセキュリティ 相談・届出窓口一覧** は、企業組織向けに初動対応相談を受け付けており、被害有無の判断、有効な応急処置、専門業者一覧、他の相談・報告先の紹介をしている。完全に自力で抱え込まない方がよい。

## 通常時の例と、崩れた時の例

通常時の例では、社内メールが使えないという報告が来た時点で、incident ticket を起票する。要約、影響、severity、対象サービス、起点時刻、security 可能性を入れ、影響確認の質問で **全社か、一部か** を確認する。チャット部屋を一つ開き、担当と連絡役を分ける。初報では **メール障害を調査中であること 影響範囲 代替連絡手段 次回更新時刻** を出す。ベンダー問い合わせも ticket に紐付け、時系列で残す。復旧後は monitoring を挟み、review で impact、timeline、原因、改善項目を残す。結果として、直すだけでなく、周囲も状況を追える。

崩れた時の例では、利用者から **メールが変だ** と個別に連絡が来るが、誰も起票しない。ある人は再起動し、ある人はベンダーへ電話し、ある人は口頭で **直りそうらしい** と言う。社内告知は出ず、営業は顧客へ返信できないまま待つ。途中でもしかして **乗っ取りかもしれない** という話が出るが、誰もその判断を引き取らない。夕方に一度使えるようになるが、原因も時系列も残っていない。翌週また似た障害が起きても、前回の教訓が使えない。問題は、技術力ではなく型がなかったことである。

## 最低限ここまではやる

第21章の内容を一度に全部整えなくてもよい。だが、次の六つは早めに持ちたい。

1. request、service incident、security incident 寄りの三分を持つ
2. 初動で起票し、指揮役と severity を決める
3. 初報と更新のテンプレートを持つ
4. 重大障害では一つの source of truth とタイムラインを残す
5. close 前に、監視安定、影響収束、security 可能性消失を確認する
6. review で改善項目を担当者と期限付きで残す

この六つがあるだけで、障害対応はかなり安定する。障害対応とは、慌てて直すことではない。incident として立ち上げ、影響を把握し、連絡を出し、復旧し、記録を残し、次に同じ混乱を起こしにくくすることである。

## 今日、今週、後でやること

今日やることは、incident として立ち上げる条件、severity 三段階、初報で必ず書く項目を一枚にすることである。これがないと、毎回 **これは大ごとか** から迷う。

今週やることは、共有チケット、チャット部屋、更新メモの置き場を決め、障害時にそこへ情報を集約する練習をすることである。あわせて、security 可能性がある時に第22章寄りへ切り替える条件と、外部相談先も決めたい。

後でやることは、review の型を整え、実際の障害一件で短く回してみることである。ここまでできると、障害対応は属人的な火消しから、改善につながるインシデント管理へ変わる。

## 第22章

# ランサムウェア、情報漏えい、不正アクセス対応

重大セキュリティ事故では、最初にやるべきことが通常の障害対応と少し違う。障害なら、できるだけ早く戻したい。だが、ランサムウェア、情報漏えい、不正アクセスでは、急いで戻そうとした動きそのものが被害拡大や証拠消失につながることもある。再起動してしまう。感染端末を社内ネットワークへつないだままにする。乗っ取られたアカウントのパスワードだけを変えて安心する。漏えい範囲が固まるまで黙ってしまう。こうした初動ミスは、後から取り返しにくい。

第21章では incident management の共通型を扱った。第22章で扱うのは、その中でも **直す前に止める** **消す前に残す** **確定前でも報告判断を始める** 必要がある事故である。第19章の法制度説明や第23章の BCP へ広げすぎず、ここでは重大セキュリティ事故の初動と連携に絞って整理する。

最初の30分から数時間で見たい違いは、次の表で整理しやすい。

類型	まず見ること	最初にやること	やってはいけないこと
ランサムウェア	端末だけか、共有先や管理者経路まで広がっているか	隔離、封じ込め、証拠保全	再起動、再接続、バックアップ接続
漏えい疑い	何のデータか、何人分か、外部到達が続いているか	事実確認と報告判断を並行で始める	全部分かるまで待つ

類型	まず見ること	最初にやること	やってはいけないこと
アカウント侵害 / BEC	どのアカウント、どの連携、どの送金や共有に影響したか	user、session、token、app、device を止める	パスワード変更だけで終える

## ランサムウェアの被害像を甘く見ない

ランサムウェアという言葉を知ると、**ファイルが暗号化される攻撃** を想像しやすい。もちろんそれは正しい。だが、今の実務ではそれだけでは足りない。CISA の #StopRansomware Guide は、ransomware and data extortion としてガイドを整理している。JPCERT/CC も **侵入型ランサムウェア攻撃** という言い方で、攻撃者が内部ネットワークへ侵入した後に情報窃取や暗号化を行う型を区別している。つまり、ランサムウェアは **壊れた PC の修理** ではなく、**侵入、窃取、横展開、脅迫を含む事故** と見た方がよい。

この見方に変えると、最初に考えるべきことも変わる。暗号化された端末が一台見つかった時に、本当に一台だけなのか。共有フォルダはどうか。管理者アカウントは使われていないか。VPN や RDP はどうか。バックアップやクラウドストレージへ触られていないか。ここまで見る必要がある。

見る対象	なぜ見るか
影響端末	端末単体の故障か、複数感染かを切り分けるため
共有フォルダ、NAS	暗号化や削除が広がっていないかを見るため
管理者アカウント	横展開や権限昇格を疑うため
VPN、RDP、遠隔管理	侵入経路や継続接続を疑うため

見る対象	なぜ見るか
バックアップ	最後の復旧手段が汚染されていないかを見るため
クラウド同期、共有ストレージ	同期経由で被害が広がっていないかを見るため

また、身代金要求も単純ではない。CISA と FBI の注意喚起は、支払いが復旧や情報非公開を保証しないことを明確にしている。したがって、**払えば戻るかもしれない**を前提に初動を組んではいけない。第22章で重要なのは、支払うかどうか以前に、被害拡大を止め、証拠を残し、経営判断に必要な材料を集めることである。

## 感染時の初動は、隔離、封じ込め、証拠保全の順で考える

ランサムウェアらしき症状が出た時、最初の目的は **直すこと** ではない。まずは広がらないようにすることである。CISA の Ransomware Response Checklist は、影響を受けたシステムを直ちに隔離することを最初に置いている。複数システムや複数サブネットに広がっているなら、スイッチ単位でネットワークを落とす判断にも触れている。つまり、**一台ずつ様子を見る** より、広がりを止める方が先である。

初動では、少なくとも次の順で考えたい。

1. 影響端末、共有先、接続元を隔離する
2. どこまで広がっているかを粗く把握する
3. 証拠を残す
4. 重要システムの復旧優先順位を決める
5. 外部支援を呼ぶ

隔離は、端末だけの話ではない。影響を受けたユーザーアカウント、共有フォルダ、VPN 接続、リモート管理経路、クラウド同期も視野に入れる必要がある。Wi-Fi を切る、LAN を抜く、対象 VLAN を切り離す。こうした動きは、復旧作業より先である。

隔離対象	最低限やること
端末	LAN を抜く、Wi-Fi を切る、必要なら VLAN から外す
アカウント	一時無効化、管理者権限停止、共有認証の確認
共有先	書き込み停止、共有解除、接続元確認
VPN / RDP	接続停止、接続元 IP と利用アカウント確認
クラウド同期	同期停止、共有リンク確認、同期先汚染の確認
バックアップ	直ちに再接続せず、汚染範囲確認まで保護する

ここで気をつけたいのが、電源断を乱発しないことだ。CISA は、ネットワークから切り離せない時の最後の手段として power down に触れている。電源断は拡大防止に効くことがある一方で、揮発性メモリ上の証跡や実行中の痕跡を失うことがある。したがって、**落とした方がよさそう** で反射的に切るのではなく、切り離し不能な時の手段として考えた方がよい。

証跡保全も早い方がよい。CISA は、sample of affected devices に対する system image と memory capture を勧めている。ひとり情シスの現場で完璧なフォレンジックを自力でやる必要はないが、最低限でも次は残したい。

残すもの	理由
発見時刻	後の timeline とログ照合に必要なだから
端末名、ユーザー名	影響範囲と同一資格情報の追跡に必要なだから
ransom note	攻撃グループや要求内容の手掛かりになるから

残すもの	理由
画面表示、拡張子変更、削除痕跡	何が起きたかを専門家へ伝えやすいから
影響共有先、同期先	横展開範囲の確認に必要なだから
直前の不審ログイン、権限変更	侵入経路や lateral movement の手掛かりになるから
sample image / memory capture	専門調査へつなぐ材料になるから

ここでは完璧なフォレンジックを目指さなくてよい。ひとり情シスなら、まず時刻、端末名、ユーザー名、画面表示、ログの場所を押さえるだけでも価値がある。JPCERT/CC や専門ベンダーへ相談する時も、何がいつ起きたかがあるだけで進めやすい。

反対に、避けたい行動もある。とりあえず再起動する。とりあえず初期化する。とりあえずバックアップをつなぐ。とりあえず共有フォルダへ再接続する。これらは、状況が分からないまま行くと悪化しやすい。特にバックアップは、汚染範囲を確かめる前に触ると、最後の復旧手段まで巻き込みかねない。

また、攻撃者が組織の対応を監視している可能性も考えたい。CISA は out-of-band communication を勧めている。重大なランサムウェアでは、普段使っているメールやチャットが見られているかもしれない。少なくとも初期の意思決定では、電話や別回線を使う前提を持った方がよい。

## 情報漏えい疑いでは、事実確認と報告判断を並行で進める

情報漏えい疑いの難しさは、発生直後に全部は分からないことだ。誤送信かもしれない。外部共有設定ミスかもしれない。乗っ取られたアカウントからダウンロードされたかもしれない。ランサムウェアで窃取された可能性もある。こういう時に **全部分かってから考える** と遅れやすい。第22章では、事実確認と報告判断を並行で進める方がよい。

最初に整理したいのは、次の点である。

確認項目	何を見るか
何のデータか	個人データ、営業秘密、契約情報、認証情報など
個人データに当たるか	第19章の整理で個人データか確認する
何人分くらいか	件数感を早く持つ
確認レベル	閲覧済み、取得済み、外部送信済み、又はそのおそれか
継続性	いまも外部からアクセス可能か、共有が開いているか
起因	自社起因か、委託先起因か、乗っ取り起因か

ここでの **個人データ** の整理自体は第19章で扱った。第22章で強調したいのは、事故時にはこの確認を急ぐ必要があるという点である。個人情報保護委員会は、個人の権利利益を害するおそれがある事態として、要配慮個人情報、財産的被害のおそれ、不正目的のおそれ、1,000人超を挙げている。そして速報は **速やかに（概ね3～5日以内）** としている。つまり、調査に一週間かけてから考える、という時間感覚ではない。

個人情報保護委員会への報告判断は、次の表で見ると分かりやすい。

類型	例	速報	確報
要配慮個人情報	健康情報、診療情報、病歴など	概ね3～5日以内	30日以内
財産的被害のおそれ	決済情報、口座情報、認証情報など	概ね3～5日以内	30日以内
不正の目的のおそれ	不正アクセス、ランサムウェア窃取、悪意ある持出しなど	概ね3～5日以内	60日以内
1,000人超	1,000人を超える個人データ	概ね3～5日以内	30日以内

確報にも期限がある。個人情報保護委員会の整理では、通常は 30 日以内、不正の目的をもって行われたおそれがある場合は 60 日以内である。ランサムウェアや不正アクセスが絡むなら、この **不正の目的** の類型に入りうる。したがって、漏えい疑いの時点で、個人情報保護委員会への速報、確報、本人通知の可否を誰が判断するかを早く立ち上げた方がよい。

ここで誤解しやすいのが、公表である。個人情報保護委員会 FAQ は、公表が一律に義務付けられているわけではない一方、事案の内容に応じて望ましい場合があるとしている。また、公表が二次被害拡大につながる可能性があるなら、公表しないこともありうる。つまり、**漏えいらしいから必ず公開** でも **公開義務はないから黙る** でもなく、本人通知、二次被害、経営判断、広報判断を合わせて考える必要がある。

報告対象に該当するかまだ曖昧でも、PPC へ報告自体は可能である。つまり、**対象かどうか分かるまで一切動けない** ではない。また、2025年10月1日以降、ランサムウェア事案による個人データの漏えい等又はそのおそれでは、共通様式での報告も可能になっている。こうした制度面を知っているだけで、事故時の迷いは減る。

委託先起因の事故でも、委託元側の対応を止めない方がよい。個人情報保護委員会 FAQ Q5-9 でも、委託元は委託先における個人データの取扱状況を把握する必要があるとしている。したがって、SaaS 事業者、業務委託先、BPO 先で漏えい等が起きた場合も、ベンダーからの正式レポート待ちで止まるのではなく、自社として **何のデータが対象か** **本人通知が必要か** **社内で誰に連絡するか** を先に動かす必要がある。

## 不正アクセス、乗っ取り、なりすましでは、アカウントだけでなく連携も止める

不正アクセスや乗っ取り対応で最も多い誤解は、パスワード変更で終わってしまうことだ。実際には、それでは足りないことが多い。Microsoft Entra の緊急アクセス剥奪資料は、ユーザー無効化、refresh token の失効、端末無効化を並べている。つまり、止めるべきものは **パスワード** だけではなく、**アカウントセッショントークン** **端末** である。

まずやりたいのは、侵害された可能性のあるアカウントを止めることだ。必要に応じて一时无効化する。次に、既存セッションや refresh token を失効させる。さらに、そのユーザーが使っていた管理端末、メールクライアント、モバイル端末も見直す。管理者アカウントなら、権限棚卸しや緊急パスワード変更、MFA 再登録も要る。

止める対象	最低限やること	理由
user	一时无効化、必要なら password reset	新規ログインを止めるため
refresh token / session	revoke、sign-out、session cleanup	既存ログインが残ることがあるため
device	managed device を disable / wipe	端末上の認証状態や locally stored data を切るため
OAuth app	malicious app を disable / block	app 経由の再侵入を止めるため
mail rule / forwarding	転送、受信ルール、委任設定を確認して止める	なりすまし継続を防ぐため
sharing / API token	外部共有、API token、連携設定を確認して止める	data exfiltration を止めるため

OAuth 悪用も見落としやすい。Microsoft の consent phishing 資料では、悪意あるアプリを無効化しても、既存 access token は期限まで有効な場合があるとしている。つまり、怪しいアプリ同意を見つけて **削除したから終わり** ではない。アプリ停止に加えて、ユーザー側のセッション失効やアカウント制御も必要になる。Microsoft Entra の緊急アクセス剥奪資料でも、disable user、revoke refresh tokens、disable devices を並べている。

Google Workspace でも同じ発想が必要である。Google は、特定 app を Blocked にしたり、必要なら **Block all third-party API access** によって sign-in scopes を含む広い遮断ができると示している。緊急時には、業務影響と引き換えでも広めに止める判断が必要になることがある。

また、乗っ取りの痕跡は認証以外にも残る。最低限でも、次を確認したい。

- メール転送設定
- 受信ルール、仕分けルール
- 委任設定、共有設定
- 外部共有リンク
- OAuth 同意アプリ
- API トークンや連携設定

BEC もここに入る。取引先や経営者になりすましたメールで口座変更や送金を促す型は、単なる迷惑メールではない。IPA は BEC を独立した脅威として扱っているし、FBI は被害時に金融機関へ即連絡し、送金先金融機関への連絡を依頼するよう案内している。実務でもこれは重要である。送金が絡む時は、**社内確認が終わってから** では遅い。まず自社の金融機関へ連絡し、着金先への停止、組戻し、照会を急ぐ。その上で、警察、法務、経営へつなぐ方がよい。

## 法務、経営、広報、外部機関との連携を遅らせない

重大セキュリティ事故では、技術対応だけで閉じない。誰にどこまで連絡するかが、その後を大きく左右する。ここで遅れやすいのは、**まだ確定していないから上げない**という判断である。だが、重大事故では確定前でも初報が必要だ。

経営には、少なくとも次を早く渡したい。

- 何が起きているか
- どの業務に影響があるか
- 何がまだ不明か
- 次回更新時刻
- 外部連携が必要か

法務や個人情報保護の担当には、個人データ該当性、本人通知、公表、証跡保全、契約上の通知義務を早めに見てもらおう。広報には、問い合わせが来た時の一次回答と、公表方針の整理が必要になる。委託先や SaaS 事業者には、ログ保全、セッション無効化、共有停止、追加調査を依頼する。第21章で述べた通り、一つのタイムライン、一つの source of truth はここでも重要である。

連携先は、次の表で整理すると漏れにくい。

相手	何を頼むか
経営	影響認識、対外判断、優先順位付け
法務 / 個人情報保護担当	報告対象、本人通知、公表、契約通知の判断
広報	一次回答、公表文面、問い合わせ整理
委託先 / SaaS	ログ保全、共有停止、session cleanup、追加調査
JPCERT/CC	初動方針、被害箇所特定、調査支援

相手	何を頼むか
IPA	被害有無判断、応急処置、相談先紹介、届出
金融機関	送金停止、送金先金融機関への連絡、照会
警察	刑事被害相談、証跡提供、捜査連携

外部機関への連携も、抱え込まない方がよい。IPA は **各種インシデント発生時の初動対応に関する相談** を受け付けている。JPCERT/CC も、インシデント初動対応のサポートや被害箇所特定の支援を受け付けている。ひとり情シスが一人で判断し切れない時に、こうした窓口は現実には使える。

もちろん、全てを外へ出せばよいわけではない。法的な論点、対外公表、被害者連絡は、社内責任者と合わせて決める必要がある。ただし、**全部自分で整理してから** という順番にすると遅れやすい。重大事故では、社内外の関係者を早く立ち上げた方が、結局は速い。

## 小さな会社で現実的に回るやり方

小さな会社で、SOC やフォレンジックチームを前提にする必要はない。だが、重大事故用の最小セットは持った方がよい。現実的には、次の五つがあるだけでもかなり違う。

持つておくもの	役割
最初の30分チェックリスト	隔離、保全、連絡の順番を迷わないため
緊急連絡先一覧	経営、法務、広報、ベンダー、金融機関へすぐつなぐため
out-of-band 連絡手段	普段のメールやチャットが使えない時に備えるため
PPC 報告条件と期限の一枚メモ	速報、確報、本人通知判断を遅らせないため

持っておくもの	役割
相談先一覧	JPCERT/CC、IPA、外部専門家へつなぐため

これらは、大きな仕組みではない。だが、事故時に **何からやるか 誰へ上げるか** が分かるだけで、初動の質は変わる。

また、復旧の優先順位も平時に決めておきたい。人事給与、会計、受発注、社内認証、メール、共有ファイル。どれから戻すかが決まっていないと、事故時に声の大きい要望へ引っ張られやすい。第23章の BCP ほど大げさでなくても、重大事故向けの簡単な優先順位表は持っておいた方がよい。

## 通常時の例と、崩れた時の例

通常時の例では、経理担当の PC に ransom note が表示された時点で、端末をネットワークから外す。共有フォルダとその担当アカウントの利用状況を確認し、同じ認証情報を使う経路も一時的に止める。incident ticket を立て、時刻、端末名、ユーザー名、画面表示、影響範囲を記録する。普段のチャットが見られている可能性も考え、別経路で関係者を集める。JPCERT/CC や専門ベンダーへ相談しつつ、バックアップや重要システムの汚染範囲を確認する。人事フォルダへのアクセス痕跡が見えた時点で、個人データ漏えい等のおそれとして社内の法務、個人情報保護担当、経営へ上げ、個人情報保護委員会への速報判断を始める。結果として、復旧より先に被害拡大防止と報告判断が立ち上がる。

別の通常時の例では、営業担当の Microsoft 365 アカウントから不審な送信が見つかる。まずアカウントを無効化し、refresh token を失効させ、利用端末も止める。同時に、メール転送、受信ルール、委任設定、OAuth 同意を確認する。怪しいアプリ同意が見つかったため、そのアプリを無効化し、影響ユーザーのセッ

ションも切る。取引先へ偽請求書メールが出ていたため、営業部門、経理、法務へ連絡し、入金や送金に影響が出ていないかを確認する。こうすると、**ログインを止めたら終わり**で済まない実務が一本線で見える。

崩れた時の例では、経理 PC が暗号化されたが、とりあえず再起動する。社内 LAN にはつながったままで、ファイルサーバーにもアクセスし直す。バックアップ NAS も同じネットワーク上にあり、そのまま接続を続ける。社内への連絡は**一部 PC が不調です**程度で、セキュリティ事故として立ち上がらない。数時間後、共有フォルダと別部署端末にも被害が広がる。後から見ると、VPN アカウントと管理者権限の悪用もあったが、初動でログや画面記録が残っていない。人事データの流出可能性も出たが、誰も個人情報保護委員会報告の期限を把握しておらず、速報が遅れる。問題は、技術知識が足りなかったことより、重大事故の型がなかったことである。

## 最低限ここまではやる

第22章の内容を一度に全部整えなくてもよい。だが、次の六つは早めに持ちたい。

1. ランサムウェアや乗っ取り時に、まず隔離する対象と out-of-band 連絡手段を決める
2. 再起動や初期化の前に、時刻、端末、アカウント、画面、ログ、sample imageを残す
3. 個人データ漏えい等の報告条件と期限を一枚で見られるようにする
4. 乗っ取り時に、user、session、refresh token、device、OAuth 連携を止める手順を確認する
5. BEC や誤送金時の金融機関連絡手順を持つ
6. 経営、法務、広報、ベンダー、JPCERT/CC、IPA の連絡網を持つ

この六つがあるだけで、重大セキュリティ事故の初動はかなり変わる。重大セキュリティ事故では、直すことより先に、被害拡大を止め、証拠を残し、報告判断を始め、社内外の連携を立ち上げる必要がある。

## 今日、今週、後でやること

今日やることは、ランサムウェア、乗っ取り、漏えい疑いの三つについて、最初の 30 分でやることを一枚にまとめることである。ここがないと、事故時に毎回ゼロから迷う。

今週やることは、個人情報保護委員会報告の条件と期限、2025年10月1日以降のランサムウェア共通様式、金融機関や委託先を含む緊急連絡先、普段のチャットが使えない時の連絡手段を整理することである。あわせて、Microsoft 365 や Google Workspace で、session revoke や app block を実際にどこで行うか確認しておきたい。

後でやることは、ランサムウェア、OAuth 悪用、誤送信や外部共有ミス、BEC を題材にした机上演習を短く回すことである。ここまでできると、第22章の内容は知識ではなく、重大セキュリティ事故の初動として使える形になる。

## 第23章

## BCP、災害対応、復旧訓練

---

BCP という言葉を聞くと、分厚い文書や様式を思い浮かべる人が多い。だが、ひとり情シスの現場で本当に必要なのは、立派な冊子ではない。地震でオフィスへ入れない。停電でサーバー室が止まる。回線が半日落ちる。取引先からの供給が止まる。感染症で出社人数が一気に減る。こういう時に、何を先に守り、どこまで止まってよくて、戻るまで何で回すかを決めておくことだ。これが BCP の本体である。

第21章では障害対応と incident management の共通型を扱い、第22章では重大セキュリティ事故の初動を扱った。第23章で扱うのは、その先で **長く止まりうる時に、どう持ちこたえて戻すか** である。ここでは、重要業務、復旧優先順位、代替手段、訓練に絞って整理する。

### 重要業務と business impact を先に決める

BCP が形骸化する最大の理由は、システム一覧から考え始めることだ。サーバー、NAS、SaaS、回線、端末。もちろんそれらは大事である。だが、本当に先に決めるべきなのは、何の業務をどれだけ止めると会社が痛いからである。

Ready.gov は、BIA を **time sensitive or critical business processes** が止まった時の影響を見積もり、**recovery strategies** に必要な情報を集めるものとして整理している。NIST の BIA 定義も同じで、業務機能と中断影響を分析するプロセスと見る。つまり、出発点は IT 資産ではなく、事業である。

ここで最初に整理したいのは、次のような問いである。

- 何の業務が止まると売上や受注に直撃するか
- 何の業務が止まると法令、契約、給与、税務で問題になるか
- 何の業務が止まると顧客対応が崩れるか
- 何の業務は翌日や翌週でも許容できるか

この問いを、最低限次の表にすると、重要業務と影響がぶれにくい。

項目	何を書くか
業務名	受注、問い合わせ対応、給与、会計締めなど
止まると困ること	売上停止、契約違反、法令遅延、顧客離れなど
影響が表面化する時点	1時間後、当日中、翌営業日、月末など
許容停止時間	どこまで止められるか
暫定運用	手作業、別拠点、テレワーク、限定提供など
外部影響	顧客、取引先、規制、委託先への影響

この問いに答えると、優先順位が見え始める。たとえば、受発注、顧客対応、会計締め、人事給与、勤怠、社内認証。会社によって違うが、全部が同じ重さではない。

内閣府の事業継続ガイドラインも、重要製品、重要業務を絞り込み、その復旧時間や復旧水準を考える流れを取っている。つまり、BCP とは **全部止めない計画**ではなく、**先に戻すものを決める計画**である。

また、影響は技術だけでは測れない。Ready.gov が挙げるように、損失は売上減少、追加費用、契約違反、顧客離れ、規制上の問題として現れる。ひとり情シスとしては、業務部門や経営と会話しながら、**何時間止まると何が困るか** を日本語で書けるようにした方がよい。

また、想定する中断原因も、停電だけでは足りない。Ready.gov の BIA は、次のようなシナリオを並べている。

中断シナリオ	なぜ入れるか
建物損傷	物理的に使えない状況を考えるため
入館不可	建物は無事でも入れない事態があるため
設備故障	サーバー、空調、電源、機器障害を含むため
電力・水・通信停止	電力、水、通信の停止を考えるため
仕入先停止	仕入れ、配送、外注停止が業務を止めるため
IT 損傷、データ消失	アプリ、データ、回線、認証停止を含むため
重要要員の欠勤	人が足りず業務が回らない事態があるため

## RTO、RPO、復旧水準を業務から逆算して決める

重要業務が見えたら、次は復旧目標を決める。ここで出てくるのが RTO と RPO である。難しそうに見えるが、意味は単純だ。RTO は **どれだけ止まれるか**、RPO は **どこまで巻き戻せるか** である。内閣府ガイドラインでは、これに加えて **どの水準まで戻すか** を目標復旧レベルとして扱っている。

NIST の定義では、RTO は mission や business process に悪影響が出る前に recovery phase にいられる全体時間である。RPO は、outage 後にデータをどの時点まで戻す必要があるかである。これをひとり情シス向けに言い換えると、次のようになる。

- RTO
  - その業務は何時間、何日止まると痛すぎるか

- RPO
  - データは昨日まで戻ればよいのか、1時間前まで必要か
- 復旧水準
  - 完全復旧が必要か、まずは50パーセントの運転でもよいか

章の中では、三つを一緒に見た方が実務へ落ちやすい。

項目	何を決めるか	誤解しやすい点
RTO	何時間、何日で再開したいか	完全復旧完了時刻とは限らない
RPO	どこまでのデータ巻き戻りを許容するか	バックアップがあるだけでは満たさない
復旧水準	どの水準で再開ならよいか	最初から100パーセント運転でなくてよい

ここで大事なのは、目標を願望で置かないことだ。内閣府ガイドラインは、目標復旧時間と目標復旧レベルは、講じた対策により達成可能であり、対外的に説明できるものでなければならないとしている。つまり、**1時間で全部戻す**と書くだけでは意味がない。本当にその回線、バックアップ、代替端末、要員、ベンダー体制でできるのかを見なければならない。

したがって、RTOや復旧水準を書く時は、同時に達成条件も見たい。

確認対象	何を見るか
回線	主回線断でも代替回線でつなげるか
認証	在宅や別拠点からログインできるか
端末	予備端末やBYOD方針で人数をまかなえるか
データ	RPOに見合うバックアップと復元時間があるか
要員	代行できる人、判断者、外注先がいるか

確認対象	何を見るか
ベンダー	緊急時の連絡先、支援時間、復旧支援範囲が明確か

また、RTO と RPO はシステム単位より業務単位で置く方が分かりやすい。たとえば、受注処理は4時間以内に再開したい、給与計算は翌営業日まででよい、ファイル共有は当日中、会議室予約は翌週でもよい。この粒度で整理すると、後でIT側の復旧順へつなげやすい。

## 現地復旧だけでなく、代替戦略を持つ

BCP で最も危ういのは、元の場所が戻れば何とかなる 前提で止まることである。だが、実際には戻らない時間がある。オフィスへ入れない。停電が長引く。主回線が落ちる。サーバー室へ入れない。だから、現地復旧だけでなく代替戦略を持つ必要がある。

内閣府ガイドラインは、現地復旧戦略だけでなく、代替拠点、OEM や提携先活用、ICT ツール、テレワークの活用を含む代替戦略を示している。つまり、BCPとは元に戻す方法だけでなく、戻るまで回す方法を定めることでもある。

代替戦略は、大げさでなくてよい。小さな会社なら、たとえば次のようなものがある。

- オフィスへ入れない時はテレワークへ切り替える
- 主回線断の時はモバイル回線やテザリングで重要業務だけつなぐ
- 社内申請システムが止まった時は、一時的にスプレッドシートや紙で受ける
- 共有ファイルが止まった時は、重要フォルダだけ別手段で参照する
- 本社が止まった時は、別拠点か自宅で最低限の意思決定を行う

これも、状況別に表へした方が迷いにくい。

状況	代替手段	先に確認すること
オフィスへ入れない	テレワーク、別拠点、自宅待機と限定業務運転	ログイン、端末、連絡手段、判断者
停電	重要機器の安全停止、代替電源、別拠点運転	安全確保、通電見込み、バックアップ汚染有無
主回線断	モバイル回線、予備回線、テザリング	どの業務だけ先につなぐか
共有ファイル停止	限定フォルダ複製、手作業受付、別手段共有	最新データの所在、後でどう戻すか
仕入先停止	代替仕入先、発注停止、受注制限	どこで在庫と顧客説明を止めるか
欠員	代行者、手順簡素化、承認段階の短縮	誰が代行し、何を省略できるか

ここで重要なのは、何でも代替しようとしめないことだ。全部を平常通りに回そうとすると失敗しやすい。受注だけは止めない、給与だけは遅らせない、対外連絡だけは維持する。このように、重要業務に絞って代替策を考える方が現実的である。

また、手作業へ切り替えるなら、紙で何とかする だけでは足りない。Ready.gov の Business Continuity Plan は、手作業代替について帳票と必要資源を文書化するよう示している。つまり、最低限でも、使う帳票、誰が記入するか、どこへ集めるか、後でどのシステムへ戻すかは決めた方がよい。

手作業代替で決めること	例
使う帳票	受注票、申請票、出荷一覧
必要資源	紙、印刷手段、共有先、担当者
承認者	誰が確認して進めるか
後戻し方法	システム復旧後にどこへ再入力するか

手作業代替で決めること	例
終了条件	いつ手作業運用をやめるか

また、災害時は安全確保が前提である。内閣府ガイドラインも、業務継続や再開の前提として従業員や顧客の安全確保を置いている。したがって、BCPは**仕事を回せ**の計画ではなく、**安全を守りつつ仕事をどう絞って続けるか**の計画である。

## IT DRP は BCP に従って作る

IT DRP は、IT 部門だけの復旧手順書のように見えやすい。だが、Ready.gov は、IT DRP を BCP と一体で作り、IT の復旧優先順位と RTO は BIA から導くべきだとしている。つまり、IT DRP は BCP の下にぶら下がるべきもので、逆ではない。

この原則は、ひとり情シスにとってとても重要である。技術的に戻しやすいものから戻すと、業務優先順位とずれることがあるからだ。たとえば、ファイル共有は戻ったが、受注に必要な認証やネットワークが戻っていない。本番サーバーは起動したが、問い合わせ窓口が止まったまま。こうなると、復旧したようで業務は戻らない。

IT DRP を考える時は、少なくとも次を並べて見たい。

- どの業務を再開するためのシステムか
- そのシステムは何に依存しているか
- どの順で戻すと業務が再開できるか
- どの契約、認証、ライセンス、ベンダー支援が必要か

これも、業務単位の依存関係表にすると実務へ落ちやすい。

重要業務	必要な人	必要な認証 / 回線	必要なアプリ / データ	暫定手段	復旧順
受注	営業、受注担当	ログイン、回線、メール	受注システム、顧客データ	手作業受付	1
顧客対応	窓口担当	ログイン、電話、チャット	問い合わせ履歴、FAQ	電話と表計算	2
給与	人事、経理	ログイン、回線	人事システム、勤怠データ	翌営業日処理	3

たとえば、受注再開には回線、認証、業務アプリ、データ、担当者の端末が要るかもしれない。給与再開には人事システム、認証、ファイル、プリンタ、振込データの出力が要るかもしれない。こうして業務から逆算すると、**サーバーA NAS B**という復旧順ではなく、**受注を戻すにはこれが先**という順に変わる。

第15章で扱ったバックアップや復元手順は、ここで生きる。ただし第23章では、バックアップ方式の詳細より、それが業務再開に間に合うかどうかを見る方が重要である。

## 災害時の連絡、意思決定、対外説明を決めておく

長時間停止で詰まりやすいのは、技術より連絡と判断である。誰が発動を決めるのか。誰が全社連絡を出すのか。顧客には何をいつ伝えるのか。取引先や委託先にはどう知らせるのか。これが曖昧だと、技術復旧が進んでも会社は動きにくい。

Ready.gov は、emergency plans に連絡計画、IT 復旧、継続計画を含めるべきとしている。つまり、BCP に連絡と情報発信は最初から入っていないといけない。

最低限でも、次は決めておきたい。

- 発動条件
  - どんな事象、どの影響で BCP を立ち上げるか
- 判断者
  - 誰が業務停止、代替運用、対外連絡を決めるか
- 緊急連絡網
  - 経営、管理職、現場責任者、ベンダー、主要取引先
- 代替連絡手段
  - 普段のチャットやメールが使えない時にどうするか
- 対外説明
  - 顧客や取引先へ何をどの単位で伝えるか

これも一枚で見られる方が強い。

項目	決めること
発動条件	何時間停止、何拠点影響、何業務停止で立ち上げるか
発動権限	誰が BCP 発動を決めるか
初報先	経営、部門責任者、現場、委託先の順番
代替連絡手段	普段のメールやチャットが使えない時に何を使うか
次回更新時刻	何時に次の状況更新を出すか
対外説明	顧客、取引先、委託先へ何を誰が伝えるか

項目	決めること
安否確認 / 出社可否	誰が集まり、誰が在宅や待機になるか

また、安否確認や出社可否も BCP の一部である。オフィスが無事でも、人が動けなければ業務は回らない。したがって、災害対応では **システムが無事か** だけでなく **誰がどこで働けるか** を早く確認する必要がある。

ここでも一つの source of truth が効く。状況一覧、発動判断、代替手段、次回更新時刻を一か所に集めるだけで、混乱はかなり減る。

## 訓練、テスト、机上演習で計画を使えるものにする

BCP は、書いただけでは動かない。Ready.gov は、訓練、テスト、演習を preparedness の必須要素としている。NIST の TT&E 定義も、要員が役割を理解し、計画の妥当性を演習し、システムの動作を確認する手段と整理している。つまり、訓練はおまけではない。計画の一部である。

ここで訓練を三つに分けると分かりやすい。

- training
  - 誰が何をするかを覚える
- tabletop exercise
  - シナリオで判断や連絡を練習する
- test
  - 実際に回線切替、復旧、代替手段を試す

三つの違いを、次のように置くと運用しやすい。

種類	目的	例
training	役割と手順を理解する	緊急連絡網、発動条件、役割説明
机上演習	判断と連絡の流れを練習する	地震で入館不可 主回線断 で会議する
テスト	戦略やシステムが動くか確認する	テザリング切替、在宅切替、復元確認

読み合わせだけでは弱点が見えにくい。Ready.gov の Testing & Exercises も、代替施設の設備が planned RTO に間に合うか、夜間にチームを呼び出せるか、実際に試さないと分からないと示している。これは本質的である。

小さな会社なら、次のような訓練から始めやすい。

- 緊急連絡網の呼び出しテスト
- 主回線断を想定したテザリングやモバイル回線切替
- オフィス入館不可を想定した在宅切替
- 共有ファイル停止時の手作業運用
- 年1回の机上演習

現実的には、年間で次の最小サイクルがあるだけでも違う。

頻度	やること	見る点
四半期ごと	緊急連絡網テスト	連絡先が古くないか
半年ごと	主回線断や在宅切替のテスト	ログイン、回線、端末が足りるか
年1回	机上演習	判断者、連絡、優先順位が機能するか
年1回	復旧順や復元の確認	RTO と RPO に間に合うか

内閣府ガイドラインも、教育・訓練の結果として見つかった弱点や問題点を、是正・改善へつなげるべきとしている。つまり、訓練の目的は **実施した** ことではない。弱点を見つけて修正することである。

## 小さな会社で現実的に回るやり方

小さな会社で、最初から全社 BCM を完成させるのは難しい。だからこそ、絞った方がよい。中小企業庁は、事業継続力強化計画を **中小企業のための取り組みやすいBCP** と位置付けている。これが示しているのは、簡便な入り口から始めてよいということだ。

現実的には、まず次の五枚があるだけでもかなり違う。

持っておくもの	役割
重要業務と優先順位の表	何を先を守るかを迷わないため
業務ごとの RTO / RPO / 復旧水準表	どこまで止められ、どこまで巻き戻せるかを明確にするため
代替手段表	在宅、手作業、代替回線、別拠点をすぐ選ぶため
発動条件と緊急連絡網	誰が立ち上げ、誰へ連絡するかを迷わないため
年間の訓練計画と改善記録	やりっぱなしにしないため

重要業務は三つから五つでよい。RTO もまずは仮置きでよい。代替手段も、テレワーク、手作業、モバイル回線、別拠点のどれかで十分である。大切なのは、ゼロより一を作ることだ。

訓練も大がかりでなくてよい。中小企業庁が紹介している BCP 訓練企画シートのように、安否確認、初動対応、事業継続の三つに分けて机上演習を組むだけでも、実務には効く。

## 通常時の例と、崩れた時の例

通常時の例では、会社は重要業務を **受注 給与 顧客問い合わせ対応** の三つに絞っている。受注は 4 時間以内に再開、給与は翌営業日、問い合わせ対応は当日中という仮の RTO を置いている。主回線断ではモバイル回線へ切り替え、オフィス入館不可では在宅へ切り替え、共有ファイル停止時は一時的に限定フォルダと手作業で受ける。緊急連絡網は別経路でも届くようにし、年1回は **地震でオフィスに入れない** シナリオで机上演習を行う。結果として、災害時に **全部止まった** ではなく **何を先に回すか** がすぐ決まる。

別の通常時の例では、停電でサーバー室が止まった時に、まず安全確認と出社可否を確認し、その上で給与と受注に必要な認証、回線、ファイル、アプリの復旧順を見直す。回線復旧が遅れる前提で、受注窓口だけはモバイル回線とノートPCで先に立ち上げる。給与処理は翌営業日へずらす判断を経営が承認し、顧客や取引先への説明は決めた文面です。これなら、完全復旧までの間も会社は持ちこたえられる。

崩れた時の例では、BCP はあるが、重要業務が書かれていない。停電でオフィスが止まると、誰も何を優先するか決められない。情シスはファイルサーバーから戻そうとし、営業はメールを先に求め、人事は給与が危ないと言い、経営は顧客向け説明を急ぐ。主回線断の代替手段も試したことがなく、テザリングの台数も不足している。連絡網は古く、夜間に誰ともつながらない。結果として、技術的に全部壊れていたわけではないのに、業務の再開は遅れる。問題は設備不足より、優先順位と訓練不足である。

## 最低限ここまではやる

第23章の内容を一度に全部整えなくてもよい。だが、次の六つは早めに持ちたい。

1. 重要業務を三つに絞り、止まると何が困るかを書く
2. その三つに RTO、RPO、復旧水準を置く
3. 業務ごとの依存関係と IT 復旧順をそろえる
4. 代替手段と手作業運用の最小条件を決める
5. 緊急連絡網、発動条件、判断者を一枚にする
6. 年1回の机上演習と、小さな技術テストを実施する

この六つがあるだけで、BCP はかなり実務になる。BCP とは、何を先に守り、どれだけの時間でどの水準まで戻し、戻るまでどう回すかを平時に決めて、訓練で使える形にしておくことである。

## 今日、今週、後でやること

今日やることは、止まると困る業務を三つだけ書き出し、それぞれに **何時間なら止まれるか** **どこまで巻き戻せるか** **戻るまで何で回すか** を一言添えることである。これだけで BCP の骨格が見える。

今週やることは、その三つに必要なシステム、回線、担当者、代替手段をひも付け、発動条件、判断者、緊急連絡網を一枚へまとめることである。あわせて、主回線断やオフィス入館不可を題材にした短い机上演習の日時も決めたい。

後でやることは、代替回線、在宅切替、手作業運用、復旧順を実際に試し、結果を見直しへつなげることである。ここまでできると、第23章の内容は **読んだ知識** ではなく、長時間停止に耐えるためのBCPになる。

第VI部

# 改善、自動化、AI、引き継ぎ

属人化を減らし、次の運営へ渡す。

第24章～第27章

## 第24章

## 自動化、スクリプト、業務改善

---

ひとり情シスの仕事は、派手な大仕事より、小さな繰り返しで時間を削られやすい。入社のために同じアカウントを作る。毎月ライセンスを数える。週次で同じ管理画面を開いて異常を拾う。問い合わせ票を別の台帳へ転記する。どれも一回ごとは小さい。だが、積み重なると重い。そして手で回している限り、漏れやばらつきも起きやすい。

そこで自動化を考えたいくなる。これは正しい。だが、第24章で強調したいのは、自動化は速くするためだけに行うものではないという点である。本当に重要なのは、同じやり方で回り、漏れが減り、誰が見ても追える形にすることである。ここを外すと、自動化は便利な黒箱になり、別の属人化を生む。

### 自動化すべき仕事の見つけ方

自動化が失敗しやすいのは、手当たり次第に始めるからである。まず見たいのは、その作業が本当に自動化向きかどうかだ。

判断しやすい観点は、次の五つである。

- 頻度が高いか
- 手順が毎回ほぼ同じか
- 入力と出力が明確か
- 例外が少ないか
- 漏れた時の痛みが大きい

たとえば、入社時のアカウント発行、ライセンス棚卸し、週次レポート作成、定期リマインド送信、古い共有リンクの棚卸しは、自動化候補になりやすい。逆に、例外判断が多い作業、関係者ごとにやり方が違う作業、そもそも手順が決まっていない作業は、自動化前に標準化が必要である。

ここで大切なのは、決まっていない仕事を自動化しないことだ。決まっていない仕事を自動化すると、迷いがそのままフローやコードに埋め込まれる。すると後で例外が増え、結局人が横で補正する運用になる。自動化の前に、まず **誰が何を見てどこで終わりとするか** を決めたい。

実務では、次のように考えると選びやすい。

- まずは **月に何回もやっている単純作業** を選ぶ
- 次に **漏れると事故になる作業** を選ぶ
- その後で **人が判断すべき余地が少ない作業** を選ぶ

この順で選ぶと、効果が出やすい。

判断の型を持ちたいなら、次の表が使いやすい。

観点	自動化向き	まだ早い
頻度	毎週、毎月、入退社のたびに必ず発生する	年に数回しかない
手順固定度	誰がやっても同じ順に進む	人によって手順が揺れる
入力/出力	何を受け取り何を出すか明確	入力不足や判断待ちが多い
例外率	例外が少ない	毎回例外処理が必要
漏れた時の影響	事故、未対応、監査漏れになる	少し遅れるだけ

また、自動化の前に最低限そろえたいものもある。

先に決めること	なぜ必要か
開始条件	何を受けたら動くかを固定するため
終了条件	どこまで進めば完了かを固定するため
例外時の扱い	人へ戻す境界を決めるため
確認者	全自動か半自動かを決めるため
記録先	実行結果を後で追うため

## スクリプト、API、ノーコードの使い分け

自動化といっても、手段は一つではない。Power Automate や Jira Automation のようなノーコード型もあれば、Apps Script や PowerShell のようなスクリプト型もある。GitHub Actions のように、コードと一緒にワークフローを管理する形もある。重要なのは、どれが優れているかではなく、どの仕事に向いているかである。

ノーコードやワークフロー型は、**何かが起きたら、条件を見て、何かをする** 仕事に向いている。Atlassian Automation も、rule は trigger、condition、action の三段でできていると説明している。つまり、起票、通知、承認、同期、定期実行のような流れには向いている。見える化しやすく、現場に説明もしやすい。

一方で、処理の分岐が増える、データ整形が多い、複数ページの集計が要る、再利用したい、Git で管理したい、という時はスクリプトの方が向く。Google Apps Script の best practices も、外部呼び出しの最小化や batching を前提に、処理の組み方で性能が大きく変わることを示している。つまり、自由度が高いぶん、設計責任も自分に返ってくる。

API は、その中間ではなく、優先的に検討すべき正式な入口である。対象サービスに公式 API があるなら、まずそれを使う方がよい。画面の位置や文言に依存しにくく、再現性も高いからだ。ノーコードでも、裏では API やコネクタを使っていることが多い。したがって、選び方としては次の順が実務的である。

1. 公式 API や正式なコネクタがあるかを見る
2. それで足りるならノーコードかスクリプトかを選ぶ
3. それでも無理なら画面操作型を検討する

この選び方は、次の表にすると迷いにくい。

手段	向く仕事	主な注意点
公式 API / 正式コネクタ	アカウント、ライセンス、台帳、定期取得のような正式操作	権限設計と利用制限を確認する
ノーコード / workflow	通知、承認、起票、同期、定期リマインド	所有者、DLP、実行履歴、サービス制限を確認する
スクリプト	集計、整形、分岐、再利用、Git 管理したい処理	秘密情報、ログ、クォータ、例外処理を設計する
画面操作	API や正式コネクタがなく、操作対象が安定している時だけ	UI 変更に弱く、壊れやすいので最後に検討する

ここで見落としやすいのが、所有者である。Google Apps Script の installable trigger は作成者アカウントで動く。Power Automate でも所有モデルが安定性に影響し、重要または長時間動くフローでは service principal を勧めている。GitHub Actions の scheduled workflow も、cron を最後に変えた user が notifications の actor になり、actor の状態が実行へ影響しうる。つまり、**何で作るか**と同じくらい**誰の権限で動くか**が重要である。

実行主体は、少なくとも次のように分けて考えたい。

実行主体	向く場面	注意点
個人アカウント	個人専用の補助処理、短命な試作	異動、退職、権限変更で止まりやすい
共有保管 + 複数管理者	チームで保守するスクリプトやルール	編集者を増やしすぎない
service principal / application user	重要、長期、部署横断のフロー	接続共有、ライセンス、要求制限を確認する

Google Apps Script では shared drive が group ownership に寄せやすい。Power Automate では critical or long-running flows に service principal を勤めている。つまり、重要な自動化を **作った人の個人アカウント** 一つに載せないことが出発点である。

## 手作業削減と事故防止を同時に考える

自動化で一番避けたいのは、速くなった代わりに危なくなることである。そのためには、ロジックだけでなく、設定、権限、失敗時の戻し方も一緒に考える必要がある。

最初にやりたいのは、設定値とロジックを分けることだ。Power Automate の environment variables は、hardcoding を避け、flows を portable and easier to maintain にするための仕組みとして説明されている。これは本質的である。URL、メールアドレス、接続先 ID、環境差分、API ベース URL のようなものは、コードやフローの中へ直書きしない方がよい。

秘密情報も同じである。GitHub Actions の secrets は reusable workflows へ自動で渡らないなどの制約があり、Power Platform でも secret 型の environment variable や DLP の考え方がある。ここから言えるのは、秘密情報は **何となく見えない場所** に置けばよいのではなく、明示的に管理しなければならないということだ。

整理の型としては、次のように分けるとよい。

種類	置き場所の考え方	例
ロジック	コード、フロー定義、workflow 定義	条件分岐、整形、処理順
変わりやすい設定値	environment variable や変数	URL、メールアドレス、対象 ID
秘密情報	secrets や secret 型変数	API key、token、connection string
実行主体	identity を明示的に決める	service principal、共有実行 user

また、自動化は **成功する時** だけでなく **失敗する時** を先に考えたい。最低限でも、次は持ちたい。

- dry run やテスト方法
- 途中失敗時にどう止めるか
- 二重実行しても壊れにくい
- 手で戻す時は何を戻すか

章としては、次の表にした方が実務へ落ちる。

失敗前提で決めること	何を見るか
dry run	まず下書きや一覧出力だけで試せるか

失敗前提で決めること	何を見るか
二重実行耐性	同じ処理をもう一度流しても壊れないか
停止方法	どこで止めるか、誰が止めるか
手動代替	止まった時に人が何をするか
例外通知	誰へ、どの手段で通知するか
DLP / quota / limits	policy や制限超過で止まらないか

たとえば、アカウント発行自動化なら、最初は **申請内容を整形して下書きまで**にとどめる方法もある。いきなり作成、権限付与、端末登録まで全自動にする必要はない。半自動でも、漏れが減るなら十分価値がある。

さらに、接続の組み合わせにも注意がいる。Power Automate の DLP policy は、どの connector や desktop flow module を組み合わせられるかを管理し、違反 flow は suspend されうる。これは単なる管理機能ではない。自動化がデータ持ち出し経路になりうることを示している。第24章では、自動化を作る時点で、**どの情報をどこへ渡してよいか**を確認する必要があると書きたい。

## 運用改善を定着させる

自動化は、一回動けば終わりではない。むしろ問題はその後に出る。人が変わる。接続先が変わる。ライセンスが変わる。URL が変わる。通知先が死ぬ。第24章で重要なのは、これを前提に残すことだ。

Power Automate は co-owner を必要最小限にし、通常は run-only permissions の共有で足りる場面が多いとしている。Apps Script は shared drives を使えば group ownership に寄せられる。GitHub Actions は reusable workflows で重複を減らし、SHA pinning が stability and security の観点で safest option だとしている。これらを踏まえると、維持しやすい自動化には共通点がある。

- 担当者が明確
- 予備の管理者がいる
- 定義ファイルやコードの置き場所が決まっている
- 名前と説明がある
- 再利用部分が共通化されている

名前も重要である。test2 flow-final script\_new のような名前では、後で誰も分からない。何をする自動化か、いつ動くか、誰が見るかが分かる名前にした方がよい。

また、ルールやフローは更新後の確認も必要である。Atlassian Automation も、有効化した後に audit log の status を見て成功確認するよう案内している。つまり、自動化は 保存した時点 ではなく 一回流して確認した時点 で初めて運用に乗る。

そのため、最低限の台帳を持った方がよい。

台帳に残す列	何のために残すか
名前	何をする自動化かを識別するため
目的	何を減らすための自動化かを共有するため
owner	誰が保守責任を持つかを明確にするため
実行主体	どの権限で動いているかを把握するため
実行頻度 / trigger	いつ動くかを把握するため

台帳に残す列	何のために残すか
通知先	失敗に気づくため
ログの場所	どこで実行結果を見るかを明確にするため
保持期間	どれだけ証跡が残るかを把握するため
停止手順	危ない時に止めるため
手動代替手順	停止時も業務を止め切らないため

## 自動化の証跡とメンテナンス

自動化が危ういのは、止まることそのものより、止まったことに気づかない時である。だから、証跡と監視が必要になる。

Power Automate では、run ごとの inputs と outputs を見ながら確認できる。一方で、run history は既定で 28 日保持である。Atlassian の audit log は 90 日保持で、それ以前は復元できない。Apps Script では execution log は開発向けで短く、multi-user production environment では Cloud Logging が preferred choice とされている。つまり、**ログがあるか**だけでなく**どれだけ残るか**が違う。

ここで最低限ほしいのは、次の四つである。

- 失敗時に通知が来る
- 実行結果を後から見返せる
- どの設定値で動いたか追える
- 止め方が分かる

さらに、ツールごとの差も知っておきたい。

仕組み	実務で気をつける点
Apps Script	installable trigger は作成者権限で動く。 execution log は短く、production では Cloud Logging を寄せ先にした方がよい
Power Automate	run history は既定 28 日。DLP violation では flow が suspended になりうる
GitHub Actions	reusable workflow へ secrets は自動継承され ない。scheduled workflow の通知先は cron を最後に変えた user へ向く
Atlassian Automation	audit log は 90 日。service limits 超過時は THROTTLED で見える

さらに、量が増えるとクォータや制限も効く。Apps Script は quotas を超えると exception で止まりうる。Atlassian Automation も service limits を超えると throttled になる。Scheduled rules that can run at the same time の制限や、検索件数の上限もある。つまり、自動化は **一回通る** だけでなく **継続的にその件数で回るか** を見なければならない。

この意味で、メンテナンス対象として残したい情報は次のようになる。

- 担当者
- 通知先
- 実行頻度
- 依存する接続先
- 環境変数や secrets の場所
- 停止手順
- 手動代替手順

これがない自動化は、動いている時だけ便利で、止まった時に面倒を増やす。

## 小さな会社で現実的に回るやり方

小さな会社では、最初から複雑な自動化基盤を作る必要はない。むしろ、単純で、効果が見えやすく、止めても戻しやすいものから始めた方がよい。

最初の候補として向いているのは、たとえば次のようなものだ。

- 入社、異動、退職の依頼起票とチェックリスト生成
- ライセンス棚卸しの定期レポート
- 共有アドレスや問い合わせ箱からの転記
- 毎週の管理者向けレポート送信
- 契約更新や証明書期限の通知

ここで意図的に避けたいのは、いきなり基幹処理を全自動にすることである。まずは、通知、集計、棚卸し、下書き生成のような **壊れてもすぐ戻せる仕事** から始めた方がよい。その中で、担当者、通知、ログ、停止方法の型を作っていく方が、後から強い。

## 通常時の例と、崩れた時の例

通常時の例では、入社申請フォームが出た時に、自動で **アカウント発行 貸与端末準備 初期権限確認** のチケットを起票する。申請内容から部署、入社日、上長を読み取り、標準的な権限候補を提示するが、最終付与は情シスが確認する。フローの担当者は個人ではなく共有管理アカウントか **service principal** に寄せ、通知先は情シス共有アドレスにする。接続先 **URL** や対象グループは **environment variable** で分け、変更時にロジックを触らない。こうすると、全自動でなくても漏れは大きく減る。

別の通常時の例では、毎週 1 回、スクリプトが Microsoft 365 や Google Workspace からライセンス割り当て情報を取得し、退職者や退職済み候補、未使用ライセンス候補を一覧化する。結果は表に出して共有し、除外対象は別リストで持つ。スクリプト本体は共有リポジトリへ置き、接続情報は secrets で持つ。実行ログと失敗通知も残す。これなら、棚卸しは人が見る仕事になり、毎回集める仕事ではなくなる。

崩れた時の例では、ある社員が自分の Google アカウントで Apps Script を作り、毎朝自動でレポートを送っていた。動いている間は便利だったが、異動後に権限が変わり、installable trigger はその人の権限で動いていたため止まる。通知先も本人だけで、誰も気づかない。別のフローでは、API キーと URL が直書きされており、環境移行時に全部を手で直す。さらに、run history は既定期間を過ぎて消え、原因も追えない。問題は自動化したことではなく、所有者、設定値、証跡を設計せずに作ったことである。

## 最低限ここまではやる

第24章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 自動化候補を頻度と例外率で選ぶ
2. 重要な自動化を個人アカウント一つに依存させない
3. URL や接続先や秘密情報を直書きしない
4. 失敗通知と実行履歴の確認場所を決める
5. 止め方と手動代替手順を一行でも残す

この五つがあるだけで、自動化はかなり実務的になる。自動化とは、手作業を減らすことではない。標準化された仕事を、壊れにくく、追跡できる形で回すことである。

## 今日、今週、後でやること

今日やることは、毎週または毎月繰り返している作業を五つ書き出し、**頻度 例外率 漏れた時の痛さ**の三点で見直すことである。ここで一つだけ、自動化候補を選ぶ。

今週やることは、その候補について、担当者、通知先、停止手順、手動代替手順、設定値の置き場所を決めることである。あわせて、ノーコードで足りるのか、スクリプトが必要か、公式APIがあるかも確認したい。

後でやることは、自動化した処理を再利用しやすい形へ寄せ、ログ保持期間やクォータも見直ししながら、二つ目、三つ目の改善へ広げることである。ここまでできると、第24章の自動化は**便利な個人技**ではなく、運用改善の基盤になる。

## 第25章

# 可視化、指標、コスト最適化

ひとり情シスの仕事は、実際にはかなり多くのものを扱っている。問い合わせ対応、端末配布、権限付与、退職処理、ライセンス棚卸し、契約更新、請求確認、障害対応、ベンダーとのやり取り。だが、これらはしばしば別々の場所に散っている。チケットには問い合わせ件数がある。管理画面にはアカウントやライセンスの状況がある。請求書には金額がある。更新日はカレンダーやメールの奥に埋もれる。こうなると、忙しいことは分かっても、何に時間が吸われ、何が改善され、何にムダがあるのかは見えにくい。

第25章で強調したいのは、可視化は報告資料を作るための仕事ではないという点である。本当に重要なのは、何を続け、何を止め、どこへ時間と予算を振るかを決められる状態を作ることである。数字がないと、改善も説明も属人的になる。逆に、数字がありすぎても判断は鈍る。したがって第25章では、**全部を見る化する**ではなく、**意思決定に必要なものを継続的に見える化する**ことを軸にしたい。

最初に見る数字は、次の五つで十分である。

指標	定義の例	何が分かるか
滞留件数	月末時点で未完了の依頼と障害の件数	仕事が積み上がっていないか
SLA達成率	主要依頼で期限内に終了した割合	約束した水準を守れているか
再オープン率	完了後に再度開いた件数の割合	手戻りや雑な完了がないか

指標	定義の例	何が分かるか
休眠ライセンス率	一定期間使われていない席の割合	コスト削減余地があるか
90日以内の更新件数	近い更新や解約判断の件数	先回りが必要な契約がどれだけあるか

第25章の残りは、この五つを回し始めた後に足していく論点として読むとよい。最初から全部を同時に持とうとすると、第25章で一番避けたい **数字が多すぎて判断できない状態** に戻りやすい。

## 情シス業務を見える化する

見える化で最初にやるべきことは、グラフを作ることではない。何を一件として数えるか、どの単位で持つかを決めることである。

たとえば、次のものは分けて持った方がよい。

- 依頼
- インシデント
- 変更
- 資産棚卸し
- 契約更新
- 請求確認

実務では、次の表で分けておくと混ざりにくい。

種類	何を一件として数えるか	分ける理由
依頼	アカウント発行、端末貸与、権限変更など	需要量と標準化余地を見るため

種類	何を一件として数えるか	分ける理由
インシデント	障害、誤設定、利用不能、性能低下など	品質と再発を見るため
変更	設定変更、導入変更、承認付き変更など	リスクと変更負荷を見るため
資産棚卸し	端末、アカウント、権限、ライセンスの確認	漏れや休眠を減らすため
契約更新	SaaS、回線、保守、ドメインなどの更新判断	先回り判断をするため
請求確認	月額、従量、異常請求、差異確認など	コスト差異と異常支出を見るため

この区別がないと、数字が混ざる。新規アカウント発行と障害復旧を同じ **問い合わせ件数** に入れると、件数は出ても中身が分からない。ドメイン更新や回線更新のような期限管理も、依頼件数の中に埋めると見えなくなる。したがって、まずは **どの種類の仕事か** を分けたい。

小さな会社で最初に持つべき見える化単位は、五つで十分である。

- 人
- サービス
- 契約
- コスト
- 期限

人では、誰が使っているか、休眠か、退職済みかを見る。サービスでは、どのツールや回線やシステムがあるかを見る。契約では、年契約か月契約か、解約期限はいつかを見る。コストでは、月額か従量か、誰の意思決定で発生しているかを見る。期限では、更新日、請求日、証明書期限、ドメイン期限、保守期限を持つ。

ここで重要なのは、可視化のために新しい巨大な仕組みを作らないことである。第25章の段階では、最初から BI 基盤を導入する必要はない。ひとり情シスに必要なのは、次が一つの流れで見えることだ。

- 今月どんな仕事は何件あったか
- どこで滞留しているか
- 何にいくら払っているか
- 使っていないものは何か
- 何がいつ更新されるか

これが見えれば、すでに十分に実務的である。

## 何を KPI として追うか

数字を集め始めると、次に起きやすい失敗は、全部を KPI と呼んでしまうことである。だが、**KPI** と **運用指標** は分けた方がよい。

ここから先は、最初の五つを置いた後に、定義と補助指標を整える段階の話である。

KPI は、少数でよい。月次や四半期で、方向性が良いか悪いかを判断するものだ。一方、運用指標は、日々の詰まりや原因を見るためのものだ。件数、SLA、再オープン率、CSAT、自己解決率、未使用ライセンス率、クラウド予算差異など、どれも重要だが、全部を同じ重さで扱っていると焦点がぼける。

切り分けるなら、次のように考えるとぶれにくい。

種類	何を見るか	例
KPI	月次や四半期で良化しているか	滞留件数、SLA達成率、再オープン率、休眠ライセンス率

種類	何を見るか	例
運用指標	どこが詰まり、何が増えたか	依頼種別ごとの件数、部署別件数、特定サービスの利用率

見やすく整理すると、指標は次の六種類に分けられる。

- 量
- 時間
- 品質
- 自己解決
- リスク
- コスト

量には、問い合わせ件数、変更件数、未処理件数、滞留件数の増減がある。時間には、初回応答までの時間、解決までの時間、SLA 達成率がある。品質には、再オープン率、CSAT、初回解決率がある。自己解決には、ナレッジ閲覧や自己解決できた件数のような指標がある。リスクには、権限棚卸し完了率、バックアップ復元テスト実施率、期限切れゼロかどうかがある。コストには、ライセンス使用率、1席あたりコスト、サービス別月額、予算差異、異常支出がある。

Atlassian の一次情報でも、Jira Service Management の標準レポートは Workload、Satisfaction、Requests deflected、Requests resolved を持っている。またダッシュボード指標として、SLA 達成率、SLA breach 率、first contact resolution、reopened rate、backlog growth まで定義している。これは示唆的である。つまり、**たくさん来た** だけでは足りず、**間に合ったか やり直しが起きたか 自己解決できたか** を一緒に見るべきなのである。

小さな会社で最初に置く KPI は、たとえば次のようなもので十分である。

1. 月次の滞留件数が増えていないか
2. 主要な依頼の SLA を守れているか
3. 再オープン率や手戻りが下がっているか
4. 未使用ライセンスや休眠アカウントが減っているか
5. 更新漏れや予算超過の予兆を早く拾えているか

そしてその下に、補助指標を置く。たとえば **問い合わせ件数を部署別に見る どの依頼種別が増えたか見る どのツールで休眠率が高いか見る** といった具合である。

ここで大事なのは、指標の定義を曖昧にしないことだ。未使用ライセンス を、30 日無活動とするのか、90 日無活動とするのか。解決時間 を、受付から完了までとするのか、営業時間だけで測るのか。自己解決 を、記事閲覧数で見るとのか、deflection で見るとのか。定義が曖昧だと、前月比較も説明も崩れる。

最低限、指標ごとに次は決めておきたい。

決めること	例
名前	休眠ライセンス率
定義	90 日無活動の席数 / 全契約席数
対象期間	月次
データ源	Microsoft 365 reports、Google Workspace reports
遅延	2 日～3 日遅れ前提
使い道	次回更新での縮小判断

さらに、レポートの鮮度にも注意がいる。Microsoft 365 の usage report は通常 24～72 時間の遅れがあり、Google Workspace の reports も 1～3 日、指標によっては 1～6 日遅れる。つまり、これらはリアルタイム監視には向かない。第25章では、**即時対応の数字** と **週次・月次棚卸しの数字** を分けて考えるべきだと書きたい。

数字ごとの鮮度は、次のように分けて見た方がよい。

データ源	代表的な遅延	向く用途
Microsoft 365 usage reports	通常 48 時間以内、数日かかる場合あり	週次、月次の利用棚卸し
Google Workspace reports	1～3 日、adoption metrics は 1～6 日	週次、月次の利用棚卸し
AWS anomaly detection	最大 24 時間程度の遅れ	日次の異常支出確認
Azure cost data	open billing period は 4 時間ごと更新	日次～週次の cost monitoring

## ライセンス、回線、クラウド費用の見直し

コスト最適化というと、安い製品へ乗り換える話だと思われがちである。だが、ひとり情シスの実務で先に効くのは、使っていないもの、契約条件と合っていないもの、所有者が曖昧なものを整理することである。

最初に見たいのは、**契約している数** と **実際に使っている数** が一致しているかである。Microsoft 365 の activity reports は、サービスをほとんど使っていないユーザーを把握し、ライセンス見直しの候補を拾う材料になる。Google Workspace でも、Users Usage Report や各種 usage data から last login や利用実態を長めの期間で見られる。したがって、次のような棚卸しが可能になる。

- active だが実利用が少ないユーザー

- suspended や archived へ寄せるべきユーザー
- 退職済みだが add-on ライセンスが残っているユーザー
- guest や委託先で不要になった利用者

この棚卸しは、次の表で持つと判断しやすい。

見る列	何を見るか
契約席数	いま何席に対して払っているか
active 利用者数	実際に使っている人数は何人か
休眠候補	30日や90日無活動の候補は何人か
契約形態	FlexibleかAnnual/Fixed-Termか
archive / delete 方針	保持と削減をどう両立するか
次の減額可能日	いつ請求へ反映できるか

ただし、利用実態だけ見て席数を減らせればよいとは限らない。ここで契約条件が効く。Google Workspace の一次情報では、Flexible Plan は commitment なしだが、Annual/Fixed-Term Plan は 1 年以上の commitment があり、更新時まで minimum number of user licenses を減らせない。つまり、未使用アカウントを消した事自体には意味があるが、年間契約なら今月の請求がすぐ下がるとは限らない。この違いを知らないと、減らしたのに安くならないという誤解が起きる。

退職者対応でも同じである。Google Workspace の archived user は active license を 24 時間以内に再利用可能にしつつ、データを保持できる。一方で、archived user のデータは pooled storage に残り続ける。したがって、第25章では席を空けると保存容量を減らすは別問題だと書いておきたい。

固定費の見直し対象は、SaaS だけではない。回線、VPN、MDM、EDR、バックアップ、ドメイン、DNS、証明書監視、保守契約、SMS 認証、クラウド commit などとも並べて見た方がよい。ここでは **高いか安い** かより、次を見たい。

- いま誰が使っているか
- なくすと何が止まるか
- 代替があるか
- 契約条件は何か
- 更新判断をいつする必要があるか

クラウド費用はさらに、**請求を見た後に知る** から抜ける必要がある。AWS は cost anomaly detection で anomalous spend を検知し、service、account、region、usage type ごとに根本原因候補を見られる。Google Cloud budgets は actual cost と planned cost を比べ、alert threshold や Pub/Sub 通知を設定できる。Azure も exports や query API を前提に、タグ、management groups、custom dimensions で cost allocation することを勧めている。

役割の違いは、次のように見た方がよい。

仕組み	できること	できないこと
AWS anomaly detection	異常支出の通知、root cause 候補の確認	即時停止や完全リアルタイム検知
Google Cloud budgets	予算閾値通知、Pub/Sub 連携、scope 切り分け	予算超過時の自動 cap
Azure cost management	exports、query、tags、management groups による配賦	所有者設計なしでの明確な説明

ここで強く書いておきたいのは、予算アラートは自動停止ではないという点である。Google Cloud も、budget は spend を自動で cap しないと明記している。AWS anomaly detection も、検知まで最大 24 時間程度の遅延がありうる。したがって、予算や異常検知は **安心の仕組み** ではなく **動くきっかけ** である。通知を誰が受け、どこを見て、どこまで掘るかを決めなければ効かない。

クラウド費用では、総額だけでなく、誰の責任範囲かが分かるようにしたい。AWS は cost allocation tags や cost categories、Azure は tags や management groups、Google Cloud は project や labels で scope を切れる。第25章では、**タグがないと分析できない** という言い方より、**責任の所在が分からない費用は削減も説明もできない** と書く方が伝わる。

## 契約更新と棚卸しを定期化する

コスト最適化が失敗しやすいのは、更新の直前に慌てて判断するからである。ひとり情シスが持つべきなのは、単なる契約一覧ではなく、更新判断に必要な情報が入った更新カレンダーである。

最低限、次は持ちたい。

- 契約名
- サービス名
- 担当者
- ベンダー
- 月額または年額
- 契約形態
- 更新日
- 解約期限

- 最低利用期間
- 自動更新の有無
- 継続、縮小、停止の判断期限

ここでのコツは、更新日だけでなく、いつまでにやめる判断をしなければならぬいかを持つことである。たとえば年契約の SaaS は、更新日の 30 日前や 60 日前に解約意思表示が必要なことがある。回線や保守契約は、切替や撤去に時間がかかる。したがって、見るべき日は一つではない。

実務では、次の四段階に分けると回しやすい。

- 90 日前に利用実態と代替可否を見る
- 60 日前に継続、縮小、停止の仮判断を置く
- 30 日前に決裁とベンダー連絡を済ませる
- 7 日前に請求、設定、通知先の最終確認をする

これも表にしておくと強い。

タイミング	何を見るか
90 日前	利用実態、代替可否、影響範囲
60 日前	継続、縮小、停止の仮判断
30 日前	決裁、ベンダー連絡、解約意思表示
7 日前	請求、設定、通知先、切替作業の最終確認

更新判断の材料も、金額だけでは弱い。最低でも次が必要である。

- 現在の利用者数
- 実際の利用頻度
- 代替手段の有無

- 停止した時の影響
- 契約条件
- データ引き上げや移行の難易度

これがあれば、**何となく継続** を減らせる。逆に、これがない更新は、使っていないのに続くか、本当は必要なのに切って事故になるかのどちらかになりやすい。

ドメイン、証明書、DNS、回線、SaaS、セキュリティ製品、クラウド commit は、部署横断で影響が出ることが多い。第25章では、更新カレンダーを **ひとり情シスだけの TODO** ではなく、会社の継続運営に必要な管理表として位置づけたい。

## 経営へ説明できる数字を持つ

情シスの仕事は、数字がないと **いろいろやっているらしい** で終わりやすい。だが、件数だけの報告でも弱い。問い合わせ件数が 100 件だったとしても、それだけでは多いのか少ないのか、改善したのか悪化したのか、経営には伝わりにくい。

説明に必要なのは、**量 時間 品質 費用 価値** をつなぐことである。たとえば次のように言い換える。

- 問い合わせは 100 件だった
- そのうち 35 件はパスワードや入社手続きで、自己解決導線を作れる類型だった
- backlog growth は前月より下がった
- SLA 達成率は維持した
- 再オープン率は下がった

- その結果、月末の滞留が減り、翌月の負荷平準化につながった

これなら、単なる件数報告ではなくなる。

コストも同じである。年額 30 万円下げただけでなく、退職者と休眠ユーザーを整理して未使用席を解消した 年間契約なので即減額ではないが、次回更新時の縮小余地を確定した 今後 12 か月の固定費見込みを下げた とつなぐ方が、意思決定に耐える。

月次の一枚ダッシュボードが回り始めたら、さらに役立つのが unit economics 的な見方である。これは難しい話ではない。小さな会社なら、たとえば次のような見方で十分である。

- cost per active seat
- cost per request resolved
- cost per endpoint managed
- cost per new employee onboarded
- cost per branch office maintained

この見方も、何に使うかまで書いた方が伝わる。

見方	何に使うか
cost per active seat	ライセンスや SaaS の過不足を見る
cost per request resolved	依頼処理の改善効果を見る
cost per endpoint managed	端末管理費の重さを見る
cost per new employee onboarded	入社対応の効率を見る
cost per branch office maintained	拠点維持コストの見直し余地を見る

重要なのは、比較のために無理に全部を同じ単位へ押し込まないことだ。FinOps の一次情報でも、異なる事業やサービスを無理に横並び比較するより、同じ対象を時間推移で見る方が実務的だとしている。第25章でも、**前月比 四半期比 同じ対象の改善前後** を重視した方がよい。

また、経営向けの数字は少なくてよい。現場向けには詳細が必要だが、役員向けに毎回 20 指標を出す必要はない。むしろ次のような一枚の方が伝わる。

- 今月の主要運用量
- 今月の主要品質
- 今月の主要コスト差異
- 今後 90 日の更新とリスク
- 先月からの改善と次の打ち手

最初の月次一枚ダッシュボードは、次の形で十分である。

欄	最低限入れるもの
運用量	依頼件数、インシデント件数、変更件数、滞留件数
品質	SLA 達成率、再オープン率
コスト	主要 SaaS の席数、休眠候補、今月の差異
期限	90 日以内の更新、解約期限、保守期限
次の打ち手	今月やめること、続けること、直すことを一挙ずつ

これは chargeback のように厳密な部門課金をする話ではない。小さな会社では、まず **showback**、つまり **どこで何が使われているかを見せる** だけでも十分に意味がある。

## 通常時の例と、崩れた時の例

通常時の例では、月次レビュー用に一枚の運用サマリーを持つ。左側には依頼、インシデント、変更の件数と前月差、滞留件数の増減、SLA 達成率、再オープン率を置く。右側には主要 SaaS の契約数、active 利用者数、休眠候補、来月以降 90 日の更新予定を置く。下段にはクラウド費用の予算差異と異常検知の有無を置く。数字は多く見えるが、役割ごとにまとまっているため、**何が増えたか どこが詰まっているか 何を見直すか**がその場で決まる。

別の通常時の例では、Microsoft 365 と Google Workspace の usage report を使い、90 日以上実利用が少ない候補を毎月洗い出す。退職者は archive 候補へ、休職者は停止候補へ、委託先終了者は削除候補へ分ける。ここで契約条件も横に置き、Flexible plan なら翌月反映を狙い、Annual/Fixed-Term なら次回更新での縮小候補として積み上げる。これなら、**今月すぐ下がる費用 と 次回更新で下がる費用**を混同しない。

クラウド費用の通常時の例では、Google Cloud budgets や AWS anomaly detection の通知を情シス共有アドレスと経理担当へ送る。通知が来たら、project、service、region、usage type、invoice reconciliation を見て原因を切り分ける。月末には cost table や export を使って、どのプロジェクトや SKU が増えたかを見る。これなら、**何か高い から どこが原因か**まで進める。

崩れた時の例では、問い合わせ件数だけを毎月報告していたため、件数は横ばいなのに滞留件数が積み上がっていることに誰も気づかない。再オープン率も見えていないので、同じトラブルが繰り返し開き直されている。別の会社では、年間契約の SaaS で unused seat を削除したので来月から請求が下がると思っていたが、実際には更新時まで減額されず、経営への説明で詰まる。どちらも問題は数字不足ではない。数字の定義と見方が足りなかったのである。

さらに別の失敗では、Google Workspace や Microsoft 365 の usage report を即時監視のつもりで見えており、レポート遅延の存在を考えていなかった。そのため、当日の異常を **レポートに出ていないから問題なし** と誤解する。第25章では、これは避けたい。可視化は万能ではなく、鮮度と保持期間の前提がある。

## 最低限ここまではやる

第25章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 依頼、インシデント、変更、更新、請求を分けて数える
2. KPI を少数に絞り、定義を一行でも書く
3. ライセンス利用実態と契約条件を同時に見る
4. 予算アラートや異常通知の受け先と初動を見る人を決める
5. 更新カレンダーに解約期限と担当者を入れる

この五つがあるだけで、情シス運用はかなり見えやすくなる。可視化とは数字を並べることではない。次の判断ができる状態を、継続的に作ることである。

## 今日、今週、後でやること

今日やることは、いま毎月見ている数字を十個書き出し、そのうち **意思決定に使っている数字** と **惰性で見ている数字** を分けることである。ここで、残す指標を五つに絞りたい。

今週やることは、主要 SaaS を一つ選び、契約席数、active 利用者数、休眠候補、契約形態、更新日、解約期限を一枚で見えるようにすることである。あわせて、問い合わせ系では滞留件数、SLA、再オープン率のどれか一つを新たに見える化したい。

後でやることは、クラウド予算通知や usage report を自動取り込みし、月次レビュー資料を半自動化しながら、**総額**だけでなく**1席あたり 1件あたり**の見方へ広げることである。ここまでできると、第25章の可視化は**数字を集める作業**ではなく、改善と説明の基盤になる。

## 第26章

# 生成 AI 活用と AI ガバナンス

---

生成 AI は、導入のハードルが低い。ブラウザを開けばすぐ使える。文章の下書き、要約、翻訳、検索補助、会議メモ、社内文書の問い合わせ対応まで、効果も見えやすい。だからこそ、企業では **気づいたら各自が使っている** 状態になりやすい。だが、ここで見落とししやすいのは、生成 AI が単なる便利なチャットではないという点である。生成 AI は、情報を入力し、処理し、生成物として外へ出す、新しい情報経路である。

第26章で強調したいのは、生成 AI の論点は **使うかどうか** より前に **どの条件なら使ってもよいか** を決めることだという点である。プロンプトの書き方より先に、承認済みサービス、入力できる情報、共有範囲、外部接続、ログ、保持、監査を決めた方がよい。ここを曖昧にすると、便利さの裏で、個人情報の入力、機密の混入、過剰共有の可視化、外部 API への送信、出力の誤用が同時に進む。

最初の運用ルールは、次の五つに絞ると回しやすい。

1. 承認済みサービスだけを使う
2. 個人情報、秘密情報、認証情報は入れない
3. 外部 app、action、connector は別承認にする
4. 出力は下書きとして扱い、人が確認してから使う
5. 業務用のカスタム AI には所有者、共有範囲、停止方法を持たせる

最初に整理するなら、次の表で十分である。

用途	主なリスク	先に決めること
文書作成や要約	誤情報、権利侵害、誤送信	承認済みサービス、人手確認が必要な用途
社内検索	oversharing、権限継承、監査不足	対象データ、共有整理、ログ取得
外部接続	第三者送信、誤操作、自動実行	action / app / connector の別承認、許可 domain
顧客向け / 公開向け	ブランド毀損、法務事故、不正確な対外説明	確認者、公開前レビュー、停止連絡先

## 生成 AI を社内導入する前に決めること

生成 AI を導入する前に、最初に決めるべきものは **どのツールを入れるか** ではない。**何のために使うか 誰が使うか 何をに入れてよいか どこまでつながるか** である。

用途は大きく分けると、次の四つで考えると整理しやすい。

- 一般的な文章作成や要約の補助
- 社内文書や社内データを参照する検索補助
- 外部サービスとつながる workflow や agent
- 顧客向け、公開向けの生成

この四つは、同じ **生成 AI** でもリスクが違う。一般的な文章補助は比較的始めやすい。だが、社内文書検索になると権限と過剰共有の問題が入る。外部サービスとつながる actions や apps、connectors になると、第三者送信や誤操作の問題が入る。顧客向けや公開向けになると、誤情報、権利侵害、ブランド毀損の問題が強くなる。したがって、まずは用途で分けたい。

次に、承認済みサービスを定める。ここで重要なのは、**生成 AI** を一括りにしないことだ。OpenAI の個人向け ChatGPT、ChatGPT Business、ChatGPT Enterprise は同じではない。Google でも Gemini in Workspace、Gemini app、NotebookLM、Gem sharing は同じではない。Microsoft でも Copilot Chat、本体機能、connectors、custom agents は同じではない。学習利用、保持期間、監査、共有、管理者設定が違うからである。特に Google は、同じ Gemini app でも、qualifying Workspace edition にひもづく業務向け利用と、additional Google Service としての利用で、適用される規約とデータ取扱いの前提が変わる。

そのため、導入前に最低でも次は確認したい。

- 入力データが学習に使われるか
- prompts と responses の保持期間はどうか
- 保存先や適用法令に注意点はるか
- ログや監査証跡を取得できるか
- 共有範囲を制御できるか
- 外部 app、action、connector を制限できるか
- 退職や異動時に所有者や設定を回収できるか

製品名より、2026年3月22日時点で確認できる主な差を先に押さえたい。

代表サービス	2026年3月22日時点で確認できる主な仕様	実務上の含意
Microsoft 365 Copilot Chat	prompts / responses は foundation model 学習に使わず、監査のため Exchange に記録される。web grounding は Bing search service を使う	監査導線と web grounding を分けて考える

代表サービス	2026年3月22日時点で確認できる主な仕様	実務上の含意
Google Workspace with Gemini	prompts、outputs、training data を Google や他顧客モデルの学習に使わない。Gem sharing は Drive 共有に乗る	Workspace 本体設定と Drive 共有を一体で管理する
ChatGPT Business / Enterprise	組織データをデフォルトで学習に使わない。workspace owners は GPT sharing、actions domains、third-party GPTs を制御できる	本体チャット、社内 GPT、外部 GPT / actions を分けて承認する

総務省・経済産業省の AI 事業者ガイドライン 第1.1版は、AI 活用においてリスクの大きさに応じた対策を取る **リスクベースアプローチ** を示している。ここで言いたいのは、**全部禁止** か **全部自由** かの二択ではなく、用途とリスクに応じて線を引くべきだということである。ひとり情シスに必要なのも、この線引きである。

また、契約確認も後回しにしない方がよい。経済産業省の **AIの利用・開発に関する契約チェックリスト** は、データの利用範囲、AI 生成物の利用条件、ログ保存、監査、セキュリティ、規約改定まで確認するよう整理している。つまり、生成 AI の導入は **SaaS を一つ増やす** ではなく、**新しい情報処理先と契約する** ことである。

ここで押さえないのは、製品名やプラン名は変わっても、判断軸は大きく変わりにくいという点である。**学習利用 保持 共有 外部接続 監査** の五つを先に確認する姿勢は、どの生成 AI サービスでも崩れにくい。

## 入力禁止情報と利用ルール

次に必要なのは、入力禁止情報と利用ルールである。ここでありがちな失敗は、禁止事項だけを書いて終わることだ。だが実際には、何を入れてはいけないかと同じくらいどの条件なら使ってよいかを書いた方が運用しやすい。

入力禁止情報として早めに分けたいのは、たとえば次のようなものである。

- 個人情報
- 要配慮個人情報
- 顧客との契約情報
- 未公開の経営情報
- 営業秘密
- ソースコードや設計情報
- 認証情報、API キー、秘密情報
- 第三者から預かった非公開データ

個人情報保護委員会の注意喚起は、個人情報取扱事業者が個人情報を含むプロンプトを入力する時、利用目的達成に必要な範囲内かを確認すること、本人同意なく個人データを入力し、応答結果以外の目的で取り扱われる場合は、法令違反となる可能性があることを示している。したがって、第26章では 個人情報を入れるな とだけ書くのでは足りない。利用目的、同意、機械学習への利用有無、出力の不正確さまで含めて判断する必要がある。

ここで実務的なのは、禁止 条件付き許可 許可 の三段で持つことだ。

- 禁止
  - 顧客個人情報、要配慮個人情報、秘密情報、認証情報

- 条件付き許可
  - 社内資料の要約、匿名化した事例、公開前提でないが秘匿度の低い文案
- 許可
  - 公開済み資料の要約、一般的な文面の下書き、公開情報の整理

この三段にすると、現場は迷いにくい。

三段で書くなら、次の形が使いやすい。

区分	代表例	使う条件
禁止	顧客個人情報、要配慮個人情報、API キー、秘密情報	承認済みサービスでも入力しない
条件付き許可	社内資料の要約、匿名化した事例、秘匿度の低い草案	承認済みサービス、外部接続なし、人手確認前提
許可	公開済み資料の要約、一般文面の下書き、公開情報の整理	出力をそのまま確定文にしない

迷った時の最初の判定は、次の表で足りる。

用途	まず始めやすい	条件付きで進める	まだ避けたい
文書作成	公開済み資料の要約、一般文面の下書き	社内資料の要約	契約確定文、対外確定回答
社内検索	公開済み FAQ、公開済み手順	権限制御済みの社内文書	過剰共有が残る共有フォルダ横断検索
自動処理	下書き生成、通知作成	承認つき workflow 補助	権限付与、送金、削除の自動判断
外部接続	なし	承認済み connector のみ	未承認 API や外部 app への送信

また、入力だけでなく出力の扱いも決めたい。生成 AI の出力は、そのまま正しいとは限らない。個人情報保護委員会も、生成 AI の出力には不正確な内容の個人情報が含まれるリスクがあると注意している。したがって、次のような用途では必ず人手確認を入れた方がよい。

- 顧客へ送る文面
- 契約、規程、法務文書
- 対外公開する説明文
- 評価、査定、懲戒に関わる文面
- セキュリティ設定やコード変更提案

著作権や出典確認も同じである。生成 AI はもっともらしい文章を作るが、出典や権利関係が自動で正しく整理されるとは限らない。第26章では、出力は下書きであり、確定文ではないという姿勢を強く置いた方がよい。

## 権限、ログ、監査、過剰共有の管理

生成 AI の大きな論点は、入力情報だけではない。権限、共有、ログ、監査も同じくらい重要である。

ここで見落としやすいのが、生成 AI は新しい権限を作るより、既存の広すぎる権限を目立たせることが多いという点である。Microsoft は Microsoft 365 Copilot の oversharing を、導入時の代表的なリスクとして扱っている。理由は単純で、Copilot は既にアクセスできる情報を横断的に拾いやすくするからである。つまり、SharePoint や OneDrive の共有が広すぎれば、今まで探さないと見つからなかった情報が、AI によってすぐ見つかる情報になる。

したがって、社内文書検索型の AI を入れる前には、次を見直したい。

- 誰でも見えるフォルダや共有リンク

- 全社共有に置かれた個人資料
- 退職者の所有物や放置共有
- 管理部門だけの資料が広く見える状態
- guest や外部共有が残っている場所

AI 導入前の点検は、次の表で回すと漏れにくい。

点検対象	何を見るか	放置時の問題
誰でも見えるリンク	anyone link や全社公開の共有	AI が横断的に見つけやすくなる
全社共有フォルダ	個人資料や管理部門資料の混在	本来狭いべき情報が広く露出する
退職者や異動者の所有物	owner 不在、放置共有、継承漏れ	誰も閉じずに残り続ける
guest / 外部共有	期限切れでない招待、委託先アクセス	社内検索と外部共有が重なる
connector の権限	Visible to everyone や広すぎる source scope	意図しない横断検索になる

Microsoft Copilot connectors の資料も、permission setting が intended visibility model と合っているかを確認する必要がある、Visible to everyone は oversharing につながると明示している。さらに、access permissions は後から編集できず、必要なら connector を削除して作り直すことになる。つまり、AI を入れる前の本当の準備は、プロンプト研修より ACL の掃除である。

Google でも同じである。Gem sharing は Google Drive の共有設定に乗る。Google は Shared Gems are stored and shared in Google Drive と明記しており、外部共有を許していれば Gems も外へ共有されうる。さらに、Gem sharing を off

にしても、既に共有された Gem は Drive から引き続き共有されうる。したがって、カスタム AI の共有は、その機能単体ではなく、Drive や既存共有設定と一体で管理する必要がある。

OpenAI の GPTs でも、workspace owners は GPT sharing level、third-party GPTs、GPT actions の domain を制御できる。ここで重要なのは、**社内 GPT**と**外部 GPT**を分けることだ。third-party GPTs は custom actions や internet browsing など、workspace GPT settings では統制しきれない能力を持ちうる。第26章では、次を分けて管理したい。

- 本体チャット
- 社内用カスタム GPT / Gem
- 外部提供の GPT
- actions / apps / connectors
- 社内文書検索や grounded search

ログと監査も、サービスごとの差が大きい。Microsoft 365 Copilot Chat は auditing / eDiscovery のために interaction を記録するとしている。Google Workspace は Gemini の audit logs を提供している。OpenAI も Business / Enterprise で workspace settings、usage analytics、Compliance API、custom GPT の所有者管理を持つ。だが、保持期間や取得できる粒度は同じではない。したがって、第26章では **使えるか** の前に **誰が何をしたか**後で**追えるか**を確認すべきだと書きたい。

さらに、カスタム GPT や Gem は **個人の便利設定** として放置しない方がよい。OpenAI の shared edit access は、共同管理、ownership reassignment、unowned GPTs の把握を前提にしている。これは示唆的である。つまり、業務で使うカスタム AI は、台帳に載せ、所有者、編集者、共有範囲、外部接続、停止方法を持つべきなのである。

最小の台帳は、次の形で十分である。

名称	所有者	用途	共有範囲	外部接続	ログ / 監査	停止方法	引き継ぎ時確認
社内 FAQ GPT	情シス	情報システム手順の検索補助	情シス部門のみ	なし	管理画面と利用ログ	公開停止と共有解除	owner と editor の再割当
営業メモ要約 Gem	営業企画	会議メモの下書き	営業企画のみ	なし	Gemini 側の監査可否を記録	Drive 共有解除と削除	Drive owner と共有先を確認

## AI の利用をめぐるサイバーリスク

生成 AI のセキュリティは、**入力情報が漏れる** だけではない。出力、接続、外部コンテンツの取込みでも崩れる。

まず避けたいのは、hallucination を事実として扱うことだ。生成 AI は自然な文章を出すか、その文章の正しさまでは保証しない。特に、FAQ 回答、規程解釈、契約要約、障害原因分析、設定提案のような用途では、人手確認が必須である。

次に、prompt injection を意識したい。これは、メール、添付、Web ページ、共有文書、外部サービスから取り込んだ内容の中に、AI の挙動を変える指示や誘導が埋め込まれる問題である。第26章では、ここを難しい研究話としてではなく、**信頼していない外部コンテンツを AI に読ませる時の注意**として書く方が実務的である。特に次は危ない。

- 外部から届いたメール本文の要約
- 添付ファイルの自動処理

- Web 検索付きの要約
- 外部アプリと連携する action
- 社内外が混ざる shared knowledge

さらに、actions や apps は便利だが、同時に第三者送信の入口でもある。OpenAI も、GPT が external APIs や apps を使う場合、入力の一部が third-party service へ送られることがあると明示している。したがって、**本体チャットは承認**と **外部 app 連携も承認**は同義ではない。ここを分けないと、業務データが思わぬ外部サービスへ流れる。

外部接続や検索拡張は、次のように分けて見たい。

機能	何が起きるか	最低限の統制
Web grounding / browsing	Web 検索サービス側へ query が送られる	不要なら off、検索付き回答は出典確認
社内文書検索	既存 ACL のまま横断検索される	共有整理後に有効化、対象範囲を限定
actions / apps	第三者 API へ入力の一部が送られる	別承認、許可 domain、read-only から開始
connectors	外部 source の権限を AI が継承する	source scope を限定、広すぎる公開を禁止
third-party GPT	社内基準外の custom capabilities を持ちうる	owner-approved only、未承認 GPT は禁止

生成 AI の出力自体もリスクを持つ。誤送信しやすい文面、もっともらしいが誤った要約、権利侵害の可能性がある文章、攻撃的な返信案などは、利用者が **AI が言ったから**と採用すると事故になる。だから第26章では、AI のリスクを **利用者教育**だけでなく、**使わせる範囲の設計**と **出力確認が必要な用途の明示**で受ける方がよいと書きたい。

## 小さく始めて継続監視する

生成 AI は、いきなり全社展開しない方がよい。理由は、効果の測定より先に、共有や権限やログの崩れが出やすいからである。Microsoft の oversharing blueprint も、試行導入、展開、運用の段階で考える。第26章でも、この進め方を採った方がよい。

最初の試行導入は、低リスクで効果が見えやすい用途が向く。たとえば次のようなものだ。

- 公開済み資料の要約
- 社内規程や FAQ の検索補助
- 会議メモの下書き
- 定型メールの下書き
- 情シス内部のナレッジ整理

逆に、最初から避けたいのは次のような用途である。

- 顧客個人情報を含む問い合わせ対応
- 人事評価や懲戒の文案
- 契約確定文の自動生成
- 外部システムを更新する agent
- 権限付与や送金に関わる自動判断

試行導入では、対象部門、対象データ、対象サービス、管理者、評価基準、停止条件を決めたい。評価基準も、使った人数 だけでは弱い。次のようなものが実務的である。

- どの用途で時間短縮が出たか

- ルール違反入力起きていないか
- ログで追えるか
- 共有事故や過剰共有が出ていないか
- 利用者が出力を人手確認しているか
- 続ける価値があるか

ここで重要なのは、利用率だけで成功判定しないことだ。使われていても、機密が混ざる、監査できない、権限が広すぎるなら、その導入は未完成である。

pilot は、次の表まで決めてから始めたい。

項目	最低限決めること
対象部門	5人前後の限定部門、責任者1人
対象データ	公開済み資料や社内FAQなど低リスク情報
対象機能	本体チャットのみか、検索までか、actionsを切るか
評価基準	時間短縮、違反入力ゼロ、ログ確認可、共有事故ゼロ
停止条件	個人情報入力、監査不可、外部共有事故、重大な誤回答

## 通常時の例と、崩れた時の例

通常時の例では、情シスと経営企画の少人数で、社内規程、情報システム手順、公開済みFAQだけを対象にしたAI検索を試行導入する。対象データは個人情報や契約情報を含まないものに絞る。共有範囲は試行導入メンバーだけにし、外部GPT、actions、appsは無効にする。ログ取得可否を確認し、1か月ごとに入力違反、誤回答、利用満足度、時間短縮をレビューする。これなら、第26章で言いたい**限定部門 限定データ 限定機能**がそろろう。

別の通常時の例では、Microsoft 365 Copilot を営業支援チームの 5 人へ試行導入する。ただし、その前に OneDrive と SharePoint の共有設定を点検し、誰でも閲覧 や広すぎる共有リンクを是正する。Copilot は 今まで見えなかった情報を新たに解放する のではなく 既に見える情報を見つけやすくする ので、先にデータ側を整える。導入後は prompts や outputs の監査導線も確認する。これで、AI 導入と権限整理が一つの流れになる。

崩れた時の例では、社員が個人向け生成 AI で議事録要約を始め、次第に顧客名や商談内容も入力し始める。会社側には承認済みサービス一覧も入力禁止情報もなく、何がどこへ入ったか追えない。別の例では、社内用 GPT を一人の担当者が作り、actions で外部 SaaS と接続していたが、退職時に所有者整理がされず、どのデータがどこへ送られていたか分からなくなる。さらに別の例では、Gem sharing を許したまま Drive の外部共有も広く、社内用のつमोरの Gem が外へ共有される。問題は AI を使ったことではなく、使う条件 を決めずに広げたことである。

## 最低限ここまではやる

第26章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 承認済みサービスと未承認サービスを分ける
2. 入力禁止情報と条件付き許可情報を一枚で定める
3. 外部 GPT、actions、apps、connectors の扱いを決める
4. ログ、保持期間、監査手段を確認する
5. 低リスク用途だけで試行導入を始める

この五つがあるだけで、生成 AI はかなり管理しやすくなる。生成 AI とは、便利なチャットを増やすことではない。会社がどの条件で使ってよいかを決め、その条件の中で価値を出すことである。

## 今日、今週、後でやること

今日やることは、いま社内で行われている生成 AI サービスを洗い出し、承認済み未承認 不明 に分けることである。あわせて、個人情報、顧客情報、秘密情報、認証情報を入力禁止情報として仮置きしたい。

今週やることは、承認済みサービスの一つを選び、対象ユーザー、対象用途、入力可能情報、外部接続可否、ログ取得可否、保持期間を一枚で整理することである。その上で、低リスク用途に絞った試行導入を設計する。

後でやることは、カスタム GPT や Gem や社内検索型 AI を業務資産として台帳化し、所有者、編集者、共有範囲、接続先、停止手順、退職時引き継ぎまで整えることである。ここまでできると、第26章の AI ガバナンスは 禁止一覧 ではなく、使える形で守る運用になる。

## 第27章

## ひとり情シスの働き方、引き継ぎ、次の一手

---

本書の最後に置きたいのは、ひとり情シスは大変だが頑張ろう という話ではない。むしろ逆である。ひとり情シスの仕事は、頑張るほど一人に集まりやすい。問い合わせも、管理者権限も、契約更新も、ベンダー連絡も、障害初動も、社内の相談も、最後には **あの人しか分からない** に寄っていく。これは能力の高さの証明にも見えるが、会社にとっては停止リスクでもある。

第27章で強調したいのは、ひとり情シスの完成形は **全部知っている人** になることではないという点である。本当に目指したいのは、自分が不在でも最低限回る状態を作ることだ。長期休暇でも、急病でも、異動でも、退職でも、会社に知識と運営が残る。その状態を作るには、仕事の持ち方、時間の使い方、引き継ぎ資料、代替体制、次に進める改善テーマを意識的に設計し直す必要がある。

### 何を自分で持ち、何を任せるか

ひとり情シスが最初に見直したいのは、**自分が全部やる前提** で仕事を抱えていないかである。ここで大事なのは、仕事を **作業** の単位だけで見ないことだ。NISTのNICE Frameworkが示す通り、仕事はroleやtaskのまとまりで見の方が整理しやすい。つまり、**PC** を配る **権限を付ける SaaS** を契約するを別々の雑用として持つのではなく、どこに判断責任があり、どこが単純実行で、どこが他者へ渡せるのかを先に分ける。

実務では、次の四つに分けるとよい。

- 自分で決める仕事

- 他者に実行してもらう仕事
- 自動化する仕事
- やめる仕事

四分けは、次の表で見ると混ざりにくい。

区分	自分に残すもの	渡す先や仕組み	代表例
自分で決める仕事	基準、優先順位、説明責任	実行は他者や仕組みに渡せる	高権限付与基準、導入判断、重大時のエスカレーション
他者に実行してもらう仕事	最終承認、例外判断	人事、部門長、経理、ベンダー	入社情報回収、一次請求確認、保守作業
自動化する仕事	入力条件、例外時の止め方	フォーム、台帳、定型処理	定型依頼、棚卸し通知、月次集計
やめる仕事	廃止判断、代替確認	なし	見ていない報告書、使っていない二重記録

**自分で決める仕事** には、たとえば次が入る。

- 重要システムの導入判断
- 高権限の付与基準
- 外部委託の責任分界
- 障害や重大インシデント時のエスカレーション判断
- 経営へ上げるべきリスクや予算の説明

ここは、ひとり情シスが持つべきである。単なる作業ではなく、会社としての説明責任があるからだ。

一方で、**他者に実行してもらう仕事**は意外に多い。たとえば、入社者情報の回収は人事、アプリ利用申請の妥当性判断は部門長、経費や請求書の一次確認は経理、保守作業はベンダー、専門的なネットワーク調査やフォレンジックは外部専門家へ渡せる。ここでのポイントは、**全部外に出す** ことではない。判断は社内に残し、実行を渡すことである。

さらに、**自動化する仕事** と **やめる仕事** も同じくらい重要である。毎回同じ内容のアカウント発行依頼を手入力しているなら、申請フォームと定型化でかなり減る。毎月見ていない報告書を手で作っているなら、その仕事はやめた方がよい。ひとり情シスでは、**やるべきか** を見直さずに **どう早くやるか** だけ考えると、忙しさが固定化する。

ここで一つ意識したいのは、**自分しかできない仕事** と **自分しかやっていない仕事** は違うという点である。後者は、仕組みを作っていないだけのことが多い。第27章では、この差を見抜けるようになることが重要である。

## 忙しさに飲まれない仕事設計

ひとり情シスの忙しさは、件数の多さだけでは決まらない。**いつでも割り込まれるどこから依頼が来るか決まっていない** **同じ種類の仕事を毎回ばらばらに処理する** **期限管理が頭の中にしかない** という設計だと、件数がそれほど多くなくても疲弊する。

したがって、第27章では働き方を気分の問題ではなく、仕事設計の問題として見たい。最初に整えたいのは、依頼の入口を一つに寄せることである。第2章や第16章でも触れた通り、依頼窓口がチャット直投げ、口頭、個人メール、会議ついでに散っていると、優先順位も証跡も崩れる。ひとり情シスほど、入口は一つに寄せた方がよい。

次に、仕事を一定のリズムで持つ。たとえば、次のように切る。

- 毎日: 新規依頼の一次確認、緊急度判定
- 毎週: 滞留案件と未完了の変更作業の見直し
- 毎月: 契約更新、ライセンス棚卸し、主要ログ確認
- 四半期: 権限棚卸し、ベンダーレビュー、改善テーマの進捗確認

このリズムは、何を見るかと何を残すかまで決めると崩れにくい。

頻度	見るもの	最低限残すもの
毎日	新規依頼、緊急度、止まっている案件	緊急案件メモ、保留理由
毎週	滞留案件、未完了変更、例外対応	次週へ持ち越す案件一覧
毎月	契約更新、ライセンス、主要ログ、請求差異	月次確認メモ、更新判断
四半期	権限棚卸し、ベンダーレビュー、改善進捗	直すこと、続けること、やめること

これを決めておくと、**思い出した時にやる** から抜けられる。ひとり情シスでは、作業力より、忘れにくい構造を作る方が効く。

また、似た作業はまとめて処理した方がよい。新規入社の手末準備を一件ずつ都度対応するより、毎週決まったタイミングでまとめる方が安定する。ソフトウェア申請、共有フォルダ権限、貸与品の更新も同じである。まとめ処理にするだけで、割り込みはかなり減る。

さらに、**集中して考える時間** を守る必要がある。導入判断、変更計画、障害レビュー、規程見直しのような仕事は、空き時間の切れ端では進まない。毎日一時間でもよいので、連絡を返す時間と、考える時間を分ける方がよい。忙しい会社ほど、ここを守らないと、いつまでも運用の火消しだけで終わる。

忙しさは美德ではない。忙しい状態が続いているなら、努力不足より先に、入口、標準化、頻度、例外の扱いに設計不良がないかを見る方がよい。

## 引き継ぎ資料と後継者不在リスク

ひとり情シスで最も危ないのは、後継者がいないことそのものではない。いま自分が止まったら、会社がどこを見ればいいのか分からない状態である。したがって、後継者採用の前に、最低限の引き継ぎ束を作る必要がある。

ここでよくある失敗は、完全版マニュアルを目指して何もできないことだ。だが、最初に必要なのは全システムの詳説ではない。緊急時に見れば最低限回せる一式である。

Atlassian の KCS や ServiceNow の knowledge management が示す通り、知識は対応の後にまとめて書くのではなく、対応の中で残し、使いながら直す方が現実的である。第27章で作りたい引き継ぎ束も、最後の宿題ではなく、日々の運用から育てる前提で持つ方がよい。

最低限の引き継ぎ資料には、次を入れたい。

- 主要サービス一覧と重要度
- 管理画面や契約の正式名称
- 管理者権限の所在
- 緊急用アカウントの有無と利用条件
- 秘密情報や復旧手段の保管場所
- ベンダー、保守、リセラー、ISP の連絡先
- ドメイン、証明書、回線、SaaS の更新期限
- バックアップと復元テストの状況
- 日次、週次、月次で見るべきもの

- 障害やインシデント時の最初の連絡先

最初の引き継ぎ束は、次の表で十分である。

項目	最低限書くこと	なぜ要るか
主要サービス一覧	正式名称、重要度、用途	何が止まると困るかをすぐ把握するため
管理者権限	管理者アカウント、緊急用アカウント、利用条件	権限回復の入口を失わないため
復旧情報	秘密情報の保管場所、回復手段、利用時の連絡先	人の記憶頼みを避けるため
連絡先	ベンダー、保守、リセラー、ISP、社内決裁者	深夜や不在時でも連絡を始めるため
更新期限	ドメイン、証明書、回線、SaaS、保守	自動更新や失効を防ぐため
定期確認	日次、週次、月次で見るもの	引き継いだ人が何を見ればよいか分かるため
初動手順	障害、重大インシデント時の最初の連絡順	技術判断と経営判断を混ぜないため

この一式があるだけで、長期休暇前の安心感はかなり変わる。

管理者権限の代替体制は、特に早く整えたい。Google は複数の super admin アカウントを別々の個人へ割り当てることを勧めている。Microsoft も、クラウド専用の緊急用アカウントを二つ持つことを勧めている。ここから分かるのは、**管理者権限が一人にしかない**状態は、小さな会社でも避けるべきだということである。

また、管理者アカウントは日常利用アカウントと分けた方がよい。普段のメールや Web 閲覧に高権限アカウントを使うと、誤操作もフィッシングも一気に重くなる。第8章でも扱ったが、第27章ではこれを **セキュリティの話** にとどめず、**引き継ぎしやすさ** の話として見る。高権限が用途ごとに分かれていれば、後から見た時に構造が分かりやすい。

Google と Microsoft の一次情報を並べると、最低限の代替経路は次のように整理できる。

領域	Google Workspace	Microsoft Entra	実務上の意味
複数管理者	super admin を複数人へ分ける	cloud-only emergency access accounts を二つ持つ	一人依存を避ける
日常利用分離	super admin を daily activities に使わない	Global Administrator 用の別アカウントを持つ	高権限の日常利用を避ける
復旧前提	recovery options、backup codes、spare security key、DNS ownership verification を準備する	認証情報を secure location に保管し、通常時は使わない	人の記憶でなく会社の保管へ移す
監査	admin log events を見る	sign-in monitoring と通知を前提にする	利用時に追えるようにする

緊急用アカウントや回復情報の保管も、一人の記憶や個人端末に置かない方がよい。たとえば、管理部門の責任者がアクセスできる安全な場所に、利用条件と連絡フローつきで保管する。利用したら必ず記録する。高優先度アラートや監査ログが取れるなら、そこまで整える。重要なのは、**誰でも使える** ではなく **必要な時に認可された人が使える** 状態にすることである。

そして、後継者不在リスクを考える時は、**次の情シス担当がまだいない** ことと、**いま誰も代替できない** ことを分けて考えたい。前者は採用や体制の問題だが、後者は今日から下げられる運営リスクである。

## 相談相手と外部ネットワークの作り方

ひとり情シスは、一人で実務を回すことはあっても、一人で全部判断するべきではない。にもかかわらず、現場では **困ったら自分が考える** が常態化しやすい。これを防ぐには、相談先を事前に設計する必要がある。

まず、社内の相談先を整理する。たとえば、次は分けて持ちたい。

- 予算や契約条件の判断を上げる相手
- 個人情報や労務に関わる相談先
- 業務影響や優先順位を決める部門責任者
- 緊急時に利用停止や対外連絡を判断する経営層

これが決まっていないと、障害やインシデントの時に、技術判断と経営判断が混ざる。

次に、外部の相談先も **誰に何を聞くか** まで整理したい。候補はたとえば次である。

- Microsoft や Google のリセラー、販売代理店
- ネットワーク保守や回線事業者
- セキュリティ事故時に支援を頼める専門会社
- 個人情報や契約確認を相談できる法務や社労士
- 請求や支払い影響を見られる経理担当

相談先一覧は、次の列があると使いやすい。

相談先	何を聞くか	一緒に残す情報
経営層、管理部門	予算、停止判断、対外連絡	決裁条件、連絡順、代行者
法務、労務、個人情報窓口	契約、労務、個人情報の扱い	相談時に必要な資料、記録先
リセラー、ベンダー、保守	製品仕様、障害、保守作業	契約番号、受付時間、サポート範囲
回線、ISP、ネットワーク保守	回線障害、機器障害	契約名義、回線番号、拠点情報
外部専門家	重大インシデント、フォレンジック、復旧支援	緊急連絡先、初動で渡す情報

ここで大事なものは、連絡先だけでなく、契約番号、受付時間、一次切り分けに必要な情報、緊急時の連絡順まで残すことである。電話番号だけあっても、深夜障害で契約名義が分からなければ進まない。

さらに、同じ立場の人と話せる外部ネットワークも持っておく方がよい。必ずしも formal な組織である必要はない。リセラーの担当者、地域の IT コミュニティ、同業の管理部門、情報交換できる社外の知人でもよい。重要なのは、**自分の会社の常識だけで判断しない**ための比較軸を持つことである。

相談は、困ってから探すと遅い。第27章では、相談先一覧も引き継ぎ資料の一部として持つべきだと書きたい。

## 次に整えるべきテーマを選ぶ

本書をここまで読むと、直したい場所はたくさん見つかる。だが、ひとり情シスが全部を同時に進めると、だいたい崩れる。大事なものは、**何が抜けているか**を知ることより、**次にどれから埋めるか**を決めることである。

ここで役立つのが、現状と目標の差分で考えるやり方である。NIST CSF の Current Profile と Target Profile の考え方を使えば、いまはどうか次にどこまで持っていきたいかを見比べられる。CISA の CPG も、中小規模組織では高い効果が見込める少数の重要対策を優先する考え方を示している。つまり、次年度計画は網羅性より順序である。

実務では、次の四つで見ると絞りやすい。

- リスクをどれだけ下げるか
- 不在時でも回る度合いをどれだけ上げるか
- 毎月の工数をどれだけ減らせるか
- 属人化をどれだけ減らせるか

ここは、現状と目標の差を一枚で持つと決めやすい。

テーマ	現状	6～12 か月後の目標	差	最初の一手
管理者権限の代替体制	管理者が一人に集中している	代替経路と緊急用アカウントがある	不在時の停止リスクが高い	主要サービスの管理者数を洗い出す
契約と更新期限	個人カレンダーやメールに散在	更新期限と担当が一覧化されている	自動更新や失効が起きやすい	契約一覧へ解約期限を入れる
依頼の入口	口頭、チャット、個人メールに散在	窓口が一つで優先順位が見える	証跡と標準化が崩れている	申請先を一つに寄せる

たとえば、候補が十個あったとしても、次の6か月から1年では三つくらいに絞った方がよい。たとえば次のような組み合わせである。

- 管理者権限と緊急用アカウントの代替体制を整える
- 端末、アカウント、契約の台帳を一本化する

- 問い合わせの入口と定型依頼を標準化する

これなら、リスク、継続性、工数削減の三方向を同時に押せる。

逆に、よくない進め方は、**思いついた改善を全部未整理のまま積み** ことである。項目が 30 個あっても、それは計画ではない。やる順、終わりの条件、担当、依存関係がない限り、ただの不安一覧である。第27章では、**テーマは少なく、条件は具体的に** を強めに置きたい。

## 通常時の例と、崩れた時の例

通常時の例では、長期休暇の二週間前に、ひとり情シスが **緊急時だけ見る引き継ぎ束** を一つにまとめる。そこには、主要サービス一覧、管理者権限の場所、緊急用アカウント、ベンダー連絡先、更新期限、障害時の最初の連絡先が入っている。Google Workspace では super admin が複数あり、Microsoft では緊急用アカウントが整備され、利用時の記録ルールもある。経営層と管理部門には、どの条件でどこへ連絡するかも伝わっている。これなら、担当者が不在でも会社は止まりにくい。

別の通常時の例では、今年やりたいことを十個書き出したあと、**高権限の代替体制 契約と更新期限の見える化 問い合わせの標準化** の三つに絞る。四半期ごとに進捗を見る。進めないテーマは捨てる。これにより、**何もかも少しずつ未完了** を避けられる。

崩れた時の例では、管理者権限が一人しかなく、そのアカウントを日常のメールにも使っている。バックアップコードも回復情報も本人しか知らない。別の例では、ドメイン管理が昔の制作会社の個人アドレスに紐づいたままで、証明書更新もその人しか分からない。さらに別の例では、契約更新日と解約期限が個人カレンダーにしかなく、退職時に何が自動更新されるか誰も把握できない。どれも問題は専門知識不足ではない。個人依存を運営へ変えていないことにある。

## 最低限ここまではやる

第27章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 仕事を自分で決める 他者に実行してもらおう 自動化する やめる に分ける
2. 管理者権限の代替経路と緊急用アカウントを整える
3. 緊急時に見る引き継ぎ束を一つ作る
4. 相談先、ベンダー、更新期限を一覧化する
5. 次の6か月から1年の改善テーマを三つ以内に絞る

この五つがあるだけで、ひとり情シスはかなり続けやすくなる。大事なのは、全部を自分で抱えたまま強くなることではない。会社に知識と運営を残し、自分しか回せない状態を減らすことである。

## 今日、今週、後でやること

今日やることは、いま自分が抱えている仕事を書き出し、判断 実行 自動化候補 やめる候補 に分けることである。あわせて、管理者権限が一人しかないサービスがないかを確認したい。

今週やることは、引き継ぎ束の初版を作ることである。完璧でなくてよい。主要サービス、管理者権限、緊急連絡先、更新期限、秘密情報の保管場所だけでも先に一か所へ集めたい。

後でやることは、次の6か月から1年の改善テーマを三つに絞り、四半期ごとの見直し日を決めることである。ここまでできると、第27章の終わり方は 頑張り続ける ではなく、続けられる形に変える になる。

付録

# すぐ使えるひな型とチェックリスト

台帳、入退社運用、有事初動をそのまま実務へ落とし込む。

全3本

## 付録A

# 台帳ひな型

付録Aでは、本書で繰り返し登場した台帳の最小ひな型をまとめる。最初から完璧な CMDB を作る必要はない。重要なのは、**何があるか** **誰が持つか** **いつ見直すか** が後から追えることである。

## A-1 サービス台帳

項目	何を書くか
サービス名	正式名称、略称
用途	何の業務で使うか
重要度	高、中、低
管理者	主担当、代替担当
提供元	ベンダー名、契約窓口
契約形態	月額、年額、自動更新など
更新期限	契約更新日、解約期限
認証方式	ID/PW、SSO、MFA
保管データ	個人情報、契約情報、ログなど
バックアップ	取得有無、復元方法
障害時連絡先	ベンダー、社内判断者

## A-2 アカウント台帳

項目	何を書くか
対象サービス	Microsoft 365、Google Workspace など
アカウント種別	個人、共有、管理者、サービス
所有者	利用者名、部門
権限	一般、管理者、限定管理者など
MFA	有効、無効、例外理由
代替管理者	緊急時に使える人
棚卸し日	最終見直し日
状態	有効、停止、退職処理中、削除済み

## A-3 端末台帳

項目	何を書くか
資産番号	社内管理番号
種別	ノート PC、スマホ、タブレット
利用者	氏名、部門
OS	Windows、macOS、iOS、Android
管理方法	MDM、手動、未管理
配布日	貸与日
保証期限	保守、保証、リース期限
状態	利用中、予備、修理中、退役
備考	紛失、交換履歴など

## A-4 契約台帳

項目	何を書くか
契約名	サービス名、保守契約名
契約先	ベンダー、販売代理店
契約番号	見積番号、注文番号、契約番号
金額	月額、年額、従量
更新条件	自動更新、都度更新
解約期限	いつまでに止める必要があるか
支払方法	請求書、カード、口座振替
社内担当	契約管理者、経理窓口
関連サービス	どのシステムに紐づくか

## A-5 連絡先台帳

項目	何を書くか
相手先	ベンダー、ISP、保守会社、法務など
目的	障害、契約、請求、法務相談など
連絡手段	メール、電話、ポータル
受付時間	平日、24時間など
契約情報	契約番号、顧客番号
連絡条件	どの状況で連絡するか
社内判断者	誰の承認で連絡するか

## A-6 最小構成で先に持つべき台帳

最初から全部を細かく作らなくてよい。まずは次の四つを優先するとよい。

- サービス台帳
- アカウント台帳
- 端末台帳
- 契約台帳

この四つがそろっただけで、入退社、障害、更新、引き継ぎの多くがかなり見えやすくなる。

## A-7 台帳を運用へ乗せる最小ルール

台帳は、項目だけあっても回らない。最初に **誰が更新するか** **どの頻度で見直すか** **一行目をどう書くか** を決めておくと、作って終わりになりにくい。

台帳	主な更新者	見直し頻度の目安	一行記入例
サービス台帳	情シス主担当、必要に応じて業務部門責任者	月次、又は契約更新前	Google Workspace / メール、Drive、Chat / 高 / 情シス 山田、副 佐藤 / Google / 年額、自動更新 / 解約期限 2026-12-31 / SSO、MFA / 顧客連絡、社内文書 / 障害時はサポートケース起票

台帳	主な更新者	見直し頻度の目安	一行記入例
アカウント台帳	情シス、又はサービス管理者	入社、異動、退職の都度。月次で棚卸し	Microsoft 365 / 個人 / 営業部 田中 / 一般ユーザー / MFA 有効 / 代替管理者なし / 棚卸し 2026-03-01 / 有効
端末台帳	情シス、又は貸与品管理担当	配布、回収、交換の都度。月次で状態確認	PC-042 / ノート PC / 経理部 鈴木 / Windows 11 / Intune 管理 / 2025-11-01 配布 / 保証 2028-10-31 / 利用中
契約台帳	情シス主担当、経理窓口	月次、又は更新期限の 90 日前	Microsoft 365 Business Premium / 販売代理店 ABC / 契約番号 2025-104 / 年額 1,200,000 円 / 自動更新 / 解約期限 30 日前 / 請求書 / 情シス 山田、経理 佐藤 / 社内認証基盤
連絡先台帳	情シス	四半期、又は担当変更時	ISP 障害窓口 / 回線障害、工事、問い合わせ / 電話、ポータル / 24 時間 / 顧客番号 123456 / 主要回線断時に連絡 / 社内判断者は情シス責任者

見直し頻度で迷うなら、まず次の三つだけ決めればよい。

- 月次で、サービス台帳、アカウント台帳、契約台帳の変更有無を見る
- 配布、回収、異動、退職のたびに、端末台帳とアカウント台帳をその場で更新する

- 四半期ごとに、連絡先台帳の受付時間、契約番号、窓口変更がないか確認する

## 付録B

# 入社、異動、退職チェックリスト

---

付録Bでは、第9章を実務で使いやすい形へ落とすために、入社、異動、退職の最小チェックリストをまとめる。会社ごとに追加項目はあるが、まずは抜けやすい場所を塞ぐことを優先する。

## B-1 入社前

- 氏名、所属、役職、開始日を確定した
- 必要なアカウントを洗い出した
- 標準端末を確保した
- ライセンスの空きと追加要否を確認した
- 所属部門へ必要権限の承認者を確認した
- メールアドレス、表示名、命名規則を確認した
- 端末配布日と初回ログイン方法を決めた

## B-2 入社初日

- 本人確認をした
- 初期パスワード変更または初回サインインを完了した
- MFA 登録を完了した
- 必要最小限のグループと権限だけ付与した
- 端末の受領確認を取った

- 問い合わせ窓口と基本ルールを案内した
- 共有アカウントや代表アドレスの扱いを説明した

### B-3 入社後1週間以内

- 利用していないアカウントやライセンスがないか確認した
- 過剰権限が付いていないか確認した
- 端末やスマホの管理状態を確認した
- 必要な共有先だけ見えているかを確認した

### B-4 異動時

- 異動日と新旧所属を確認した
- 新しい業務に必要な権限を承認ベースで付与した
- 旧所属で不要になった権限を剥奪した
- 共有フォルダ、メーリングリスト、チャンネル参加を見直した
- 管理者権限や例外設定が残っていないか確認した
- 利用端末や貸与品の変更有無を確認した

### B-5 退職前

- 最終入社日と雇用終了日を確認した
- 当日停止が必要か、引き継ぎ後停止でよいかを決めた
- データ引き継ぎ先を決めた
- 共有アカウント、代表アドレス、外部サービスの所有を確認した
- 貸与品の回収方法を決めた

- 委託先や外部ゲストの終了時は接続先も洗い出した

## B-6 退職当日

- サインインを遮断した
- セッションやトークンを失効した
- MFA 手段や回復手段を無効化した
- 管理者権限を剥奪した
- VPN、Wi-Fi 証明書、MDM 登録を停止した
- 貸与 PC、スマホ、IC カード、物理鍵を回収した

## B-7 退職後

- メール転送や自動返信の可否を決めた
- ファイル所有権や共有設定を整理した
- アーカイブ、停止、削除のどれにするか決めた
- ライセンスを回収した
- 契約、請求、外部サービスに本人名義のものが残っていないか確認した
- 台帳と棚卸し記録を更新した

## B-8 承認者と実行者のメモ欄

この付録を実際の運用票にするなら、各チェック欄の横に **承認者** **実行者** **実施日時** の三つを足すとよい。特に権限付与、権限剥奪、データ引き継ぎ、削除判断は、**誰が決めたか** と **誰が実施したか** が後から追える方が安全である。

項目群	主な承認者	主な実行者	記録に残すこと
入社時の標準権限付与	所属部門の上長	情シス	付与したグループ、付与日時
例外権限の付与	業務責任者、又はシステム責任者	情シス、又はサービス管理者	例外理由、終了予定日
異動時の旧権限剥奪	旧所属の責任者、必要に応じて新所属責任者	情シス	外した権限、外した日時
退職時のデータ引き継ぎ	所属部門責任者、必要に応じて法務、人事	情シス	引き継ぎ先、保持期限、削除可否
ライセンス削除、アーカイブ、削除	情シス責任者、必要に応じて人事、法務	情シス	最終状態、実施日時、復元期限
貸与品回収	人事、総務、所属部門責任者	情シス、総務	回収物、回収日時、不足物

## B-9 委託先、外部ゲストで特に見ること

委託先や外部ゲストは、社員と同じ流れで見つつ、次の差分を補うと抜けが減る。

観点	社員と同じでよいこと	追加で見ること
開始時	必要サービス、必要権限、MFA、端末有無	契約期間、委託元責任者、利用終了日
権限付与	最小権限、グループ付与、記録	社内標準より強い権限を付ける理由、共有先の範囲
利用端末	管理端末かどうかの確認	私物端末、持込端末、他社管理端末の扱い
データ共有	共有先、所有権、引き継ぎ	外部共有リンク、個人メール宛て転送の有無

観点	社員と同じでよいこと	追加で見ること
終了時	サインイン遮断、権限剥奪、セッション失効	プロジェクト単位のspace、チケット、VPN、共有リンクの終了確認

## B-10 迷った時の優先順位

迷った時は、次の順で考えるとよい。

1. まずサインインと権限を止める
2. 次にデータと所有権を引き継ぐ
3. 最後にライセンス、アーカイブ、削除を整理する

退職処理で危ないのは、削除を急ぎすぎて証跡やデータを失うことである。遮断と失効を先に行い、その後に整理する方が事故が少ない。

## 付録C

# 障害、インシデント初動チェックリスト

---

付録Cでは、第21章と第22章の初動を、現場で見返しやすい最小チェックリストへ落とす。ここで重要なのは、すぐ直そうとする前に、影響、連絡、証跡、切り分けを押さえることである。

## C-1 まず共通で確認すること

- 何が起きているかを一文で書いた
- 発生時刻または発覚時刻を記録した
- 影響範囲を確認した
- 影響を受ける業務と利用者を確認した
- 単一障害か広域障害かを見た
- 一つのチケットまたは記録場所を決めた
- 連絡担当を決めた

## C-2 通常の障害初動

- 回線、認証、DNS、クラウド障害情報を確認した
- 直前の変更や更新を確認した
- 影響サービス、端末、拠点を切り分けた
- 利用者向けの一次連絡を出した
- 切り戻しが必要かを判断した

- 暫定復旧でよいか、本復旧が必要かを決めた

### C-3 セキュリティ事故の疑いがある時

- 不正アクセス、情報漏えい、ランサムウェアの可能性を切り分けた
- すぐ止めるべきアカウントや端末を隔離した
- ログ、メール、画面、アラートを保全した
- 安易に削除や再起動をして証跡を消していないか確認した
- 経営、法務、個人情報保護の判断者へ連絡した
- 外部専門家やベンダーへ支援要否を判断した

### C-4 利用者向け一次連絡

- 何が起きているかを簡潔に書いた
- 影響範囲を書いた
- いま分かっていることと未確定事項を分けて書いた
- 利用者に止めてほしい行動を書いた
- 次の更新予定時刻を書いた

### C-5 外部連絡前に確認すること

- 契約番号、顧客番号、テナント情報を手元に置いた
- 影響範囲と発生時刻を整理した
- 何を依頼したいかを一文で言える状態にした
- 社内承認が必要な連絡か確認した

## C-6 復旧後に必ずやること

- 復旧時刻を記録した
- 根本原因と直接原因を分けて整理した
- 暫定対応と恒久対応を分けた
- 再発防止の担当者と期限を決めた
- 責任追及を目的にしない振り返りを行った
- 手順書、台帳、監視、設定を更新した

## C-7 初動で避けたい行動

- 記録せずに個人チャットだけで進める
- 影響範囲が分からないまま **復旧しました** という
- 証跡保全前に削除や初期化をする
- 連絡窓口が複数に分かれて情報がずれる
- 直った後にレビューをやらず終える

## C-8 重大度の見分け方

最初から正確な severity を決め切る必要はない。だが、次の三段階を仮置きしておく、誰に上げるかを迷いにくい。

重大度	目安	すぐ上げる相手
低	単一利用者、又は限定部署だけ。代替手段があり、通常時間内に対応できる	情シス内、又は関係部門責任者

重大度	目安	すぐ上げる相手
中	複数部署に影響し、主要業務が遅れる。代替手段はあるが、利用者周知が必要	関係部門責任者、情シス責任者、必要に応じてベンダー
高	全社影響、顧客影響、長時間停止、security incidentの疑い、個人データや法務判断が絡む	経営、法務、個人情報保護判断者、広報、外部支援先

次のどれかがあれば、通常障害でも **高** 寄りで扱う。

- 顧客や取引先へ説明が必要になる
- 不正アクセス、漏えい、改ざんの可能性が残る
- 人事、会計、受発注、認証基盤など止められない業務が止まる
- 1回の復旧で戻り切らず、再発や拡大の可能性が高い

## C-9 一次連絡の最小テンプレート

一次連絡は、完全な説明ではなく、混乱を増やさないための最小情報を揃えることが目的である。次の型をそのまま使えばよい。

件名: [障害/インシデント] ○○で影響が発生しています

発生時刻:

2026-03-22 10:15 頃から確認

影響範囲:

営業部と経理部でMicrosoft 365 ヘサインインしにくい状態

現時点で分かっていること:

認証系の異常を調査中。原因はまだ未確定

利用者に止めてほしい行動:

パスワード再設定や端末初期化を行わず、そのまま待機

次回更新時刻:

11:00 までに続報を共有

障害でもインシデントでも、初動の質は **どれだけ早く全部分かるか** ではなく、**分かっていないことを含めて整理しながら進められるか** で決まる。

# 奥付

---

- 書名: ひとり情シス大全
- 状態: 本文ドラフト
- 作成日: 2026年3月21日
- 構成: 全27章、付録3本
- 備考: 製品名、管理画面名、制度、公式ガイドの版は今後の更新にあわせて見直す前提

この PDF は、本文構成と内容確認のための制作途中版である。