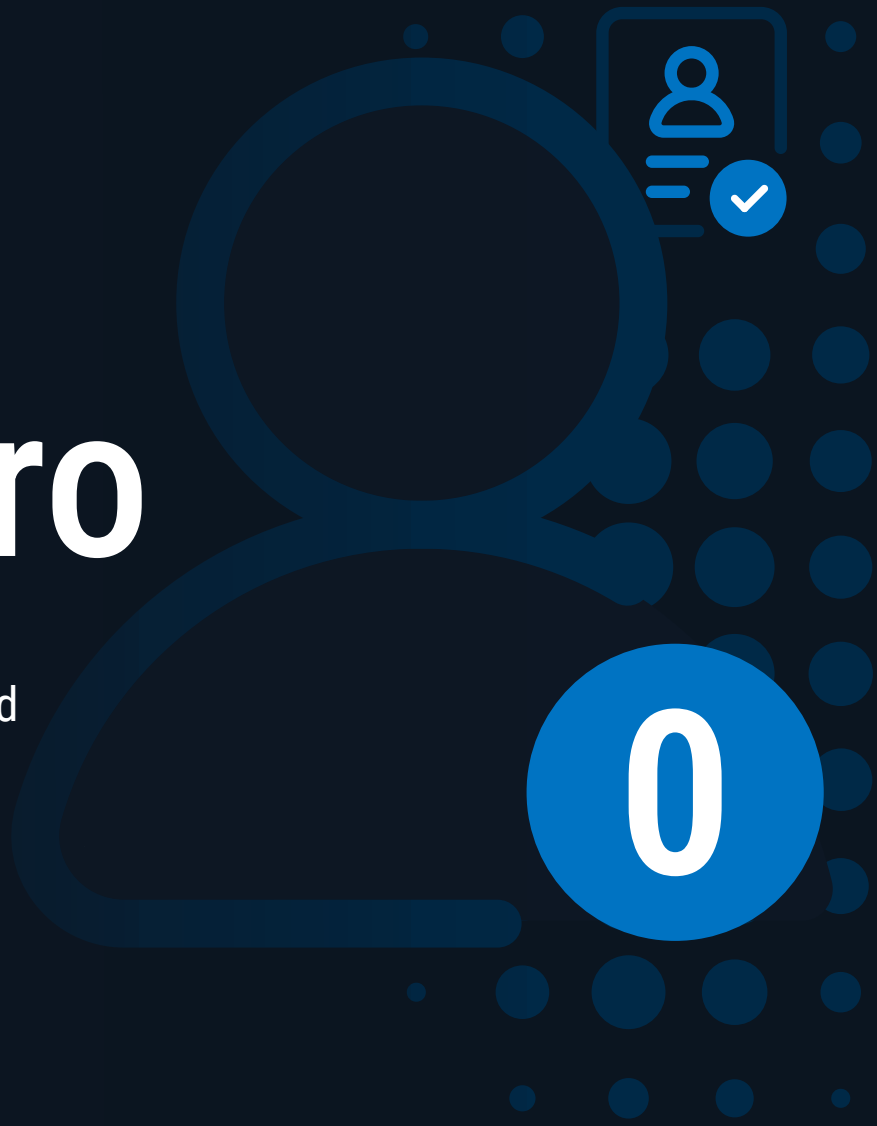
digicert®

DigiCert Customer Zero

Lessons and best practices learned from implementing and optimizing our own modernized certificate management.



eBook | 2026

Digital certificate management is a growing challenge for any large organization. As certificate lifetimes shorten and cryptographic standards evolve, the margin for manual, fragmented operations is quickly disappearing. This is the story of how we modernized our own PKI. It's what our environment looked like before we started, what we got wrong on the first attempt, what the second attempt required to achieve success, and what the results looked like at scale. We call the initiative Customer Zero.

What you'll read here follows the same sequence as the project itself, across five stages: background, planning, discovery, automation, and scaling. The process took about a year to complete, ending late in 2025. In the second half of 2025, during and after the project completion, we recorded no customer-impacting certificate-expiration outages. Customer Zero required more up-front work than most organizations expect and surfaced significantly more certificates, gaps, and ownership problems than we anticipated. Both of those things turned out to be features, not obstacles.

The central lesson is one we didn't solve with technology alone. Instead, we solved the modernizing challenge with an operating model.

1 Background

What was the state of our PKI before Customer Zero? How were certificates, both public and private, accounted for? Who, if anyone, was in charge? The answers were not up to our own standards, and we knew we had to set into practice the recommendations we make to our customers.

2 Planning

The main reason this migration proved successful was we spent a lot of time coming to an agreement on what the solution should look like. The rest of the project became much easier with a strong plan in place.

3 Discovery

A series of scans found all certificates that could be reached. The results were fed into an inventory that continues to serve as the key to keeping the PKI agile.

4 Automation

DigiCert took a standards-first approach to automation, and it worked. But there were still many certificates that needed extra effort. With a small number of exceptions, manual management processes were no longer sufficient.

5 Scaling

After a successful pilot study with one group, the rest of the company migrated group by group until a modernized PKI was running across the entire organization.

Background

How fragmented certificate operations created the need for the Customer Zero initiative

Our starting point looked a lot like many enterprises: decentralized ownership, incomplete inventories, inconsistent processes, and too much manual work.

Fragmented ownership, incomplete inventories, and manual processes created operational risk that shorter certificate lifecycles and future cryptographic transitions would make unsustainable.

Our first CLM rollout failed because teams adopted the technology without a shared architecture, governance model, or procedures, reproducing fragmentation inside the new platform.

The successful approach combined one Trust Lifecycle Manager environment and central governance with distributed execution, effectively eliminating manual renewals and unclear ownership.

Without control and direction, PKI will descend into disorganization

We created Customer Zero to use DigiCert ONE solutions for as much of our own internal and public PKI operation as is practical. Although we're a certificate authority, the operational problem wasn't unique to a CA. We had employee and device certificates, internal servers, customer-facing websites and services, cloud and network infrastructure, and certificates hosted by third parties. These are the same categories that complicate certificate management in other large enterprises.

The defining weakness in PKI operations wasn't an absence of technical skill. It was the absence of a single inventory, common policy, and clear line of responsibility across the organization.

This is the way things happen in growing enterprises. We were founded in 2003 and since that time, we've built up a large organization that operates across the globe, in many time zones and legal jurisdictions. It's a challenging environment for enterprise PKI, and the tools to bring order to these systems are only recently available.

The operational consequences



Before Customer Zero, fragmentation created work at every stage of the lifecycle. To enroll a device, employees could be required to visit a self-service portal and download a certificate before connecting to the corporate network. New hires and certificate expirations generated waves of service desk tickets when devices lost access. Server teams manually requested, installed, bound, and verified certificates. Public services could use the same DNS name across a web application firewall, load balancers, and multiple NGINX servers, with separate certificate instances and deployment actions at each layer.

The most dangerous certificates were not necessarily the most visible. They were the certificates with no acknowledged owner. A known certificate on a critical service could usually be renewed under pressure. An unknown certificate could expire while every team assumed that someone else was responsible. An unassigned certificate pool eventually contributed to an internal chat-service outage when an expiration notice didn't reach an accountable owner.

Third-party control added another boundary. For services hosted by companies like Adobe, Moodle, and Oracle, we could obtain the certificate but couldn't complete installation or renewal without the provider. Legacy infrastructure added still more variation: appliances with proprietary APIs, hardware that accepted only self-signed certificates, and systems that could import a certificate but couldn't automate binding or restart behavior.

Why the old model proved unsustainable

Impending changes to PKI have removed any remaining tolerance for manual operation. Public TLS certificate validity is moving from roughly annual cycles toward 47 days. At our current scale of about 18,000 certificates, 47-day lifetimes require more than 140,000 certificate actions per year. Relying on an expert to remember an annual task won't survive that frequency.

The problem also extended beyond renewal. An enterprise cannot rotate algorithms, respond quickly to a compromised key, or prepare for post-quantum cryptography if it can't identify where certificates are deployed and who controls them. The difficulty customers had in the transition away from SHA-1 was a warning of this problem. Inventory, ownership, policy, and automation are parts of the same crypto-agility challenge rather than separate administrative improvements.

The first attempt

Our first attempt failed to shift company operations to DigiCert ONE, owing to mistakes made in planning. In fact, it was a general lack of planning that doomed the project.

In the first attempt, teams were told to move to DigiCert Trust Lifecycle Manager (our certificate lifecycle management solution) within a limited period, but no central owner had first defined the target architecture, business-unit structure, policies, or common procedures. Each team created its own DigiCert ONE account and implementation. Some received help from the product or engineering groups while others proceeded largely on their own. About half used one platform environment and half another. They configured integrations, agents, and automation differently while connecting to the same DigiCert CertCentral account.

That connector imported the full CertCentral account into each implementation. Teams therefore saw one another's certificates and tried to filter away the noise. When the resulting inventories were compared, the same certificates appeared two, three, or four times. Some data was never included in central reporting at all. A mandate intended to centralize operations had reproduced the original fragmentation inside a new tool.

We learned a lot from this failure, both about our own operations and the right way to execute a project as big as moving an enterprise to a single CLM solution.

What we learned helped us develop a design that would lead to success in the second attempt. Specifically, we realized the objective was no longer to simply adopt a tool. It was to establish a common set of systems and procedures for discovering, owning, issuing, deploying, renewing, and reporting on certificates throughout DigiCert, and with automation implemented wherever possible. Most importantly, it began with a long and thorough planning process.



A mandate intended to centralize operations had reproduced the original fragmentation inside a new tool.

The Customer Zero operating model

Customer Zero placed central policy and administration in our Trust Office while leaving day-to-day execution with operating teams organized in Trust Lifecycle Manager business units. The Trust Office established the rules, approved exceptions, managed the enterprise structure, and reported progress. Operating teams retained the application and infrastructure knowledge needed to deploy certificates safely.



The program made discovery a prerequisite for automation, required ownership and tagging, and established automation as the default. It combined central governance with distributed use: teams could manage their own certificates and workflows, but within common profiles, policies, and reporting.

The benefits are now visible at enterprise scale. Manual-touch renewals have fallen below one percent, unowned certificates have declined from more than 400 to about 50, and the inventory has expanded from an initial working estimate of roughly 7,000 certificates to over 18,000.

The central lesson is that in our second attempt to deploy CLM, we didn't solve a technology shortage. We solved an operating model problem. Trust Lifecycle Manager supplied the inventory, policy, and automation platform, but the benefits appeared only after we defined who owned the platform, how teams would participate, and what standards every certificate would follow.



87.5%
reduction in unowned certificates

61%
Increase in inventoried certificates

99%
Reduction in manual-touch cert management

Planning

Designing the operating model for enterprise certificate management

Customer Zero established a governance plan for ownership, discovery, policy, automation, and escalation before enterprise implementation began.

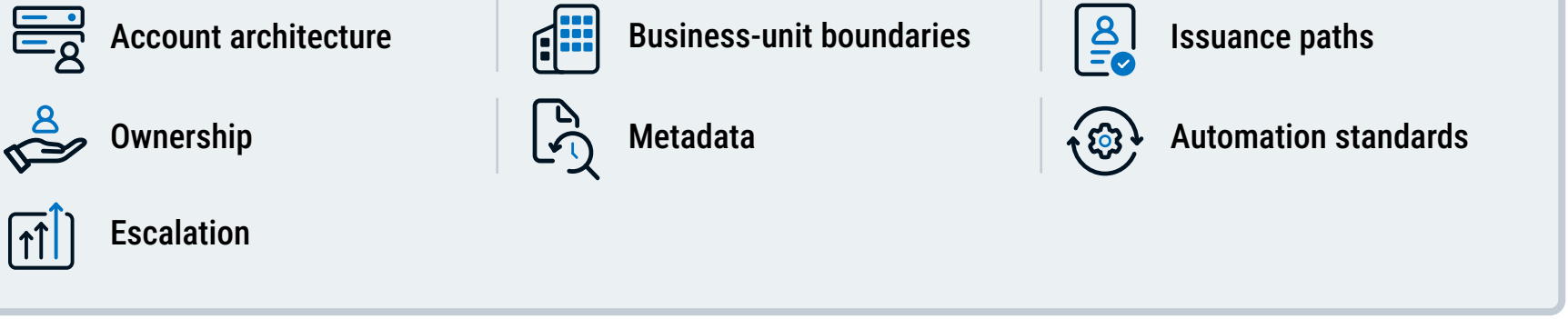
Project leaders chose to implement the full PKI with one DigiCert ONE account, centralized policy, and team-aligned business units. They set standards for the project including automation to the greatest extent practical.

The team made sure to involve executive sponsors, the Trust Office, operating teams, application owners, security stakeholders, and specialists in the project, which began with an SRE Operations pilot.

Before discovery, we mapped network coverage, defined metadata and ownership, aligned source systems, and prioritized standards-based automation. We conducted weekly meetings to discuss problems and progress.

Planning for a unified program

Customer Zero formally brought the work under a unified program roughly eighteen months after some teams had begun using Trust Lifecycle Manager independently. Christina Holterhoff, Principal PKI Trust Operations and DigiCert's global TLM administrator, recalls that planning for the new, centrally governed implementation was relatively short and focused on seven elements:
Account architecture



Who had to be involved

Customer Zero required both centralized authority and local expertise. Executive sponsorship provided the mandate to remove cross-functional roadblocks and pursue complete automation for public TLS certificates. A central PKI administrator in the Trust Office owned enterprise policy, approved certificate authorities, automation requirements, business-unit design, user administration, and reporting to executives.



Operating teams and application owners supplied the deployment knowledge that no central group could infer: where a certificate was installed, how a service restarted, which account or API controlled it, and what an outage would affect. The six principal certificate-managing teams included Service Desk, Engineering, Back Office, SRE, Networking, and Infrastructure Systems. Product management, engineering, support, and subject-matter experts helped resolve Trust Lifecycle Manager behavior and product gaps. Security and risk stakeholders supplied policy authority and quantified outage exposure.

We selected the SRE Operations group as the initial pilot group. The choice of that group was deliberate. SRE managed high-value customer-facing services, knew the deployment topology, and was motivated to reduce the substantial manual workload. The close collaboration between the pilot team and central administrator exposed connector and inventory problems while keeping the scope small.

We have the benefit of being our own PKI provider, so this process reminded us of a functional reality that all other types of organizations need to consider. Enterprise PKI implementation benefits from specialist expertise that most organizations don't carry internally. Planning for professional services support from a PKI vendor compresses timelines and resolves integration problems that would otherwise stall the project for weeks.

Three decisions required before discovery

Discovery is often described as the first implementation activity, but several decisions must precede it.

1. First, the team needs to map the network rather than merely list expected certificates. Sensors can only scan segments they can reach. Planning includes enumerating regions, VLANs, isolated zones, cloud accounts, firewall paths, and systems that require local agents or API-based connectors.

2. Second, the organization must define a minimum metadata and ownership model. We established a tag taxonomy covering attributes like issuing CA, service or application, environment, automation state, and whether a certificate could expire without renewal. Defining tags before discovery avoided trying to retrofit meaningful metadata across tens of thousands of records.

3. Third, source systems must be prepared for mapping. Nearly all teams had issued certificates from the main CertCentral division, so a CertCentral connector couldn't determine which Trust Lifecycle Manager business unit should receive a certificate. We planned matching CertCentral divisions and Trust Lifecycle Manager business units, then required teams to identify and move their existing certificates. New issuance profiles also had to point to the correct division, so the inventory would remain organized after the initial cleanup.

The standards set for Customer Zero

The target design used a single enterprise DigiCert ONE account with operating teams represented as business units. Central policy would be uniform, while business-unit leaders could approve profiles and manage their own inventory. Every certificate needed an accountable operating owner, and every automation also needed an owner. New and misassigned certificates would enter a defined triage process rather than remain indefinitely in a catch-all queue.

Automation was the default, with standards-based enrollment preferred wherever the endpoint supported it. ACME was the first choice for TLS, because it can automate both issuance and domain control validation. EST and SCEP covered device and network enrollment, with SCEP accepted where broad device support outweighed the disadvantages of its older shared-secret design. APIs, connectors, agents, sensors, and controlled scripts would address systems without native protocol support.

The governance rules were equally important. Public TLS certificates would move to 47-day validity. Renewal could not depend on recurring human approval. Automation identities would use service accounts or privileged-access management rather than personal credentials. Lifecycle actions would be logged and tied to the deployment target. Exceptions would be time-limited, technically justified, approved by the PKI administrator, and reported to the executive sponsor. A failed first automation attempt was not, by itself, grounds for an exception.

7 Rules for DigiCert's PKI Modernization

1. All roles, such as administrator, must be service accounts (Role-Based Access Control). There must be 2 or more users for each role.
2. One PKI administrator oversees policy enforcement and policy communication.
3. Each team has their own business unit in Trust Lifecycle Manager.
4. Each team has a minimum of two certificate approvers (if they use public trust) to avoid blocking cert issuance while waiting on approval workflows.
5. All unassigned certs must be reviewed within 7 days of detection and assigned to business units.
6. Teams can move certificates from their business unit to the unassigned business unit, because discovery sometimes puts things in the wrong place.
7. Automation is required for all certs. Exceptions are granted on a limited time basis for where automation isn't possible or where the cert will not require renewal. These are reviewed at least every 6 months. The order of preference is ACME > SCEP > EST > API.

Cadence, phases, and escalation

During deployment, Customer Zero teams met weekly to review progress, results, blockers, and next actions. Slack handled communication between meetings. Decisions and product or process gaps were recorded in Jira, both to preserve an audit trail and to prevent the same question from being reopened by another team. The project also learned to define escalation routes in advance. An unclear boundary between support and product delayed one issue by several weeks.

The implementation sequence was: planning, discovery, automation, and then scale. Within that sequence, the program worked in manageable increments rather than attempting a single enterprise cutover. The pilot proved the structure, discovery established the real inventory, and automation began with high-risk services while favoring the easiest reliable mechanism within each risk tier. Difficult appliances and third-party dependencies moved to a controlled backlog rather than blocking the majority.

This cadence preserved normal operational objectives and produced visible results for sponsors. It also allowed the standards to mature without becoming optional. The plan was not a static document created before technical work. Instead, it was a governance framework that turned discoveries, implementation issues, and product feedback into controlled program improvements.



Discovery

Building an accurate, owned, and continuously updated certificate inventory

Our pre-discovery estimate of total certificates was 7,000. In fact, there were 18,000, and this kind of surprise is typical.

Discovery using Trust Lifecycle Manager employed sensors, agents, CA and cloud connectors, and APIs to find certificates across network endpoints, local stores, issuance records, and cloud platforms.

The process revealed that DigiCert's inventory exceeded 18,000 certificates, not the estimated 7,000. There were many certificates with errors, and more than 400 had no clear owner.

The complete discovery process required people to assign ownership, correct business-unit mappings, validate network coverage, triage ambiguous records, and set up continuous discovery for ongoing changes.

What discovery does with Trust Lifecycle Manager discovery

Trust Lifecycle Manager combines several discovery methods, because no single scan can reveal an enterprise's entire certificate estate. Network scans use a DigiCert sensor on a dedicated host to probe defined address ranges, ports, appliances, and services. System scans use agents installed on Windows or Linux servers to inspect local certificate stores, files, and supported applications. CA connectors import issuance records from platforms such as CertCentral or an internal CA. Cloud connectors and APIs enumerate certificates and deployment endpoints in services such as load balancers, content delivery networks, and certificate managers.

The methods overlap by design. A CA connector can show that a certificate was issued even if it hasn't been deployed. A network sensor can find a certificate from an unexpected CA on a reachable service. An agent can identify a client-authentication, S/MIME, or file-based certificate that never appears on a listening port. Trust Lifecycle Manager correlates these observations into the inventory and preserves deployment endpoints, so one certificate used on several systems is not mistaken for one automation task.

What we could identify immediately

The fastest starting point was CertCentral. Its connector could import the account's issuance history and map CertCentral divisions to Trust Lifecycle Manager business units. That provided broad visibility into known public certificates. It did not, however, produce an owned enterprise inventory on its own. Nearly all teams had issued from the same main division, so the connector couldn't determine which operating group controlled each certificate. Unmapped records entered an unassigned business unit for triage.

Team spreadsheets and existing records contained useful information, but only about half of the eventual inventory was being tracked cohesively. Network scans then found certificates that didn't come from the expected CertCentral path, including Let's Encrypt certificates and other out-of-band deployments.



Why discovery required manual investigation

Automation could collect technical facts, but people had to supply operational meaning. We asked each team to identify their certificates in the common CertCentral division. Trust Operations administrator, Christina Holterhoff, created corresponding CertCentral divisions and Trust Lifecycle Manager business units, then helped move certificates into the correct containers. Teams reviewed issuance profiles so future certificates would enter the correct division instead of returning to the catch-all queue.

Ownership research was the most important manual task. A serial number, subject, issuer, SAN list, key properties, validity dates, and an endpoint are valuable, but they don't identify the person who can safely replace a certificate. Teams had to trace services, consult application and network owners, distinguish obsolete systems from live dependencies, and decide which business unit would accept responsibility. Misassigned and unassigned certificates were placed into a standing triage process.

Discovery also required coverage engineering. Sensors cannot report on a network they can't reach, and an empty result can mean either "no certificates" or "no route." We missed an isolated network segment because no sensor had a permitted path to it. The gap was found by accident roughly six months into the project. The network wasn't large enough to force a change in the overall design, but the episode established a clear rule: reconcile sensor placement against the network topology, not against an assumed certificate list.

Limits of a scan

A scan must be told what to inspect. Discovery isn't a universal button that understands every virtual host, address, or proprietary service. A web server may present a default certificate while several other sites remain behind different names or addresses. Nonstandard ports, firewalls, and segmented networks require explicit scope. Certificates that never terminate a network connection require an agent or source-system connector.

These limitations explain why discovery is a process rather than an event. Coverage must be tested, exceptions investigated, and multiple evidence sources reconciled. The objective is not merely to collect certificates. It is to prove that every relevant network, issuance source, cloud control plane, and local store has an intentional discovery method.

The inventory Customer Zero created

The inventory records technical identity and lifecycle data, including serial number, common name, SANs, issuing CA, status, and validity. It also records the organizational and operational context needed to act: business unit, owner, tags, certificate profile, automation status, and deployment endpoints. A certificate deployed in several locations may appear through multiple endpoint records because renewal and installation must succeed at each destination.

Keeping the inventory current

Customer Zero left the discovery infrastructure in place. Sensors continue network scans in the relevant regions and segments. Agents continue local system discovery. CA and cloud connectors periodically import new issuance and asset data. This matters because new projects, vendors, cloud resources, and employee-driven issuance continually create certificates outside the original inventory.

New discoveries are assigned automatically where metadata and rules are sufficient. Ambiguous records enter the unassigned queue for review, normally within a defined service level. The Trust Officer asks who issued or deployed the certificate, which business unit should own it, and whether it complied with policy. The issuing path is corrected when a certificate bypasses Trust Lifecycle Manager.

Automation

Using standards first, integrations second, and controlled scripting for the edge cases

Customer Zero brought automated enrollment and renewal close to 100 percent of eligible endpoints, with manual-touch renewals below one percent of the inventory.

Customer Zero automated nearly all eligible certificate enrollment and renewal: 50 percent through ACME, 40 percent through SCEP or EST, and most remaining endpoints through APIs.

We prioritized standards, then used connectors, sensors, agents, AWR workflows, and controlled scripts to integrate platforms lacking native enrollment protocol support. Many of these integrations are pre-built for customer use.

Some difficult applications required custom engineering, while a small number of technically or contractually blocked systems became owned, monitored exceptions.

Standards first

The final operating performance figures are more meaningful than a claim of absolute coverage. Customer Zero distinguished between systems that could be automated and systems whose technical or contractual design prevented it. Exceptions were identified, assigned, and monitored instead of disappearing into spreadsheets. Automation became the central operating objective once we had a complete and detailed inventory.

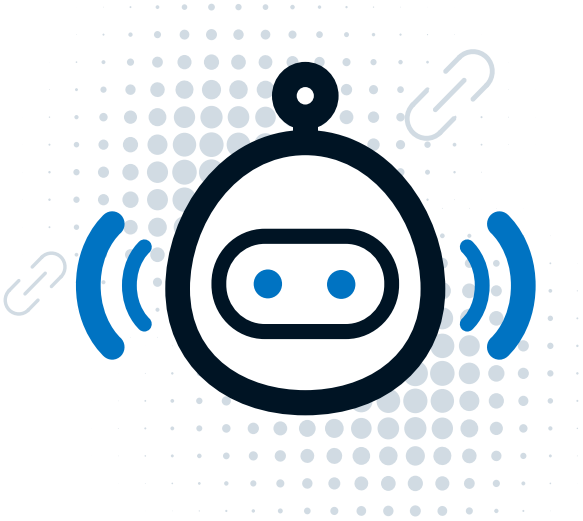
A standards-first policy

We preferred standard enrollment protocols before proprietary integration. ACME was the first choice for TLS-capable systems because it can automate account authorization, issuance, renewal, and domain control validation. That end-to-end behavior becomes especially important as certificate validity and validation-reuse periods shrink.

EST and SCEP extended enrollment to devices and managed endpoints. EST provides modern enrollment over HTTPS. SCEP remains widely supported across network equipment, mobile-device management, and Microsoft-oriented environments, so Customer Zero used it where endpoint support made it the practical option. The corporate laptop program is a clear example. Intune and SCEP replaced a manual self-service process for device certificates, sharply reducing loss-of-access tickets for new hires and expiring credentials.

The policy was not “use ACME at any cost.” It was “use a supported standard before creating custom logic.” Sensors can proxy ACME, EST, and SCEP traffic into Trust Lifecycle Manager, allowing internal clients to use a controlled outbound path. Standard profiles also constrain issuing CA, validity, key type, domains, and ownership metadata.

Connectors, sensors, agents, and Admin Web Requests



Connectors integrate Trust Lifecycle Manager with external certificate authorities, cloud services, key stores, and network platforms. They can import existing certificates for discovery and, where supported, use the platform API to deploy renewed certificates without installing software on the target. Sensors provide the network gateway used by many connectors and can reach appliances or cloud control planes over API, HTTPS, or SSH. Agents run on individual servers, where they can generate keys, discover local certificates, deliver certificates, and invoke controlled pre- or post-delivery scripts.

Admin Web Request (AWR) is the administrator-initiated enrollment and delivery framework in Trust Lifecycle Manager. For custom applications, an administrator selects an AWR-enabled certificate profile, delivery agent and format, renewal schedule, ownership metadata, and an optional post-delivery script. The agent receives the issued certificate and runs the verified script to install it locally or connect to a remote target by API or SSH. This provides a governed bridge between Trust Lifecycle Manager and an application that lacks ACME, EST, SCEP, or a packaged connector.

AWR usage was not tracked as a separate corpus-wide metric, so a precise certificate count is not available. Its role in the automation toolkit is nevertheless clear. It turns a custom deployment script into a centrally scheduled, logged, repeatable lifecycle rather than an operator’s one-off command.

We have created integrations for many devices and services at the heart of critical infrastructure, such as PAM systems and network boundary devices.

What was comparatively easy

Native protocol support made the cleanest deployments. Device enrollment through Intune and SCEP worked well after the network accepted the new trust path. Windows autoenrollment was relatively straightforward once the component, Group Policy, firewall rules, and DCOM permissions were understood. Network port scanning also worked largely as documented, giving infrastructure teams immediate visibility into open and unsecured services.

Existing integrations reduced custom work for common platforms. Several of these were blockers to our automation requirement because they lacked automation support. We worked closely with these vendors to introduce easy automation principles, promoting ACME as the preferred solution. Supported cloud and network connectors allowed Trust Lifecycle Manager to manage certificate objects through the vendor control plane. Once a working pattern was established, it could be repeated across similar endpoints and business units.

What required engineering

Infrastructure and network appliances created most of the difficult cases. Many didn't support ACME or another suitable enrollment protocol. Some exposed an API but required target-specific code. Others required SSH, maintenance windows, service restarts, or commands to bind the new certificate after delivery. Windows autoenrollment installed certificates but didn't automatically bind every use case. RDP could require a PowerShell step, while SQL Server could require manual configuration and a restart.

Customer Zero engineers reused what they could. One engineer built a small library of Python functions for Trust Lifecycle Manager API work and placed it in GitHub for reuse across projects. PowerShell automated DCOM and configuration changes. Our integrations page and product-solutions repository also provide integration and AWR script templates developed for repeatable customer and internal use. Customer Zero did not maintain a separate corpus-wide count of scripts that originated as pre-existing customer solutions.

No corpus-wide certificate count was maintained for custom scripting. Some teams operated in environments where almost the whole network-infrastructure essentially required custom scripts, but that observation can't be applied to the full 18,000 certificate inventory. API-based doesn't necessarily mean a unique custom script per certificate. One integration can manage many endpoints, and many will be pre-built and supplied by us.



What couldn’t be automated

A small set of systems simply couldn’t be fully automated. Some locked appliances could generate only self-signed certificates or offered no supported import path. Certain complex demonstration environments didn’t fit the Trust Lifecycle Manager automation model. Certificates hosted by third parties such as Adobe, Moodle, and Oracle depended on provider-controlled installation processes. These were operational dependencies, not failures that Trust Lifecycle Manager could solve unilaterally.

Customer Zero responded by documenting the exception, assigning an owner, setting an expiration or review date, and working with vendors to add ACME or usable APIs. SRE initially monitored Trust Lifecycle Manager renewals with independent tooling and worked directly with product and engineering when jobs behaved inconsistently. Some patterns that looked like cases specific to our environment were recognized as common customer configurations and incorporated into the product.

The outcome wasn’t automation without human involvement. It was human attention moved upstream, a unified effort to design profiles, validate deployment, monitor failed jobs, and improve exceptions. Routine renewal became background activity, while operators focused on the small number of events that required human judgment.

Scaling

Applying the Customer Zero model across our worldwide infrastructure

Scaling succeeded through repeated patterns and phased onboarding—not through a single global cutover—and it remained an ongoing operational program.

We scaled Customer Zero through phased onboarding, using the structure from the SRE pilot, profiles, connector mappings, procedures, and scripts where appropriate.

Discovery for the different business units and regions required attention to the network specifics of each. Work done in planning to map these greatly eased the process.

Focusing on the highest-risk cases first and moving technically blocked cases to a backlog kept the process moving forward.

From pilot to enterprise pattern

Customer Zero used SRE Operations as its pilot. The team managed customer-facing production systems, including layered public TLS deployments where the same service name could appear on a web application firewall, load balancers, and multiple application servers. Solving those cases established the business-unit structure, connector mappings, certificate profiles, and escalation practices needed for wider adoption.

The scale phase applied those patterns across the six principal certificate-managing teams and their environments. Each operating group received a Trust Lifecycle Manager business unit with access to its own inventory, while the Trust Office retained enterprise policy and administrative control. CertCentral divisions were aligned to those business units, so newly issued public certificates arrived in the correct organizational container.

Worldwide discovery and infrastructure coverage

Infrastructure teams deployed scanning in multiple environments, including U.S. and European regions, and extended the model to internal servers, production services, corporate laptops, network appliances, cloud platforms, and third party-hosted services. Scaling discovery required more than copying a scan definition. Each region and isolated network needed a reachable sensor or local agent, correct firewall rules, and an explicit source-system connector. Identifying all of these at the planning stage greatly facilitated the scaling project.

In addition to the instance of the missing sensor and invisible segment, scaling also exposed differences in how teams consumed certificates. Corporate operations could use Intune and SCEP for laptop certificates. Windows infrastructure could use autoenrollment and Group Policy. Production SRE depended heavily on public TLS and layered deployments. Network infrastructure faced KVMs, power equipment, firewalls, and other appliances with limited standard-protocol support. A common policy therefore required several implementation patterns.

What scaled smoothly

The repeatable components improved with each team. Network scanning was described as easy to configure and useful immediately. Standard enrollment protocols and supported connectors reduced endpoint-specific work. The business-unit model made ownership and permissions understandable: teams managed their own “section of the world” without taking administrative control of another team’s certificates.



The pilot produced reusable implementation patterns. Teams could adapt established profiles, connector mappings, deployment procedures, API functions, and scripts instead of designing each integration independently. This reduced the effort required for each subsequent business unit while preserving a common policy model.

The operating cadence scaled as well. Weekly team meetings surfaced progress and blockers. Slack handled daily coordination. Jira preserved decisions and product gaps. Visible metrics—certificates automated, owners assigned, outages avoided, and time reclaimed—gave executives evidence that the phased approach was working. When one team reached full automation, the result motivated another team to accelerate its own plan.

Where scaling remained difficult



The rollout was not uniformly smooth. Ownership ambiguity, network reachability, vendor-specific APIs, certificate-binding requirements, and service restarts all complicated deployment. Product inconsistencies required engineering support and, for high-value systems, temporary independent monitoring to confirm that renewal jobs ran reliably.

The long tail did not become easier merely because the program was global. Every new hardware vendor could introduce a different API, import method, binding command, or restart requirement. Some appliances offered no usable automation. Third party providers controlled installation for certificates used on their services. These dependencies required vendor engagement, custom code, or documented exceptions. The project avoided allowing that long tail to stop the rollout. It automated the highest-risk services first, selected the most reliable available mechanism within each tier, and moved technically blocked systems to a known backlog. This produced far more risk reduction than waiting for a universal solution before scaling anything.

Operational outcomes at scale

By late 2025, the managed corpus of certificates had grown by over 150%. Previously unknown certificates had been detected and automated, and unowned certificates were almost eliminated.

The benefits appeared in both reliability and workload. We recorded no customer-impacting certificate-expiration outage in the second half of 2025, while improved visibility caught several renewal failures as near misses. The project's model estimated about 1,500 hours of recurring manual certificate work eliminated annually, worth roughly $147,000 at the blended labor rate used in the analysis. Estimated annual value including outage-risk reduction ranged from about $630,000 to $1.2 million.

Corporate operations saw a practical version of the same benefit: automated device certificates greatly reduced service desk tickets caused by employees losing network access. The program also created a foundation for 47-day public TLS certificates and for future cryptographic migration, because we can now locate, classify, and rotate most of our certificate estate through managed workflows.

Scaling is an operating state, not an end date

As Yannick Gelmo, an Information Systems Engineer on the Infrastructure team explained, Customer Zero remains ongoing. Infrastructure changes, new cloud services, acquisitions, organizational changes, and vendor replacements continually introduce certificates and integration requirements. Continuous discovery must find new assets, triage must assign them, and policy must prevent new issuance paths from recreating fragmentation.

The lasting global capability is therefore not a frozen 18,000-certificate inventory. It is a system for absorbing change: one policy authority, delegated operating teams, permanent discovery, standards-first automation, reusable integrations, monitored exceptions, and an escalation path into our product and engineering teams. Scaling did not remove every complication. It made the complications visible, owned, measurable, and repeatable enough to manage worldwide. most of our certificate estate through managed workflows.