SWIFT Infrastructure

Reliability, Observability and Monitoring of Critical Systems at KCSD

Practical experience building a SWIFT environment: architecture, fault tolerance and observability

Agenda

01

A2 Architecture

2 data centers, Alliance Access + Web Platform, 3 Service Bureau gateways

02

Reliability: Mirror Disk & Backup

DB Recovery, saa_export, message and event archives

03

SWIFT CSP / CSCF

Mandatory security controls, LAU as a mandatory requirement

04

LAU for MT and MX

S-block MDG, XML signature for pacs.008, keys and rotation

05

ISO 20022: CSD Migration (CBPR+ 2025)

pacs.008/009, camt.053/054, DataPDU envelope

06

Observability: SNMP → Grafana/Loki

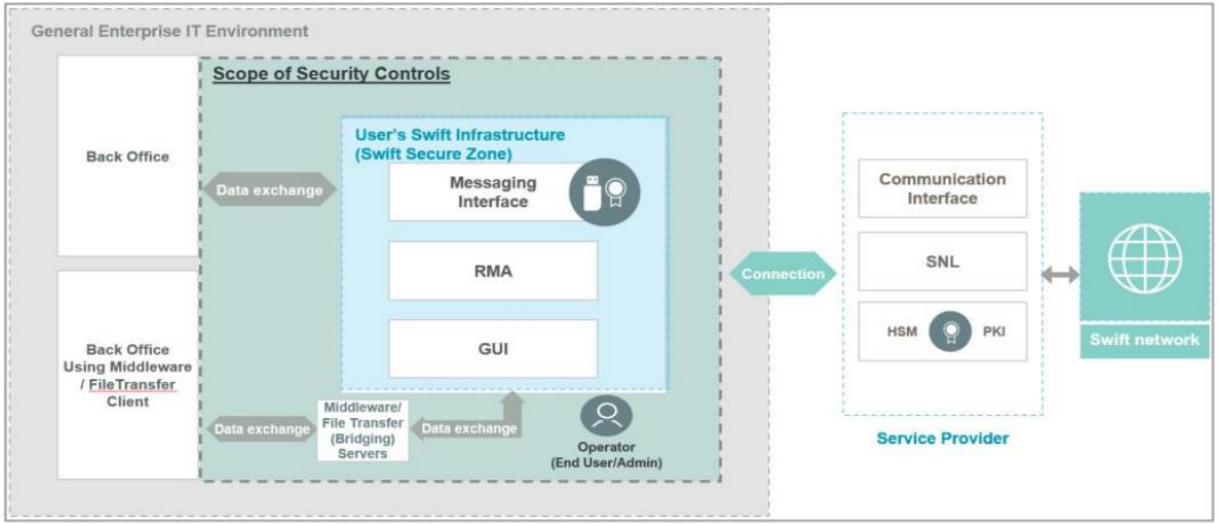
OID mapping, real-world incidents from logs, dashboard ideas

Our Architecture: A2 Connection Type

Architecture A2 – Users owning the messaging interface, but not the communication interface

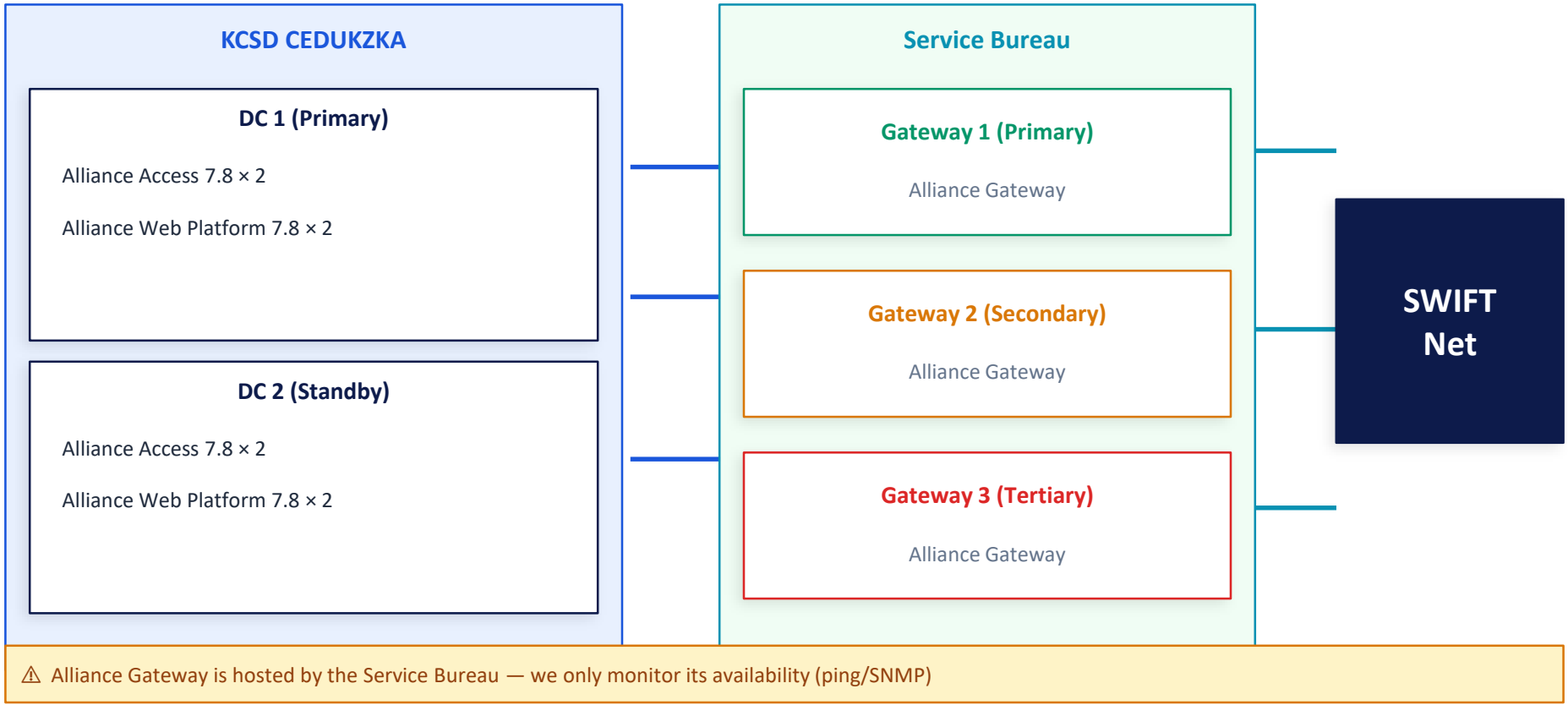
The messaging interface is owned, but a Swift connectivity provider (for example, a service bureau), a Group Hub or Swift[6] owns the licence for the communication interface.

Drawing 4 displays the case where the messaging interface is owned by the user and resides within the user environment.



Drawing 4: Architecture A2 – Messaging Interface only within the user environment

Our Architecture: A2 Connection Type



Reliability: Mirror Disk and Configuration Backups

Mirror Disk (Database Recovery)

- License required: 14:DATABASE RECOVERY
- Mirrors Alliance Access DB to a separate disk in real time
- Compress Recovery Backups — reduces disk space usage
- Supports restoring Backup to other Alliance Access instances
- Maximum Read Rate — limits I/O load during backup

```
Primary DB  →  Mirror Disk  →  Backup Disk
```

Configuration Backups

Database Backup

Full DB copy. Manual or scheduled. Manual/Automatic mode.

Recovery Backup

Recovery points on a separate disk. Restore to the last known good state.

Config Replication (saa_export)

Export configuration: routes, partners, templates, DNs. saa_export/saa_import.

Message + Event Archives

Archives of messages and events. Configurable schedule. Included in Recovery Backup.

SWIFT CSP / CSCF: Security Requirements

SWIFT Customer Security Programme (CSP) — annual attestation against CSCF (Customer Security Controls Framework). LAU is a mandatory security control.

1.1 — Mandatory

Internet Access Restriction

Isolate SWIFT infrastructure from the public internet

1.2 — Mandatory

Privileged Account Control

Role separation: Operator, Administrator

2.2 — Mandatory

Security Updates

Regular updates of Alliance Access and OS

2.9 — Mandatory

Transaction Control (LAU)

LAU protects the file exchange between Backoffice and Alliance Access

6.1 — Mandatory

Malware Protection

Antivirus protection on all SWIFT components

7.2 — Advisory

Incident Management

SNMP monitoring, alerts, Grafana/Loki — our approach

⚠ Non-compliance with CSP/CSCF may result in restricted access to the SWIFT network. Annual attestation is mandatory.

LAU for MT Messages: S-block with MDG (HMAC-SHA256)

Without a valid signature, Alliance Access rejects the message. Protects against spoofed files during automated processing.

MT 540 + S-block (example)

```
{1:F01CEDUKZK0AXXX0000000000}
{2:I540ZYASBEB0XECLN}
{4:
:16R:GENL
:20C::SEME//10947116124422
:35B:ISIN US48581R2058
:36B::SETT//UNIT/100,
:95P::PSET//MGTCBEBEECL
:95R::DEAG/ECLR/24943
-}{S:{MDG:CBB0AC1CB18AF27E626F65E52A6B7BEFC497C60D...}}
```

Computing the MDG Value

1. Strip any S:-block. Keep only blocks {1:}...{5:}
2. Concatenate Left Key + Right Key → 32 ASCII chars (256 bits)
3. Compute HMAC-SHA256(message_bytes, merged_key)
4. Convert result to uppercase hex → 64 characters
5. Append to message: {S:{MDG:...}}

LAU for MX: XML Signature in pacs.008 (ISO 20022)

Every MX message must contain an <Saa:LAU> block with a W3C XML DSig signature before </Saa:DataPDU>. Computed by mx_compute.py for each message.

pacs.008 + <Saa:LAU> (example)

```
<Saa:DataPDU xmlns:Saa=...>
  <Saa:Revision>2.0.13</Saa:Revision>
  <Saa:Header>
    <Saa:Sender>
      ou=xxx,o=cedukzka,o=swift
    <Saa:MessageIdentifier>
      pacs.008.001.08
  </Saa:Header>
  <Saa:Body>
    <pacs:Document>...</pacs:Document>
  </Saa:Body>
  <Saa:LAU>
    <ds:DigestValue>
      CiqvbomM5/8E...gTc=
    <ds:SignatureValue>
      JPTZ/s5yhINW...KiK=
  </Saa:LAU>
</Saa:DataPDU>
```

LAU Computation Algorithm for MX

1. **Add empty <Saa:LAU></Saa:LAU>**
   Insert before </Saa:DataPDU>
2. **Compute DigestValue**
   Exclusive C14N(DataPDU) → SHA-256 → Base64
3. **Build <ds:SignedInfo>**
   Insert DigestValue from step 2
4. **Compute SignatureValue**
   C14N(SignedInfo) → HMAC-SHA256(Left+Right Key) → Base64
5. **Insert final <Saa:LAU>**
   DigestValue + SignatureValue from steps 2 and 4

LAU: Key Management and Rotation

LAU Key Structure and Parameters

Key length	256 bits = 32 ASCII characters (hex)
Left Part Key	First 16 characters (0-9, A-F)
Right Part Key	Last 16 characters (0-9, A-F)
Storage	Split knowledge: Left and Right entered separately
Algorithm	HMAC-SHA256 (RFC 2104)

```
Left Key:   0123456789ABCDEF
Right Key:  0123456789ABCDEF
Merged:     0123456789ABCDEF0123456789ABCDEF
```

Key Rotation and Alliance Access Configuration

⚠ **Key rotation is mandatory every 90 days**

- Message Partners → select partner → Authentication tab
- Enable Local Authentication → select HMAC-SHA256
- MT: Left Key + Right Key (To / From / To & From)
- MX: Receive Key First Part + Receive Key Second Part
- Update keys in DRSCB/MFET simultaneously with Alliance Access
- LAU Key Automatic Renewal — 90 days before expiry

ISO 20022: KCSD Migration (CBPR+ 2025)

Per CBPR+ 2025: last year KCSD migrated to pacs.008 and pacs.009. Ready to receive camt.053 and camt.054. Our team is ready to implement any ISO 20022 message formats.

DataPDU Envelope (XMLv2 / Saa 2.0)

```
<Saa:DataPDU xmlns:Saa=...>
  <Saa:Revision>2.0.13</>
  <Saa:Header>
    <Saa:Sender>
      ou=xxx,o=cedukzka,o=swift
    <Saa:MessageIdentifier>
      pacs.008.001.08
  </Saa:Header>
  <Saa:Body>
    <pacs:Document>
      <!-- ISO 20022 payload -->
    </pacs:Document>
  </Saa:Body>
  <Saa:LAU>...</Saa:LAU>
</Saa:DataPDU>
```

ISO 20022 Formats: Status & Roadmap

✓ 2025	**pacs.008 / pacs.009**	CBPR+ 2025: implemented. Inbound and outbound transfers.
⟳ Ready	**camt.053 / camt.054**	Ready to receive: account statements and credit notifications.
→ Exp.	**Any ISO 20022**	Our team implements any CBPR+ formats.
~ 2026+	**Next formats**	Per CBPR+ roadmap or agreement with counterparties.

Observability: SNMP Traps → JSON → Grafana

OID → JSON Field Mapping

OID (abbreviated)	JSON field	Example
...18494.2.1.1 / 5.1.1	system	cedu / SWP01
...18494.2.1.4 / 5.1.4	component	MXS, BSA, GEN...
...18494.2.1.5 / 5.1.5	sev_code	10117, 3007, 10004
...18494.2.1.6 / 5.1.6	level	Warning, Severe, Info
...18494.2.1.7 / 5.1.7	category	Communication, Security
...18494.2.1.8 / 5.1.8	message	Validation error, login.failure
...18494.2.1.9 / 5.1.9	details	HEX → UTF-8 (auto-decoded)
...18494.5.1.11/.12	client_ip	10.10.136.140, 10.10.137.72

2 SNMP Trap Sources

Alliance Access

OID: 1.3.6.1.4.1.18494.2.X

MXS, BSA, GEN, SSS, SNIS...

Alliance Web Platform

OID: 1.3.6.1.4.1.18494.5.X

Security, Software, Configuration

⚠ OID .9 (details) is transmitted as hex (4d_65_73...) — swift-snmp-trapper decodes to UTF-8 automatically

Real-World Incidents from Logs (30/04/2026)

Operator Account Lockout (Operator disabled)

```
11:33:55 — BSA — Operator USER15 : disabled after too many invalid signons
```

7 login.failure attempts → automatic lockout. Every attempt captured via SNMP. Unlocked at 16:51 (BSA — Operator enable).

Failed Login — Expired Password

```
18:11:14 — AWP — USER8: The password of user USER8 has expired
```

Alliance Web Platform notifies via SNMP trap. Source IP recorded: 10.10.10.10.

MX Message Validation Errors

```
18:25 — MXS — Text validation error | 22:11 — Format error (offset 29) | 22:10 — Data length error × 5
```

Validation error (code 10117) — field failed validation. Format/Length errors — structural errors on the sender's side. UUMID identifies the exact message.

Alliance Access Unreachable + SAG Connection Lost

```
01:30 — SEVERE — Alliance Access instance unreachable (Connection refused)
10:57 — Severe — SAG connection lost + cedukzka_rmadistribution Queue Closed
```

SEVERE from AWP: lost connection to AA instance. Recovered manually via Administration (15:20/15:29 — modifyInstance). SAG loss affected multiple PIDs simultaneously.

What You Can Build with SNMP/System Log Data

Every SNMP trap contains: timestamp, component, severity, category, details and user IP address. This enables a full audit trail and operational monitoring dashboard in Grafana/Loki.

User Access Audit

- Who logged in / out of the terminal and when
- Source IP address for every session

Message Processing Monitoring

- Validation / Format errors by Message Partner
- UUMID of rejected messages for investigation

Critical Failures & Infrastructure

- Alliance Access unreachable — immediate alert
- SAG connection lost — gateway connectivity loss

ABS ↔ SWIFT File Exchange

- File Transfer Errors – ABS - SWIFT
- NackPrt — rejection receipts requiring attention

Security & CSP Compliance

- User activity map by IP subnet
- Suspicious logins outside business hours

SLA & Operational Metrics

- ABS availability and uptime metrics
- Alliance Access availability over period (uptime)

Key Takeaways

1. 2 DCs + Mirror Disk + saa_export — recovery without data or config loss; 3 Service Bureau gateways ensure continuity
2. LAU (CSP 2.9) for MT {S:{MDG:…}} and MX <Saa:LAU> — perimeter protection for automated processing without operators
3. CBPR+ 2025: pacs.008/009 live; camt.053/054 ready; team can implement any ISO 20022 format
4. SNMP OID .2.X (AA) + OID .5.X (AWP) → swift-snmp-trapper → JSON → Grafana/Loki — full observability
5. From logs: access audit, message monitoring, critical failures, file exchange SLA — all in one dashboard