SKYE Identity Standard (SIS) v1.0

SSO + SCIM Production Requirements

Issued by **Skyes Over London LC** (Solenterprises AI Division)

Effective date: **2026-02-27**

Purpose: Define the mandatory identity controls that every SKYE deployment must implement and pass before any tenant goes live. This standard prevents identity drift, reduces support burden, and aligns SKYE products with enterprise procurement expectations.

Scope: All SKYE-branded products, client portals, internal tools, and marketplace apps that serve organizations or teams.

Profiles: SIS-Core (default) and SIS-Enterprise (regulated / large orgs).

Contents

1. Core Principles (Non-Negotiables)

SSO and SCIM are separate controls with one combined mission: the customer owns identity, and SKYE enforces it correctly.

- When SSO is enabled, SKYE does not store customer passwords.
- Identity is tenant-bound. Every auth decision is evaluated inside an org/tenant boundary.
- Deprovisioning must be fast. Removal in the customer IdP must quickly eliminate access in SKYE.
- No silent failures. Every auth and provisioning event is logged and visible to tenant admins.
- Secure by default. Correct signature validation, correct issuer/audience checks, and correct rotation behavior. No shortcuts.

SSO answers: **Can this person authenticate right now?**
SCIM answers: **Should this person exist in the app at all, and what access do they get?**

2. Terminology (SKYE Vocabulary)

- **Org / Tenant**: Customer environment with its own policies, IdP config, SCIM token(s), and role mappings.
- **IdP**: Identity Provider (Okta, Microsoft Entra ID, Google Workspace, Ping, etc.).
- **SSO Protocols**: OIDC (preferred) and SAML 2.0 (required for enterprise compatibility).
- **SCIM**: SCIM 2.0 provisioning and group sync.

3. SKYE SSO Standard (Production Requirements)

3.1 Supported protocols

- MUST support OIDC (Authorization Code + PKCE).
- MUST support SAML 2.0 for enterprise compatibility.
- MAY support both simultaneously per tenant, but only one can be enforced as the primary sign-in method.

3.2 Tenant configuration and routing

- MUST support per-tenant IdP configuration (OIDC issuer/client ID/redirect URIs/JWKS discovery; SAML entity ID/ACS URL/certificate(s)/SSO URL).
- MUST support routing by at least one: domain hint, org slug, or email discovery (enter email then route).
- MUST ensure redirect/callback URLs match exactly (no wild assumptions).

3.3 Token and assertion validation (no exceptions)

OIDC:

- Validate signature against IdP JWKS.
- Validate issuer (iss) and audience (aud).
- Enforce exp and nbf if present.
- Enforce state and nonce (anti-CSRF and replay).
- Reject tokens missing required claims.

SAML 2.0:

- Validate XML signature using pinned certificate(s).
- Validate audience restriction, destination, and recipient.
- Enforce assertion time window.
- Prevent replay (assertion ID cache).

3.4 Account linking rules

- MUST link SSO identities via stable identifiers: OIDC sub + issuer; SAML NameID or stable attribute + issuer.
- MUST handle email changes without breaking identity (email is not a stable identity key).
- MUST block account takeover by email collision across IdPs.

3.5 MFA and enforcement

- MUST support tenant-level enforcement: SSO required; disable password login.
- MUST allow MFA enforcement via IdP policy; record MFA enforcement when signals exist.
- SHOULD support step-up auth for sensitive actions (billing changes, role escalation, key export).

3.6 Session security

- MUST support session revocation on deprovision/disable, role downgrade, and (recommended) org policy updates.
- MUST use short-lived access tokens with refresh or rotating server sessions.
- MUST harden cookies (HttpOnly, Secure, SameSite configured).

4. SKYE SCIM Standard (Production Requirements)

4.1 SCIM compatibility

- MUST implement SCIM 2.0 endpoints at a tenant-scoped base path: /scim/v2/Users and /scim/v2/Groups.
- MUST support create, update (PATCH), deactivate/disable, and delete (if tenant chooses hard delete).
- MUST support filtering and pagination per SCIM expectations.

4.2 Provisioning lifecycle rules

- MUST support active=false or equivalent disable semantics.
- MUST deprovision within defined SLA: target under 2 minutes (Enterprise), maximum under 15 minutes (Core).
- MUST ensure deprovision triggers session revocation.

4.3 Group sync to roles and permissions

- MUST support group-to-role mapping per tenant.
- MUST map by group ID (not name) to survive renames.
- MUST define deterministic conflict rules; SKYE default is least privilege unless tenant overrides.

4.4 Idempotency and retries

- MUST be idempotent; repeated requests cannot create duplicates.
- MUST tolerate out-of-order events and retries.
- MUST surface actionable error messages for admins (not silent failure).

5. SKYE Admin Experience (Identity Console)

Every tenant that enables SSO or SCIM must have an Identity Console section for supportability and operational clarity.

- SSO status: enabled/disabled, protocol, issuer/entity IDs, last successful login time.
- SCIM status: enabled/disabled, last sync time, last error, token rotation controls.
- Group mappings: IdP groups mapped to SKYE roles.
- Test tools: Test SSO connection; Validate JWKS/certificate freshness; Send SCIM test user.
- Security posture summary: SSO required; password disabled; MFA enforced (when known).

6. Break-Glass Access (Emergency Admin)

Break-glass is required for enterprise tenants to prevent lockout while maintaining strict controls.

- MUST support a tenant-controlled break-glass method for lockout recovery.
- MUST require separate method from normal SSO (or an emergency IdP group), MFA, tight auditing, and optional IP allowlist.
- MUST alert and log every break-glass event.

7. Audit Logging (WORM-Mindset)

Identity changes must be traceable. SKYE deployments must generate immutable audit entries for auth, provisioning, and admin changes.

Minimum events:

- Auth: login success/fail, logout, token refresh, session revoked, tenant and IdP identifiers, IP, user agent, timestamp.
- Provisioning: SCIM user created/updated/disabled/deleted, group membership changes, role mapping results (before/after).
- Admin: SSO config changes, SCIM token created/rotated/revoked, group mappings, policy toggles.

Retention: default 1 year; enterprise option 3-7 years.

8. Security Baselines (SKYE Production Gate)

A deployment cannot be labeled SKYE Production Certified until it passes the following gates.

- Correct token/assertion validation for configured SSO protocol(s).
- Key and certificate rotation tolerance (JWKS refresh; SAML cert rollover).
- Replay protections (state/nonce and assertion ID cache).
- Deprovision test: remove user in IdP; access blocked within SLA.
- Role mapping test: group change updates permissions deterministically.
- Logging test: events appear with correct tenant scoping and identifiers.
- Lockout resilience: break-glass works without unsafe shortcuts.

9. Default Policy Profiles (Global Scale)

SIS-Core (Default): OIDC or SAML enabled; basic group-to-role mapping; audit logs 1 year; deprovision SLA max under 15 minutes.

SIS-Enterprise (Required for regulated / large orgs): SSO required + password disabled; SCIM required; session revocation enforced; audit logs 3+ years; deprovision target under 2 minutes; break-glass; Identity Console tools mandatory.

10. Client Statement (Standard Language)

SKYE supports enterprise identity the right way: SSO controls authentication, SCIM controls lifecycle and access, and every change is audited.

Appendix A. AE Compliance Checklist (One Page)

Use this checklist as the onboarding gate. A tenant cannot be marked GO until all required items are satisfied for the selected profile and the engineer preflight has passed.

TENANT / ORG INFO	
[ ] Org (Legal): _____	[ ] Tenant Slug/ID: _____
[ ] Primary Domain(s): _____	[ ] IdP Vendor: Okta / Entra / Google / Ping / Other
[ ] Target Go-Live Date: _____	[ ] Required Profile: SIS-Core / SIS-Enterprise
SSO CONFIGURATION	
[ ] Protocol: OIDC / SAML 2.0 / Both	[ ] Routing: Domain / Org Slug / Email Discovery
[ ] Redirect/callback URLs verified exact	[ ] OIDC Issuer recorded (if used)
[ ] SAML Entity ID + ACS + Cert recorded (if used)	[ ] SAML cert expiry > 60 days to go-live
ENFORCEMENT POLICY	
[ ] SSO Required (Enterprise: Yes)	[ ] Password login disabled when required
[ ] MFA enforced by IdP (Enterprise: Yes)	[ ] Session revocation defined
SCIM (PROVISIONING)	
[ ] SCIM enabled (Enterprise: Yes)	[ ] SCIM token stored in tenant vault
[ ] Create/Update/Disable supported	[ ] Deprovision SLA acknowledged
[ ] Group sync enabled (Enterprise: Should)	
GROUP TO ROLE MAPPING	
[ ] Roles model confirmed	[ ] Mapping documented (attach sheet)
[ ] Conflict rule: least privilege default	
ADMIN IDENTITY CONSOLE	
[ ] SSO status visible	[ ] SCIM status visible
[ ] Test tools present (SSO/JWKS/SCIM)	[ ] Break-glass section present
AUDIT LOGGING	
[ ] Auth events captured	[ ] SCIM events captured
[ ] Admin config changes captured	[ ] Retention set: 1y (Core) / 3y+ (Enterprise)
GO / NO-GO	
[ ] Engineer preflight attached	[ ] Decision: GO / NO-GO
[ ] AE Name: _____	[ ] Date: _____ Signature: _____

Appendix B. Engineer Preflight Script (Run Before Go-Live)

Run this script before any tenant goes live. Automated checks must pass, then complete the manual SSO steps.

sis_preflight.sh (part 1 of 6)

```
#!/usr/bin/env bash
# SKYE Identity Standard (SIS) — Engineer Preflight
# Version: SIS-1.0
#
# What it does:
# - Verifies SKYE identity discovery endpoint
# - Validates tenant identity status (SSO/SCIM enforcement flags)
# - Validates OIDC discovery + JWKS reachability (if configured)
# - Validates SAML cert expiry (if provided)
# - Runs SCIM CRUD smoke tests (Users + optional Groups)
# - Checks audit logging endpoints (basic reachability + last event presence)
# - Emits required manual SSO steps with pass/fail prompts
#
# Requirements:
# - bash, curl
# - jq optional (falls back to python for JSON parsing)
#
# Usage:
#   chmod +x sis_preflight.sh
#   APP_BASE="https://yourapp.example.com" \
#   TENANT="acme" \
#   SKYE_ADMIN_TOKEN="YOUR_ADMIN_BEARER_TOKEN" \
#   SCIM_TOKEN="YOUR_SCIM_BEARER_TOKEN" \
#   OIDC_ISSUER="https://issuer.example.com" \
#   ./sis_preflight.sh
#
# Optional:
#   SAML_CERT_PEM="/path/to/saml_cert.pem" ./sis_preflight.sh

set -euo pipefail

# ---------- Helpers ----------
red()   { printf "\033[31m%s\033[0m\n" "$*"; }
green() { printf "\033[32m%s\033[0m\n" "$*"; }
yellow(){ printf "\033[33m%s\033[0m\n" "$*"; }
blue()  { printf "\033[34m%s\033[0m\n" "$*"; }

need() {
  local name="$1"
  if [[ -z "${!name:-}" ]]; then
    red "MISSING env var: $name"
    exit 2
  fi
}

have_cmd() { command -v "$1" >/dev/null 2>&1; }

json_get() {
  # json_get <json> <jq_filter>
  local json="$1"
  local filter="$2"
  if have_cmd jq; then
    echo "$json" | jq -r "$filter"
  else
```

sis_preflight.sh (part 2 of 6)

```
    python3 - <<PY
import json,sys
data=json.loads(sys.stdin.read())
# Minimal filter support for common paths (e.g., .field, .a.b)
path="${filter}".strip()
if not path.startswith("."):
  print("")
  sys.exit(0)
keys=[k for k in path[1:].split(".") if k]
cur=data
for k in keys:
  if isinstance(cur, dict) and k in cur:
    cur=cur[k]
  else:
    print("")
    sys.exit(0)
print(cur if not isinstance(cur,(dict,list)) else json.dumps(cur))
PY
  fi
}

http() {
  # http <method> <url> <auth_header_optional> <data_optional>
  local method="$1"
  local url="$2"
  local auth="${3:-}"
  local data="${4:-}"
  local out
  if [[ -n "$data" ]]; then
    out=$(curl -sS -X "$method" "$url" \
      -H "Accept: application/json" \
      -H "Content-Type: application/json" \
      ${auth:+-H "$auth"} \
      --data "$data" \
      -w "\nHTTP_STATUS:%{http_code}\n")
  else
    out=$(curl -sS -X "$method" "$url" \
      -H "Accept: application/json" \
      ${auth:+-H "$auth"} \
      -w "\nHTTP_STATUS:%{http_code}\n")
  fi
  echo "$out"
}

status_code() {
  # Extract HTTP status appended by http()
  echo "$1" | sed -n 's/.*HTTP_STATUS:\([0-9][0-9][0-9]\).*/\1/p' | tail -n 1
}

body_only() {
  # Remove appended status line
  echo "$1" | sed '/^HTTP_STATUS:/d'
}
```

sis_preflight.sh (part 3 of 6)

```
fail() { red "FAIL: $*"; exit 1; }
pass() { green "PASS: $*"; }

# ---------- Inputs ----------
need APP_BASE
need TENANT
need SKYE_ADMIN_TOKEN

APP_BASE="${APP_BASE%/}"

ADMIN_AUTH="Authorization: Bearer ${SKYE_ADMIN_TOKEN}"

SCIM_TOKEN="${SCIM_TOKEN:-}"
OIDC_ISSUER="${OIDC_ISSUER:-}"
SAML_CERT_PEM="${SAML_CERT_PEM:-}"

blue "SIS Preflight starting..."
echo "APP_BASE=$APP_BASE"
echo "TENANT=$TENANT"
echo

# ---------- 1) SKYE Identity Discovery ----------
blue "[1/7] Identity discovery: $APP_BASE/.well-known/skye-identity"
resp=$(http GET "$APP_BASE/.well-known/skye-identity")
code=$(status_code "$resp")
body=$(body_only "$resp")

[[ "$code" == "200" ]] || fail "Discovery endpoint missing or not 200 (got $code). Required by SKYE standard."
version=$(json_get "$body" ".version")
scim_base=$(json_get "$body" ".scim_base")
audit_base=$(json_get "$body" ".audit_base")
tenant_status_path=$(json_get "$body" ".tenant_status_path")

[[ -n "$version" ]] || fail "Discovery missing .version"
[[ -n "$tenant_status_path" ]] || fail "Discovery missing .tenant_status_path"
pass "Discovery OK (version=$version)"
echo "scim_base=$scim_base"
echo "audit_base=$audit_base"
echo "tenant_status_path=$tenant_status_path"
echo

# ---------- 2) Tenant Identity Status ----------
blue "[2/7] Tenant identity status"
status_url="$APP_BASE${tenant_status_path//\{tenant\}/$TENANT}"
resp=$(http GET "$status_url" "$ADMIN_AUTH")
code=$(status_code "$resp")
body=$(body_only "$resp")
[[ "$code" == "200" ]] || fail "Tenant status not reachable (got $code) at $status_url"

sso_enabled=$(json_get "$body" ".sso.enabled")
sso_required=$(json_get "$body" ".sso.required")
sso_protocol=$(json_get "$body" ".sso.protocol")
scim_enabled=$(json_get "$body" ".scim.enabled")
mfa_required=$(json_get "$body" ".policy.mfa_enforced")
```

sis_preflight.sh (part 4 of 6)

```
pwd_disabled=$(json_get "$body" ".policy.password_login_disabled")
profile=$(json_get "$body" ".profile") # SIS-Core or SIS-Enterprise

echo "profile=$profile"
echo "sso.enabled=$sso_enabled  sso.required=$sso_required  sso.protocol=$sso_protocol"
echo "scim.enabled=$scim_enabled"
echo "policy.mfa_enforced=$mfa_required  policy.password_login_disabled=$pwd_disabled"

[[ "$sso_enabled" == "true" ]] || fail "SSO not enabled for tenant."
[[ "$sso_protocol" == "oidc" || "$sso_protocol" == "saml" || "$sso_protocol" == "both" ]] || fail "Invalid sso
if [[ "$profile" == "SIS-Enterprise" ]]; then
  [[ "$sso_required" == "true" ]] || fail "Enterprise requires SSO required=true"
  [[ "$pwd_disabled" == "true" ]] || fail "Enterprise requires password_login_disabled=true"
  [[ "$mfa_required" == "true" ]] || fail "Enterprise requires mfa_enforced=true"
  [[ "$scim_enabled" == "true" ]] || fail "Enterprise requires SCIM enabled=true"
fi
pass "Tenant status checks OK"
echo

# ---------- 3) OIDC Discovery + JWKS (if OIDC configured) ----------
blue "[3/7] OIDC checks (if applicable)"
if [[ "$sso_protocol" == "oidc" || "$sso_protocol" == "both" ]]; then
  [[ -n "$OIDC_ISSUER" ]] || fail "OIDC issuer required (set OIDC_ISSUER)."
  issuer="${OIDC_ISSUER%/}"
  disc_url="$issuer/.well-known/openid-configuration"
  blue "Fetching OIDC discovery: $disc_url"
  resp=$(http GET "$disc_url")
  code=$(status_code "$resp")
  body=$(body_only "$resp")
  [[ "$code" == "200" ]] || fail "OIDC discovery not reachable (got $code)."

  jwks_uri=$(json_get "$body" ".jwks_uri")
  authz_ep=$(json_get "$body" ".authorization_endpoint")
  token_ep=$(json_get "$body" ".token_endpoint")
  [[ -n "$jwks_uri" && -n "$authz_ep" && -n "$token_ep" ]] || fail "OIDC discovery missing required fields."

  blue "Fetching JWKS: $jwks_uri"
  resp=$(http GET "$jwks_uri")
  code=$(status_code "$resp")
  [[ "$code" == "200" ]] || fail "JWKS not reachable (got $code)."
  pass "OIDC discovery + JWKS OK"
else
  yellow "Skipping OIDC checks (tenant not using OIDC)."
fi
echo

# ---------- 4) SAML Cert Expiry (if SAML configured and cert provided) ----------
blue "[4/7] SAML cert checks (if applicable)"
if [[ "$sso_protocol" == "saml" || "$sso_protocol" == "both" ]]; then
  if [[ -n "$SAML_CERT_PEM" ]]; then
    [[ -f "$SAML_CERT_PEM" ]] || fail "SAML_CERT_PEM file not found: $SAML_CERT_PEM"
    if have_cmd openssl; then
      exp=$(openssl x509 -enddate -noout -in "$SAML_CERT_PEM" | sed 's/notAfter=//')
      echo "SAML cert expires: $exp"
```

sis_preflight.sh (part 5 of 6)

```
      pass "SAML cert parsed OK (ensure >60 days remaining manually or via policy)"
    else
      yellow "openssl not found; cannot parse SAML cert expiry automatically."
    fi
  else
    yellow "SAML in use but no SAML_CERT_PEM provided. Provide cert to validate expiry."
  fi
else
  yellow "Skipping SAML checks (tenant not using SAML)."
fi
echo

# ---------- 5) SCIM Smoke Tests ----------
blue "[5/7] SCIM checks (if applicable)"
if [[ "$scim_enabled" == "true" ]]; then
  [[ -n "$scim_base" ]] || fail "Discovery missing scim_base"
  [[ -n "$SCIM_TOKEN" ]] || fail "SCIM enabled but SCIM_TOKEN not provided."
  SCIM_AUTH="Authorization: Bearer ${SCIM_TOKEN}"
  scim_root="$APP_BASE${scim_base%/}"

  # Create a unique test user
  uid="sis-test-$(date +%s)"
  user_payload=$(cat <<JSON
{
  "schemas":["urn:ietf:params:scim:schemas:core:2.0:User"],
  "userName":"$uid",
  "name":{"givenName":"SIS","familyName":"Test"},
  "emails":[{"value":"$uid@example.invalid","primary":true}],
  "active":true,
  "externalId":"$uid"
}
JSON
)

  blue "SCIM Create User: $scim_root/Users"
  resp=$(http POST "$scim_root/Users" "$SCIM_AUTH" "$user_payload")
  code=$(status_code "$resp")
  body=$(body_only "$resp")
  [[ "$code" == "201" || "$code" == "200" ]] || fail "SCIM create user failed (got $code): $body"
  scim_user_id=$(json_get "$body" ".id")
  [[ -n "$scim_user_id" ]] || fail "SCIM create response missing user id"
  pass "SCIM create user OK (id=$scim_user_id)"

  blue "SCIM Get User"
  resp=$(http GET "$scim_root/Users/$scim_user_id" "$SCIM_AUTH")
  code=$(status_code "$resp")
  [[ "$code" == "200" ]] || fail "SCIM get user failed (got $code)"
  pass "SCIM get user OK"

  blue "SCIM Deactivate User (PATCH active=false)"
  patch_payload='{"schemas":["urn:ietf:params:scim:api:messages:2.0:PatchOp"],"Operations":[{"op":"Replace","p
  resp=$(http PATCH "$scim_root/Users/$scim_user_id" "$SCIM_AUTH" "$patch_payload")
  code=$(status_code "$resp")
  [[ "$code" == "200" ]] || fail "SCIM patch deactivate failed (got $code): $(body_only "$resp")"
```

sis_preflight.sh (part 6 of 6)

```
  body=$(body_only "$resp")
  active=$(json_get "$body" ".active")
  [[ "$active" == "false" ]] || fail "SCIM patch did not result in active=false"
  pass "SCIM deactivate OK"

  blue "SCIM Optional Delete User"
  resp=$(http DELETE "$scim_root/Users/$scim_user_id" "$SCIM_AUTH")
  code=$(status_code "$resp")
  if [[ "$code" == "204" || "$code" == "200" || "$code" == "202" ]]; then
    pass "SCIM delete OK (code=$code)"
  else
    yellow "SCIM delete not supported or blocked (code=$code). Disable-only can be acceptable if tenant uses d
  fi

else
  yellow "Skipping SCIM checks (SCIM not enabled)."
fi
echo

# ---------- 6) Audit Log Reachability ----------
blue "[6/7] Audit logging checks"
if [[ -n "$audit_base" ]]; then
  audit_url="$APP_BASE${audit_base%/}/events?tenant=$TENANT&limit=5"
  resp=$(http GET "$audit_url" "$ADMIN_AUTH")
  code=$(status_code "$resp")
  body=$(body_only "$resp")
  [[ "$code" == "200" ]] || fail "Audit events endpoint not reachable (got $code): $audit_url"
  pass "Audit endpoint reachable (verify events contain auth + scim + admin config entries)"
else
  yellow "Discovery missing audit_base; cannot verify audit endpoint."
fi
echo

# ---------- 7) Manual SSO Validation Steps ----------
blue "[7/7] Manual SSO validation (required)"
echo "MANUAL STEP A - Login:"
echo "  1) Open: $APP_BASE/org/$TENANT/login  (or your tenant login route)"
echo "  2) Complete IdP login."
echo "  3) Confirm you land in the correct tenant context."
echo "  PASS CRITERIA: correct tenant, correct user identity, no unexpected prompts."
echo
echo "MANUAL STEP B - Policy enforcement:"
echo "  - If profile is SIS-Enterprise:"
echo "    * Confirm password login is disabled."
echo "    * Confirm IdP MFA is enforced (user experiences MFA when policy demands)."
echo
echo "MANUAL STEP C - Deprovision cut-off:"
echo "  1) Disable a real test user in IdP (or via SCIM active=false)."
echo "  2) Attempt access using an existing session."
echo "  PASS CRITERIA: access revoked within SLA (Enterprise target <2 min; Core max <15 min)."
echo
echo "SIS Preflight script completed."
green "AUTOMATED CHECKS PASSED (manual steps still required for GO)."
```