Certificate KeyKit — спецификация и код

Статус: **реализовано**. Группа `cert_keykit`; в окне — страница `ЭКСПОРТ И ИМПОРТ -> СЕРТИФИКАТЫ`, рядом с `SSH` (тот же `secret_export`-жанр).

Зачем: ключи всё чаще привязаны к **TPM** (non-exportable) и умирают при переустановке/релокации. Эта секция:

1. показывает, **что переживёт переустановку** (exportable) и что намертво в TPM;
2. выгружает экспортируемые personal-сертификаты в password-protected `.pfx`;
3. выгружает публичные Root/CA в `.sst`, чтобы свежая машина снова доверяла тем же CA (корп-MITM-proxy, внутренний PKI);
4. импортирует `.pfx` / `.cer` / `.sst` обратно.

Папки те же две, что у остальных паков: экспорт пишет в `output\certificates`, импорт берёт файл, привезённый в `input`. Поле `Папка сертификатов` поэтому пустое по умолчанию — и таким должно остаться: значение по умолчанию GUI и CLI передают как настоящий параметр, а он перекрыл бы выбор папки по операции.

Жанр — governance/secret_export, не debloat. Только документированные механизмы (PowerShell PKI: `Get-ChildItem Cert:\`, `Export-PfxCertificate`, `Export-Certificate`, `Import-PfxCertificate`, `Import-Certificate`).

Безопасность (обязательно к прочтению)

- `.pfx` в `output\certificates` **содержит приватные ключи**. Это секрет того же класса, что OpenSSH KeyKit export и Wi-Fi key export (это выгрузка секретов): хранить вне публичных репозиториев, удалять временные копии.
- Пароль PFX передаётся через поле формы. Когда GUI уже elevated (обычный режим), он уходит как inline `-Command` аргумент (виден в списке процессов транзиентно, на диск не

пишется). В no-elevate-режиме с последующим UAC он короткоживуще попадает в `report\elevated_*.ps1` — после операции этот payload стоит чистить.

- TPM/non-exportable ключи **нельзя** забэкапить в PFX by design — статус честно помечает их `No (TPM / non-exportable)`, а export их пропускает с пометкой SKIP.

Инварианты

- Один `service:` — `system_core.services.devops_tools:certificate_keykit`, диспетч по `parameters.mode`.
- `id` уникальны по смыслу. Группа `id: cert_keykit`.
- Elevation вычисляется по выбранному store: `LocalMachine\*` → admin.
- YAML: 2 пробела на уровень.

Манифест

Вставить в `config/tool_manifest.yaml` **внутри** `utilities`**, сразу после блока** `ssh_keykit` (то есть перед `  - id: ripgrep_status`). Отступы как у `ssh_keykit` (`- id` на 6 пробелах).

```
      - id: cert_keykit
        title: "Certificate KeyKit"
        title_ru: "Сертификаты (экспорт/импорт)"
        description: "Inspect, export and import Windows certificates so exportable
keys survive reinstall; TPM-bound keys are flagged as non-exportable."
        description_ru: "Просмотр, экспорт и импорт сертификатов Windows:
экспортируемые ключи переживут переустановку, а TPM-привязанные помечаются как non-
exportable."
        fields:
          - id: cert_store
            type: "select"
            group: "target"
            label: "Certificate store"
            label_ru: "Хранилище сертификатов"
            default: "CurrentUser\\My"
            options:
              - value: "CurrentUser\\My"
                label: "CurrentUser\\My (personal)"
              - value: "LocalMachine\\My"
                label: "LocalMachine\\My (personal)"
              - value: "CurrentUser\\Root"
                label: "CurrentUser\\Root (trusted roots)"
              - value: "LocalMachine\\Root"
                label: "LocalMachine\\Root (trusted roots)"
              - value: "CurrentUser\\CA"
                label: "CurrentUser\\CA (intermediate)"
              - value: "LocalMachine\\CA"
                label: "LocalMachine\\CA (intermediate)"
          - id: cert_backup_dir
            type: "folder"
            group: "target"
            label: "Certificates folder"
            label_ru: "Папка сертификатов"
            default: ""
            placeholder: "output\\certificates"
            hint: "Where export writes: empty means output\\certificates. Import
takes the file named in Advanced, not this folder."
            hint_ru: "Куда пишет экспорт: пусто — output\\certificates. Импорт берёт
файл, указанный в Advanced, а не эту папку."
          - id: pfx_password
            type: "text"
            section: "advanced"
```

```
            label: "PFX password"
            label_ru: "Пароль PFX"
            default: ""
            hint: "Required for PFX export/import. Stored only for this run; the .pfx
is encrypted with it."
            hint_ru: "Нужен для export/import PFX. Используется только в этом
запуске; им шифруется .pfx."
          - id: import_file
            type: "file"
            section: "advanced"
            label: "Import file (.pfx/.cer/.crt/.sst)"
            label_ru: "Файл импорта (.pfx/.cer/.crt/.sst)"
            default: ""
            hint: "Put the file brought from another machine into input and pick it
here."
            hint_ru: "Привезённый с другой машины файл кладите в input и выбирайте
здесь."
          - id: import_store
            type: "select"
            section: "advanced"
            label: "Import target store"
            label_ru: "Целевое хранилище импорта"
            default: "CurrentUser\\My"
            options:
              - value: "CurrentUser\\My"
                label: "CurrentUser\\My (personal)"
              - value: "LocalMachine\\My"
                label: "LocalMachine\\My (personal)"
              - value: "CurrentUser\\Root"
                label: "CurrentUser\\Root (trusted roots)"
              - value: "LocalMachine\\Root"
                label: "LocalMachine\\Root (trusted roots)"
              - value: "CurrentUser\\CA"
                label: "CurrentUser\\CA (intermediate)"
              - value: "LocalMachine\\CA"
                label: "LocalMachine\\CA (intermediate)"
        children:
          - id: cert_status
            title: "Certificate status"
            title_ru: "Статус сертификатов"
            description: "List the selected store: subject, thumbprint, expiry,
private key and exportability (flags TPM/non-exportable keys)."
            description_ru: "Список выбранного store: subject, thumbprint, срок,
```

```
приватный ключ и exportability (помечает TPM/non-exportable)."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "safe"
            risk_level: "readonly"
            parameters:
              mode: "status"
          - id: cert_export_pfx
            title: "Export personal keys to PFX"
            title_ru: "Экспорт personal ключей в PFX"
            description: "Export exportable private-key certs from the selected store
to password-protected .pfx in output\\certificates. TPM-bound keys are skipped.
Output contains PRIVATE KEYS."
            description_ru: "Экспорт сертификатов с экспортируемым закрытым ключом из
выбранного store в password-protected .pfx в output\\certificates. TPM-ключи
пропускаются. Файлы СОДЕРЖАТ закрытые ключи."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "dangerous"
            risk_level: "secret_export"
            parameters:
              mode: "export_pfx"
          - id: cert_export_roots
            title: "Export store to SST (public)"
            title_ru: "Экспорт store в SST (публично)"
            description: "Export the selected store's public certificates (no private
keys) to a timestamped .sst bundle in output\\certificates."
            description_ru: "Экспорт публичных сертификатов выбранного store (без
закрытых ключей) в .sst с timestamp в output\\certificates."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "safe"
            risk_level: "project_write"
            parameters:
              mode: "export_roots"
          - id: cert_import_pfx
            title: "Import PFX"
            title_ru: "Импорт PFX"
            description: "Import the selected .pfx (with password) into the target
store, marking the key exportable. Changes the certificate store; reboot not
required. Pick file and password in Advanced."
            description_ru: "Импортировать выбранный .pfx (с паролем) в целевой
store, пометив ключ exportable. Меняет хранилище сертификатов; reboot не нужен. Файл
и пароль — в Advanced."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "dangerous"
```

```
            risk_level: "system_change"
            parameters:
              mode: "import_pfx"
          - id: cert_import_cert
            title: "Import certificate / CA"
            title_ru: "Импорт сертификата / CA"
            description: "Import a public .cer/.crt/.sst into the target store (e.g.
trust a corporate root CA). Changes trust; choose target store carefully. Pick file
in Advanced."
            description_ru: "Импортировать публичный .cer/.crt/.sst в целевой store
(например, доверие корпоративному root CA). Меняет доверие; выбирайте store
аккуратно. Файл — в Advanced."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "dangerous"
            risk_level: "system_change"
            parameters:
              mode: "import_cert"
          - id: cert_open_folder
            title: "Open certificate export folder"
            title_ru: "Открыть папку экспорта сертификатов"
            description: "Open output\\certificates; no certificate changes."
            description_ru: "Открыть output\\certificates; без изменений
сертификатов."
            service: "system_core.services.devops_tools:certificate_keykit"
            kind: "safe"
            parameters:
              mode: "open_folder"
```

Сервис

Добавить в `system_core/services/devops_tools.py` (module-level, рядом с `ssh_keykit`). Использует существующие `resolve_user_path`, `open_path`, `ps_quote`, `run_ps_command`, `context.paths.output`, `context.paths.input`.

```
_CERT_VALID_STORES = {
    "CurrentUser\\My",
    "LocalMachine\\My",
    "CurrentUser\\Root",
    "LocalMachine\\Root",
    "CurrentUser\\CA",
    "LocalMachine\\CA",
}


def _cert_store_or_default(value: Any, default: str) -> str:
    store = str(value or "").strip().replace("/", "\\")
    return store if store in _CERT_VALID_STORES else default


def certificate_keykit(context: JobContext) -> dict[str, object]:
    params = context.operation.parameters
    mode = str(params.get("mode") or "status").strip().lower()

    default_cert_dir = (
        context.paths.input if mode.startswith("import") else context.paths.output /
"certificates"
    )
    backup_dir = resolve_user_path(
        context, params.get("cert_backup_dir"), default=default_cert_dir
    )
    backup_dir.mkdir(parents=True, exist_ok=True)

    if mode == "open_folder":
        open_path(context, backup_dir)
        return {"folder": str(backup_dir)}

    store = _cert_store_or_default(params.get("cert_store"), "CurrentUser\\My")
    import_store = _cert_store_or_default(params.get("import_store"), store)
    password = str(params.get("pfx_password") or "")
    import_file = str(params.get("import_file") or "").strip()

    context.log(f"Certificate KeyKit mode: {mode}")
    context.log(f"Store: {store}")
    context.log(f"Backup dir: {backup_dir}")

    if mode == "status":
```

```
        script = f"""
$ErrorActionPreference = 'Continue'
$Store = {ps_quote(store)}
Write-Host ('=== Certificates in Cert:\\' + $Store + ' ===')
$items = @(Get-ChildItem -Path ('Cert:\\' + $Store) -ErrorAction SilentlyContinue)
if (-not $items) {{ Write-Host '(empty or not accessible from this context)'; exit 0
}}
foreach ($c in $items) {{
  $exportable = 'n/a (no private key)'
  if ($c.HasPrivateKey) {{
    $exportable = 'Unknown'
    try {{
      $rsa =
[System.Security.Cryptography.X509Certificates.RSACertificateExtensions]::GetRSAPriva
teKey($c)
      if ($rsa -and $rsa.Key -and ($rsa.Key.PSObject.Properties.Name -contains
'ExportPolicy')) {{
        if ($rsa.Key.ExportPolicy -band
[System.Security.Cryptography.CngExportPolicies]::AllowExport) {{
          $exportable = 'Yes'
        }} else {{
          $exportable = 'No (TPM / non-exportable)'
        }}
      }}
    }} catch {{ $exportable = 'Unknown' }}
  }}
  Write-Host ('-' * 60)
  Write-Host ('Subject    : ' + $c.Subject)
  Write-Host ('Thumbprint : ' + $c.Thumbprint)
  Write-Host ('Expires    : ' + $c.NotAfter)
  Write-Host ('PrivateKey : ' + $c.HasPrivateKey)
  Write-Host ('Exportable : ' + $exportable)
}}
Write-Host ('-' * 60)
Write-Host ('Total: ' + $items.Count)
exit 0
"""
        run_ps_command(context, script, check=False, progress_seconds=60.0,
elevated=store.startswith("LocalMachine"))
        return {"mode": mode, "store": store}

    if mode == "export_pfx":
        if not password:
```

```
            raise RuntimeError("PFX password is empty. Set 'PFX password' before
exporting private keys.")
        context.log("Exporting exportable private-key certificates to password-
protected .pfx files.")
        context.log("TPM-bound / non-exportable keys are reported as SKIP - they
cannot survive reinstall as PFX.")
        script = f"""
$ErrorActionPreference = 'Continue'
$Store = {ps_quote(store)}
$BackupDir = {ps_quote(backup_dir)}
$Password = ConvertTo-SecureString -String {ps_quote(password)} -Force -AsPlainText
$ok = 0; $fail = 0
foreach ($c in @(Get-ChildItem -Path ('Cert:\\' + $Store) -ErrorAction
SilentlyContinue | Where-Object {{ $_.HasPrivateKey }})) {{
  $out = Join-Path $BackupDir ($c.Thumbprint + '.pfx')
  try {{
    Export-PfxCertificate -Cert $c.PSPath -FilePath $out -Password $Password -
ChainOption BuildChain -ErrorAction Stop | Out-Null
    Write-Host ('OK   ' + $c.Thumbprint + '  ' + $c.Subject)
    $ok++
  }} catch {{
    Write-Host ('SKIP ' + $c.Thumbprint + '  ' + $c.Subject + '  -> ' +
$_.Exception.Message)
    $fail++
  }}
}}
Write-Host ''
Write-Host ('Exported: ' + $ok + ', skipped/non-exportable: ' + $fail)
Write-Host ('Backup folder (CONTAINS PRIVATE KEYS): ' + $BackupDir)
exit 0
"""
        run_ps_command(context, script, check=False, progress_seconds=300.0,
elevated=store.startswith("LocalMachine"))
        return {"mode": mode, "store": store, "backup_dir": str(backup_dir)}

    if mode == "export_roots":
        context.log("Exporting selected store's public certificates to an .sst bundle
(no private keys).")
        script = f"""
$ErrorActionPreference = 'Stop'
$Store = {ps_quote(store)}
$BackupDir = {ps_quote(backup_dir)}
$Stamp = Get-Date -Format 'yyyyMMdd_HHmmss'
```

```
$out = Join-Path $BackupDir (($Store -replace '\\\\','_') + '_' + $Stamp + '.sst')
$items = @(Get-ChildItem -Path ('Cert:\\' + $Store) -ErrorAction Stop)
if (-not $items) {{ Write-Host 'Store is empty; nothing exported.'; exit 0 }}
$items | Export-Certificate -FilePath $out -Type SST | Out-Null
Write-Host ('Exported ' + $items.Count + ' public certs to: ' + $out)
exit 0
"""
        run_ps_command(context, script, check=False, progress_seconds=120.0,
elevated=store.startswith("LocalMachine"))
        return {"mode": mode, "store": store, "backup_dir": str(backup_dir)}

    if mode == "import_pfx":
        if not import_file:
            raise RuntimeError("Import file is empty. Select a .pfx file in
Advanced.")
        if not password:
            raise RuntimeError("PFX password is empty. Set 'PFX password' to import a
.pfx.")
        context.log(f"Importing PFX into Cert:\\{import_store} (private key marked
exportable).")
        script = f"""
$ErrorActionPreference = 'Stop'
$ImportStore = {ps_quote(import_store)}
$ImportFile = {ps_quote(import_file)}
$Password = ConvertTo-SecureString -String {ps_quote(password)} -Force -AsPlainText
Import-PfxCertificate -FilePath $ImportFile -CertStoreLocation ('Cert:\\' +
$ImportStore) -Password $Password -Exportable |
  Format-List Subject, Thumbprint | Out-String | Write-Host
Write-Host ('Imported PFX into Cert:\\' + $ImportStore)
exit 0
"""
        run_ps_command(context, script, progress_seconds=120.0,
elevated=import_store.startswith("LocalMachine"))
        return {"mode": mode, "import_store": import_store, "import_file":
import_file}

    if mode == "import_cert":
        if not import_file:
            raise RuntimeError("Import file is empty. Select a .cer/.crt/.sst file in
Advanced.")
        context.log(f"Importing public certificate/CA into Cert:\\{import_store}.")
        script = f"""
$ErrorActionPreference = 'Stop'
```

```
$ImportStore = {ps_quote(import_store)}
$ImportFile = {ps_quote(import_file)}
Import-Certificate -FilePath $ImportFile -CertStoreLocation ('Cert:\\' +
$ImportStore) |
  Format-List Subject, Thumbprint | Out-String | Write-Host
Write-Host ('Imported certificate(s) into Cert:\\' + $ImportStore)
exit 0
"""
        run_ps_command(context, script, progress_seconds=120.0,
elevated=import_store.startswith("LocalMachine"))
        return {"mode": mode, "import_store": import_store, "import_file":
import_file}

    raise RuntimeError(f"Unknown Certificate KeyKit mode: {mode}")
```

Проверка после правки

```
runtime\python.exe -m py_compile system_core\services\devops_tools.py
runtime\python.exe -m pytest -q
install\verify_portable_env.cmd
```

Ручной обход:

- страница `СЕРТИФИКАТЫ` стоит рядом с `SSH` в разделе `ЭКСПОРТ И ИМПОРТ`;
- `Статус сертификатов` для `CurrentUser\My` печатает subject/thumbprint/expiry и колонку `Exportable` — TPM-ключи помечены `No (TPM / non-exportable)`;
- `Экспорт личных ключей в PFX` (с заданным паролем) создаёт `<thumbprint>.pfx` для экспортируемых и пишет SKIP для TPM-ключей; confirm-плашка появляется;
- `Экспорт store в SST` для `LocalMachine\Root` создаёт `.sst` без приватных ключей;
- `Импорт PFX` / `Импорт сертификата / CA` принимают файл и целевой store из блока `ДОПОЛНИТЕЛЬНО` формы.

Заметки

- Поле `pfx_password` — секрет; в журнал его не логируем (логируются только mode и пути). Не добавлять пароль в `field_updates`.