Constraint Expression Set Migration – Solution Design

1) Summary

A two-step Python toolchain migrates a single **Constraint Expression Set** (CML) between Salesforce orgs using Salesforce CLI for auth/context and REST API for data transport.

- **Exporter** pulls a narrow set of metadata rows and the **CML blob** into CSV + file artifacts.
- **Importer upserts** ExpressionSet & Context link, **re-maps** reference IDs (Products, Classifications, ProductRelatedComponents) by **natural keys**, recreates **ExpressionSetConstraintObj** rows, and uploads the **CML blob**.

The design favors safety (narrow scope, explicit remapping), repeatability (idempotent upserts), and transparency (printed SOQL & counts).

---

2) Goals & Non-Goals

Goals

- Migrate *one* Expression Set Definition & specific Version end-to-end (metadata + blob).
- Be repeatable, auditable (CSV outputs), and mostly idempotent in target org.
- Avoid bulk, org-wide moves; keep queries tight to limit blast radius.

Non-Goals

- Full PCM data migration (Products, Attributes, Classifications) — assumed pre-seeded.
- Cross-object diffing, conflict resolution UIs, or rollback of partial runs.
- Multi-version orchestration across many Expression Sets in one run.

---

3) Scope

In scope

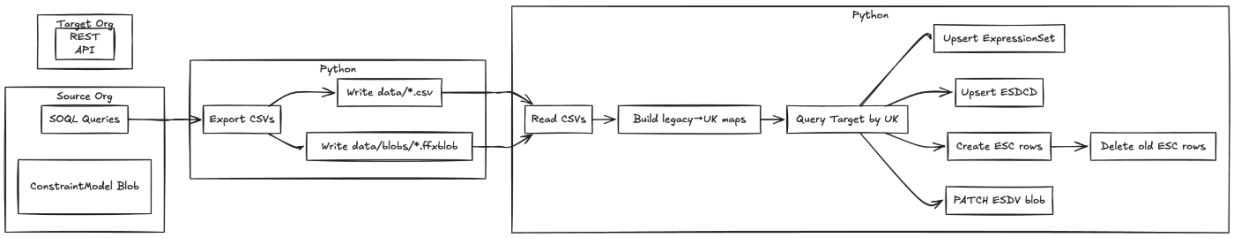
- ExpressionSet, ExpressionSetDefinitionVersion (ESDV), ExpressionSetDefinitionContextDefinition (ESDCD), ExpressionSetConstraintObj, and referenced Product2, ProductClassification, ProductRelatedComponent lookups.
- CML blob (ExpressionSetDefinitionVersion.ConstraintModel) download & upload.

Out of scope

- ContextDefinition creation; tool validates presence by DeveloperName.
- Attribute/PCM schema creation.

---

4) Architecture Overview



Key Externalities

- **sf org display —json** → Access token + instance URL resolution.
- **/services/data** → API version discovery (latest index).
- **REST endpoints** → SOQL query, sObject POST/PATCH/DELETE, ESDV blob PATCH.

---

5) Data Model & Key Fields

- **ExpressionSet**: `ApiName` (natural key), `UsageType`, `InterfaceSourceType`, etc.
- **ExpressionSetDefinitionVersion**: `DeveloperName` (= `${ExpressionSet.ApiName}_V${VersionNumber}`), `VersionNumber`, `ConstraintModel` (blob).
- **ExpressionSetDefinitionContextDefinition (ESDCD)**: link by `ExpressionSetDefinitionId` ↔ `ContextDefinitionId`.
- **ExpressionSetConstraintObj (ESC)**: per-object association with `ReferenceObjectId`, `ConstraintModelTag`, `ConstraintModelTagType`.
- **Reference Objects**: `Product2`, `ProductClassification`, `ProductRelatedComponent`.

Natural Keys (UK) used for remap in target

- `Product2`: `Name`.
- `ProductClassification`: `Name`.
- `ProductRelatedComponent`: composite `ParentProduct.Name | ChildProduct.Name | ChildProductClassification.Name | ProductRelationshipType.Name | Sequence`.

---

6) Detailed Flows

6.1 EXPORT FLOW (EXPORT_CML.PY)

1. **Args**: `--developerName`, optional `--version` (default `1`). Build `api_name_versioned = <dev>_V<ver>`.
2. **Auth**: `sf org display --json` (alias `srcOrg`).
3. **Resolve API version**: GET `/services/data/` → last item.version.
4. **Queries** (SOQL printed to console):
    - `ExpressionSetDefinitionVersion` filtered by `ExpressionSetDefinition.DeveloperName` & `VersionNumber`.
    - `ExpressionSetDefinitionContextDefinition` filtered by ESD DeveloperName.
    - `ExpressionSet` filtered by ESD DeveloperName.
    - `ExpressionSetConstraintObj` filtered by **ExpressionSet.ApiName** (equals dev name).

1. **Reference harvesting**: parse `ExpressionSetConstraintObj.ReferenceObjectId` prefixes → collect IDs for `01t` (Product2), `11B`(Classification), `0dS` (ProductRelatedComponent).
2. **Support exports**: fetch minimal subsets for referenced objects (by ID list).
3. **Blob download**: from `ExpressionSetDefinitionVersion.ConstraintModel` URL → save `data/blobs/ESDV_<dev>_V<ver>.ffxblob`.

6.2 IMPORT FLOW (IMPORT_CML.PY)

1. **Auth**: `sf org display --json` (alias `tgtOrg`). Resolve latest API version.
2. **Load**: read CSVs into memory.
3. **Upsert ExpressionSet**
    - Query by `ApiName`. If exists → PATCH (omit `ApiName` in body), else POST.
1. **Resolve IDs for ESDCD**
    - From CSV, get `ContextDefinitionApiName` → query `ContextDefinition.Id`.
    - From `ExpressionSet.ApiName` → query `ExpressionSetDefinition.Id`.
    - **Upsert ESDCD**: query by `ExpressionSetDefinitionId`; if present → PATCH `ContextDefinitionId`, else POST.
1. **Build legacy→UK maps** from exported reference CSVs; query target org to build UK→TargetId maps.
2. **Recreate ESC rows**
    - Query existing ESC rows for `ExpressionSetId` → capture IDs.
    - For each exported ESC row: replace `ReferenceObjectId` with target ID via maps; POST.
    - If all posts succeed → DELETE prior ESC rows; else retain (mixed state warning).
1. **Upload blob**
    - Base64 encode file and `PATCH` **ESDV** record by `DeveloperName` lookup, field `ConstraintModel`.

---

7) Idempotency & Safety

- **ExpressionSet**: upsert by `ApiName`.
- **ESDCD**: effectively upsert by `ExpressionSetDefinitionId`; updates only the `ContextDefinitionId`.
- **ESC**: create-new then conditional delete of old set → avoids mass delete if remap fails.
- **Blob**: last-write-wins PATCH; guarded by ESDV `DeveloperName` lookup.

Invariants before import

- Target contains the matching **ContextDefinition** by `DeveloperName`.
- Target contains **Products / Classifications / PRC** with UKs matching the source names.

---

8) Error Handling & Observability

- Exporter prints:

    - SOQL issued, record counts, and file save paths.
    - HTTP error payloads on failure.
- Importer prints:
    - Upsert outcomes (Created/Updated), unresolved UKs, counts of new ESCs, and deletion results.
    - Blob upload result (HTTP status 204 expected).
- **Partial failure policy**: if any ESC create fails → skip deletion and emit warning.

---

9) Security Considerations

- **Auth**: short-lived access token from `sf org display` (no refresh token stored).
- **Data at rest**: CSV & blobs written to local disk under `data/` — ensure repo/gitignore hygiene.
- **Data in transit**: HTTPS enforced by Salesforce endpoints.
- **Secrets**: no tokens checked into code; avoid printing tokens.

---

10) Configuration & CLI

- **Aliases**: `srcOrg`, `tgtOrg` (can be changed in code/flags if needed).
- **Inputs**:
    - Export: `--developerName`, optional `--version`.
    - Import: no args; reads `data/` files and uses alias `tgtOrg`.
- **Directories**: `data/` and `data/blobs/` are created if missing.

---

11) Performance & Limits

- Narrow SOQL reduces payloads; reference queries use `IN` filters by collected names.
- ESC recreation is row-by-row POST; acceptable for typical set sizes. For very large sets, consider composite or bulk API.
- API version discovery is O(1) list fetch; acceptable overhead.

---

12) Testing Strategy

Local

- Unit: pure helpers (`get_field_value`, UK builders, query builders) with small fixtures.
- Dry data: fabricate CSVs with 1–2 rows; run importer against a scratch org seeded with minimal PCM.

Integration

- Happy path: full export→import for a toy ExpressionSet.
- Negative cases:

- Missing ContextDefinition in target.
- Product name mismatch → unresolved UK logged; old ESCs preserved.
- Blob missing → importer warns, continues.

Acceptance

- Validate ExpressionSet visible & active as expected.
- Validate ESDCD links the correct ContextDefinition.
- Validate ESC rows match counts and tags.
- Validate blob content compiles/executes in target (CML smoke test).

---

13) Rollback & Recovery

- **ESC**: because old rows are deleted only after successful recreate, recovery is usually re-running after fixing UK mismatches.
- **ExpressionSet / ESDCD**: rerun is safe; last-write-wins.
- **Blob**: rerun export/import or restore from version control of `data/blobs`.

---

14) Extensibility & Future Enhancements

- **Flags**: add `--srcAlias`, `--tgtAlias`, `--dryRun`, `--keepOldEsc`.
- **Bulk**: use REST Composite or Bulk API v2 for ESC upserts.
- **Diff**: pre-flight diff of ESC sets with pretty print.
- **Mapping**: configurable UK strategies (e.g., external IDs) to avoid name collisions.
- **Multi-set**: accept a list of DeveloperNames; batch export/import.
- **Validation**: preflight check for ContextDefinition & PCM coverage with fail-fast.

---

15) Risks & Mitigations

- **Name collisions** in target PCM → unresolved IDs. *Mitigation*: external IDs or stricter matching.
- **Partial state** if network errors during ESC creation. *Mitigation*: idempotent rerun; optional transactional wrapper via Composite Tree.
- **Blob schema drift** if API version mismatched. *Mitigation*: always use latest version from instance; optionally pin.

---

16) Operational Runbook

1. Authenticate `srcOrg` and `tgtOrg` via `sf auth:web:login`.
2. Run exporter with `--developerName <ESD.ApiName> [--version N]`.
3. Verify CSVs & blob exist; skim counts.
4. Ensure target has required ContextDefinition & PCM (names match).

5. Run importer; confirm created/updated logs and 204 on blob.
6. Spot check in target org UI/API.

---

17) File Layout

```
export_cml.py
import_cml.py
README.md
/data
  ExpressionSet.csv
  ExpressionSetDefinitionVersion.csv
  ExpressionSetDefinitionContextDefinition.csv
  ExpressionSetConstraintObj.csv
  Product2.csv
  ProductClassification.csv
  ProductRelatedComponent.csv
  /blobs
    ESDV_<Dev>_V<Version>.ffxblob
```

---

18) Open Questions

- Should we pin a specific **API version** for stability instead of “latest”?
- Do we need **delete-or-replace** semantics for ESC rows gated by a checksum/diff?
- Should UK mapping become **configurable** per environment (e.g., name → external id)?