Export your UNS to RDF/OWL/GraphDB

Serialize the UNS back to an RDF/OWL graph file for diff, archive, or upload to a triple store.

Technical Reference **Solution** **Designer** **Settings and Tools** **Export** UNS as RDF/OWL Graph

Version 10.1.5+

N-Triples as the mental model. The Unified Namespace is a triple store by construction. Every UserType, Tag, AssetTree folder, and member corresponds to one or more RDF triples of the form `<subject> <predicate> <object> .` — the W3C N-Triples textual form. The export here is a serialization step onto that underlying triple model. See Industrial Ontology Integration How-to for the full standards map.

Primary path: the Export UNS format selector

1. In Designer, open **Solution Export**.
2. Click **Export UNS** on the toolbar.
3. In the **Format** dropdown pick one of:
   - **JSON**. The standard FrameworX-native multi-file dump (UnsUserTypes.json, UnsTags.json, etc.). Round-trips with the FrameworX importer; not a graph format.
   - **RDF/JSON** (`.rj`, `application/rdf+json`). W3C RDF/JSON. Niche but stable; loaded by dotNetRDF, GraphDB, Stardog, Fuseki.
   - **JSON-LD** (`.jsonld`, `application/ld+json`). JSON-LD 1.1 flattened form with `@context` and `@graph`. Mainstream modern format — schema.org, every major triple store, every LD client library.
   - **Turtle** (`.ttl`, `text/turtle`). The most-used RDF format in industrial-ontology workflows (IOF, ISA-88). Most readable for humans.
   - **N-Triples** (`.nt`, `application/n-triples`). One fully-expanded triple per line. Most universally accepted; best for diff and audit.
4. Click **Export UNS**. The output lands in the solution's Exchange folder under `<SolutionName>_UNS_<TAG>/UnsExport.<ext>` (folder tags: `RDF`, `JSONLD`, `TTL`, `NT`; the plain `JSON` choice writes to `<SolutionName>_UNS/` as multiple per-table files).

Round-trip in 10.1.5: RDF/JSON or N-Triples. All four W3C formats are fully implemented on the export side. The importer in 10.1.5 reads **RDF/JSON** and **N-Triples**; JSON-LD and Turtle readers ship in a follow-up release. If you need to re-import what you exported, use `rj` or `nt` end-to-end — the underlying triple stream is identical, so either works. Publishing to a triple store, sharing with an ontology team, or archiving — pick whichever format your destination accepts.

Alternative path: AI-driven via the MCP tool

```
export_graph_model(format="rj", policy="iof")
# Aliases accepted: jsonld | json-ld | turtle | ttl | ntriples | n-triples | nt
```

See skill Skill Export UNS as RDF.

Round-trip fidelity

Faithful round-trip at the UserType / Tag / member-def level. Columns expand back to triples:

FrameworX column	OWL emission
`SourceIri`	Subject IRI (class, individual, or predicate). `/Attr` tail stripped on export.
`Description`	`rdfs:comment`
`DisplayText`	`rdfs:label`
`Labels` (semicolon-delimited)	`skos:altLabel` per entry
`BaseUserType` (UDT only)	`rdfs:subClassOf` parent UDT's registered IRI
`Attributes` (parsed JSON)	Individual triples keyed by `prefix:localName`. `@lang`-suffixed keys emit language-tagged literals.

What does NOT round-trip

- `owl:Restriction` flavor (`someValuesFrom` / `allValuesFrom` / cardinality). FX captures target type, not restriction style.
- `owl:unionOf` / `owl:intersectionOf` anonymous classes. Flattened at import.
- Blank nodes outside RDF list structure. Collapsed at import.

Ontology stewards who need full OWL axiom fidelity should keep their source ontology as the truth and use FrameworX for operational data.

Runtime modifications and determinism

The exporter reads **cold-start** `.dbsln` values so the same solution produces the same graph every time. Runtime modifications to `@Tag.X.Attributes` made by scripts do NOT round-trip automatically. If you want runtime updates in the next export, run the `promote_retentive_attributes` MCP tool first. It copies the retentive overlay back into the cold-start columns.

No SPARQL push (10.1.5)

Direct POST-to-SPARQL-endpoint is not in 10.1.5. Upload the exported file to GraphDB / Stardog / Fuseki manually. Direct push is on a future roadmap.

Related: Industrial Ontology Integration How-to · Import an OWL/RDF ontology into your UNS · Generate a visual report of your UNS · Local AI Ontology Demo

---