RDF Triples and the Industrial Ontology Foundry (IOF)

FrameworX consumes ontologies as **RDF triples** — the universal data shape behind every OWL/RDF graph, including the Industrial Ontology Foundry (IOF) suite. This page explains the triples concept and walks through importing IOF into a FrameworX solution.

Version 10.1.5+

What is a triple?

A **triple** is the atomic unit of RDF (Resource Description Framework). Each triple states ONE fact in three parts:

- **Subject** — the entity the fact is about
- **Predicate** — the property or relation
- **Object** — the value or related entity

Example triples about a single piece of equipment:

```
:Mixer_M101  rdf:type           s88:Unit .
:Mixer_M101  rdfs:label         "High-Shear Mixer M-101" .
:Mixer_M101  s88:capacityL      "200"^^xsd:double .
:Mixer_M101  s88:hasEquipmentModule  :Heater_H101 .
:Heater_H101 rdf:type           s88:EquipmentModule .
```

An entire ontology is just a (potentially huge) collection of triples. The 8-class ISA-88 ontology shipped with the LocalAI KnowledgeGraph Demo carries a few hundred triples; a real industry ontology like IOF carries hundreds of thousands.

FrameworX reads triples from three serialization formats — pick whichever your tooling produces:

- `.rj` — RDF/JSON (W3C standard, JSON-encoded)
- `.nt` — N-Triples (one triple per line, plain text)
- `.ttl` — Turtle (compact, human-readable — the example above)

The Unified Namespace itself is a triple store by construction — every UserType, Tag, AssetTree folder, and member corresponds to one or more RDF triples. See Industrial Ontology Integration How-to for the full architectural alignment.

The Industrial Ontology Foundry (IOF)

The Industrial Ontology Foundry (IOF) publishes the de-facto industry-standard ontology suite for manufacturing — covering classes for Process, Material, Asset, Production, Maintenance, Quality, and Supply-Chain domains. IOF is built on the Basic Formal Ontology (BFO) upper-level framework and aligned with ISO 15926, ISA-95, and OAGIS standards. Home: industrialontologies.org.

Why care about IOF in a SCADA/HMI context? Customer assets modeled in IOF terms can flow into the FrameworX UNS as a typed Object Model, giving you:

- **A pre-built UserType library** — every IOF class becomes a FrameworX UDT (Pump, Bioreactor, ChromatographyColumn, BatchRun, etc.), ready to instantiate as equipment in the UNS.
- **Standards alignment** — your UNS terms match what the rest of the customer's enterprise systems use (ERP, MES, asset management).
- **Round-trip portability** — equipment metadata can flow back out to external knowledge graphs and dashboards via `export_graph_model`.

The IOF release is published as RDF/JSON (`.rj`) and other RDF formats — download from industrialontologies.org or use a customer-supplied excerpt.

Importing an IOF ontology into a FrameworX solution

IOF is a **schema-only** ontology (TBox) — it defines classes, properties, and axioms but ships zero equipment instances (ABox). The import workflow has two stages.

Stage 1 — Schema import (creates the UserType library)

Run `import_graph_model` (Designer MCP) on the IOF `.rj` file with a schema-only policy:

```
import_graph_model(
    source="file",
    file_path="C:/.../IOF_OntologyPlusExampleTriples.rj",
    policy="iof",
    dry_run=true
)
# review the summary, then re-run with dry_run=false
```

The importer creates one UserType (UDT) per IOF class. For the full IOF release, expect ~1,046 UserTypes. Curate via policy if you only need a subset (e.g., Process + Asset, skip Supply-Chain).

Stage 2 — Equipment instantiation (manual, dual-shape pattern)

IOF carries no equipment instances; you author them with `write_objects` using the dual-shape pattern (see Industrial Ontology Integration How-to Two paradigms):

- **Naked UserType instance** at `<path>/<entity>` — for each real-world piece of equipment, instantiate the IOF UDT (e.g., `Plant/Reactor_R101` typed `iof:Bioreactor`), with dynamic members (Temperature, Pressure, Speed) wired to Devices, Scripts, or simulators.
- **Optional `/Attr` envelope** at `<path>/<entity>/Attr` — for static ontology-derived metadata about the entity (rdfs:label, manufacturer, design constants like Capacity_L). All `/Attr` members carry `StartValue`.

The LocalAI KnowledgeGraph Demo shows this end-to-end on a small ISA-88 (not IOF) ontology — the same dual-shape pattern scales directly to IOF.

What you get end-to-end

An IOF-grounded UNS is a SCADA solution that speaks the customer's industry vocabulary natively. Equipment that the customer references as `iof:CentrifugalPump` in their MES becomes a FrameworX UDT with the same name and the same IRI. Live process data flows on naked instances; ontology metadata flows on `/Attr` envelopes; both round-trip to external knowledge-graph systems.

See also

- Import an OWL/RDF ontology into your UNS — the importer reference, including the predicate-mapping table.
- Industrial Ontology Integration How-to — the full dual-shape pattern + standards map.
- LocalAI KnowledgeGraph Demo — worked example on ISA-88.
- UNS UserTypes Reference — UDT modeling pattern.
- Export your UNS to RDF/OWL/GraphDB — serializing UNS back to RDF formats.
- industrialontologies.org — IOF home, ontology downloads.
