Technical Report

ISO/TR 22126-2

First edition
2025-01

Financial services — Semantic technology —

Part 2: OWL representation of the ISO 20022 metamodel and e-repository

Services financiers — Technologie sémantique —

Partie 2: Représentation OWL du métamodèle et du référentiel de l'ISO 20022

Reference number
ISO/TR 22126-2:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 68, *Financial services*, Subcommittee SC 9, *Information exchange for financial services*.

A list of all parts in the ISO 22126 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The purpose of this document is to demonstrate the feasibility and explore the design space for the representation of the ISO 20022 e-Repository in OWL DL, a decidable dialect of the OWL ontology language. This document demonstrates multiple representations of the auth.016 sample messages illustrating possible modelling choices.

The current e-Repository is constructed in a purpose-built system comprising multiple layers of object-oriented modelling built on the Eclipse Modeling Framework, a specialization of the Essential Meta-Object Facility (EMOF) standardized by the Object Management Group (OMG). This system can create class definitions and object instances in a computer language such as Java[1)] that serve as containers for data, but does not define the semantics, or meaning, of those objects. The EMOF model is a foundation to write software on (a "blank slate") but requires custom software to establish the meaning and behaviour of those objects.

In OWL DL, on the other hand, it is possible to construct concepts based on logical definitions. Although custom software can still be necessary, an OWL DL ontology has a meaning and defined behaviours when used with OWL standard tools. Out-of-the-box an OWL ontology works with ontology editors, reasoners and graph databases that allow queries with the SPARQL language. An OWL ontology can be used together with other OWL ontologies. Unlike more expressive systems such as ISO Common Logic, however, OWL DL is based on a subset of first-order logic that is decidable so that in addition to reasoning instances (for instance querying instances that match a logical definition) it can prove the consistency of an ontology. (As OWL is defined in terms of first-order logic, OWL axioms can be exported to other logic-based tools that can be more expressive, such as theorem provers, or more application-oriented, such as business rules engines.)

It is common for schemas and documentation for message families such as ISO 20022, FIX Protocol and SWIFT MT to be written in formats idiosyncratic to the family. Although ISO 20022 messages conform to an XML schema, authoritative definitions exist in the e-Repository as a set of Ecore-serialized objects specific to ISO 20022. The elements of FIX messages (which can be encoded in various wire formats) are defined in FIX Orchestra, an XML file which is specific to the FIX trading community. Tools designed to work with the schema of one standard do not natively work for other standards.

Applications that process financial messages have their own schemas that exist either explicitly or implicitly, for instance a set of tables, constraints, stored procedures and other projects in a relational database, or a hierarchy of classes in a programming language such as Java. Although disparate descriptions for the structure of messages and software systems fulfil the requirements of each system, technologists lack the global view necessary to take rapid and correct action.

To have that global view, it is desirable to bring schemas and instances from disparate systems into a single system which can be queried to support software development, make visualizations and do inference. RDF is a suitable framework for this: not only does it come with standard vocabularies such as RDFS, OWL and SHACL for representing schemas, but it can represent arbitrary data structures as a graph of nodes that possess datatype properties and are interconnected with object attributes. Some major features of RDF are:

a) the XML schema datatypes are available to represent common primitive data types that occur in general computing;

b) nodes and predicates, relationships between nodes, are named with URIs to establish a global namespace in which terms from arbitrary vocabularies can be combined;

c) text strings are tagged with ISO 639 language codes to provide a straightforward mechanism to represent labels and descriptions in multiple languages.

It is possible, as seen in Clause 7, to represent any kind of message in RDF through a simple form of mechanical translation – it possible in RDF to work without a schema of any kind. Clauses 9 and 10 demonstrate that, with more effort, it is possible to express the e-Repository and ISO 20022 messages as an OWL DL ontology.

1) Java is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

The EMOF form of the e-Repository is built in a number of layers starting from the M3 layer (the foundation of EMOF), the M2 layer (an object model used to describe the M1 model) and the M1 layer (definitions of business components, message components, etc.). Users of the e-Repository can create instances of objects defined at the M1 layer in the M0 layer that define individual messages or business objects. For the most part, the ontology is contained in the M1 layer and defined in terms of an idiosyncratic vocabulary defined in the M2 layer. This vocabulary is fit for purpose, but incompatible with other schema and ontology definition systems.

OWL DL requires a flatter organization where there is a clear separation between ontology objects (definitions of classes and properties) and instance objects (instances of the defined classes.) Most terminology from the M2 layer is replaced with terminology from the OWL standard, making the ontology automatically interoperable with other ontologies. Properties that cannot be expressed with built-in OWL properties, such as XML element names for the message components, are represented with annotation properties which, as defined in the ontology, not participate in inference. Objects in the e-Repository which are not naturally treated as class or property definitions are defined as instances in OWL DL with corresponding definitions derived from the M3 layer.

Users of this ontology can add additional instance and ontology objects. OWL, in particular, allows users to write logical definitions in terms of specific lists of objects, the attributes of objects and logical combinations such as AND, OR and NOT thereof. This makes it possible for a business or regulator to define a particular kind of transaction (a subclass) in terms of size, the time it took place, the security traded, the attributes of the organizations involved, etc. Annex A demonstrates that many categories defined and discussed in regulation and documents are straightforward to express in this manner, a significant step towards making business documents interpretable by machines.

Financial services — Semantic technology —

Part 2: OWL representation of the ISO 20022 metamodel and e-repository

1 Scope

This document is concerned with the representation of the ISO 20022 e-Repository contents in RDF and OWL by developing a case study around the ISO 20022 auth.016 sample message (hereafter simply referred to as "auth.016"). This includes:

a) transformation of the sample message into an RDF instance graph;

b) demonstrating a set of SPARQL rules that transform the auth.016 message into a FIX TradeCaptureReport(35=AE) message (hereafter simply referred to as "FIX AE");

c) expressing the metamodel, business components and message components exactly with a custom RDF vocabulary;

d) representing those schemas as OWL schemas using OWL vocabulary when possible and annotation properties otherwise;

e) creating instance graphs for the auth.016 sample messaging using the vocabulary of the business components and message components.

This document also discusses the choices that arise in structuring RDF documents equivalent to documents in XML, and FIX Tag-Value format balancing considerations such as preserving the order of parts of the message versus creating graphs that are suitable for RDFS and OWL inference.

2 Normative references

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

No terms and definitions are listed in this document.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.2 Abbreviated terms

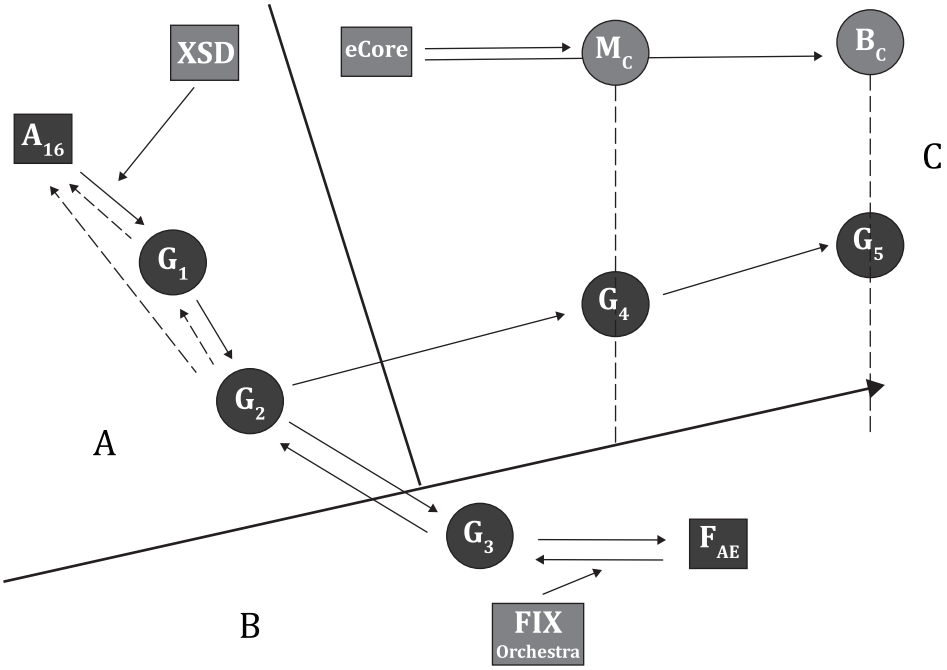
API	application programming interface
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
EMOF	Essential Meta-Object Facility
FIX	Financial Information Exchange
JVM	Java Virtual Machine
LEI	Legal Entity Identifier
MOF	Meta-Object Facility
OMG	Object Management Group
OWL DL	Web Ontology Language – Description Logic
RDF	Resource Description Framework
RDFS	RDF Schema
RMI	Remote Method Invocation
SHACL	Shapes Constraint Language
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
SPIN	SPARQL Inferencing Notation
SWIFT MT	SWIFT Message Type
SWRL	Semantic Web Rule Language
UML	Unified Modelling Language
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier
XML	Extensible Markup Language
XSD	XML Schema Definition

4 Overview

4.1 Graph transformations

4.1.1 General

Figure 1 depicts the data transformations described in this document.



Key

A auth.016 XML message

B FIX AE message

C graphs aligned to message and business components

A_{16} sample auth.016 message as XML document

G_1 A_{16} translated mechanically into RDF retaining element order

G_2 simplified graph that removes inessential information such as element ordering from G_1

G_3 a FIX AE message equivalent to A_{16} expressed as key-value pairs in RDF

F_{AE} G_3 converted to a complete FIX AE tag-value message

G_4 G_2 rewritten using vocabulary defined in M_c (message components)

G_5 G_4 rewritten using vocabulary defined in B_c (business components)

M_c message Components Ontology in OWL

B_c business Components Ontology in OWL

NOTE Circles represent RDF graphs. Squares represent data in some other format. Blue lines represent transformations that have been implemented. Red lines implement transformations that remain unimplemented. Blue objects represent instance information (a message). Green objects represent schema and ontology information.

Figure 1 — Data transformations of RDF to and from other formats

A_{16} is the auth.016 sample message in XML format. In Clause 7, this message is represented as the G_1 and G_2 graphs which are close to the structure and vocabulary of the XML document. Furthermore, a set of production rules are demonstrated that can convert the A_{16} message into an equivalent F_{AE} FIX AE message, a practical data transformation task.

Clauses 8 and 9 develop representations of the message components M_c and business components B_c in RDF. Clause 8 represents the contents of the e-Repository translated mechanically while Clause 9 represents the contents of the e-Repository in OWL.

In Clause 10 the message is converted to the G_4 and G_5 graphs. The G_4 graph precisely represents the A_{16} content using vocabulary from M_c, a process which can be mechanically implemented. The final destination is the G_5 graph which uses vocabulary from B_c, which is shared between multiple messages. Complete capture of the message's meaning, however, requires additional terms that do not exist in B_c.

4.1.2 Graphs derived from the XML document

The method used to create G_1 is mechanical and can be tuned to convert a wide range of XML documents to RDF graphs.

Graph G_1 was produced by combining the A_{16} sample message with information from the XSD schema. XML element and attributes were rewritten to RDF property names by concatenation: for instance, an element named `<Pric>` is rewritten as the property `ns:Pric` where is `ns` is a chosen RDF namespace prefix. Leaf nodes of the XML document are written as datatype properties while upper nodes are connected with object properties. The XSD schema provides default values and resolves question such as "does this instance of the string `'55'` represent an integer or a character string?".

G_1 retains information about the order of elements in the document as well as information about where nodes (in terms of line number and character position) appear in the source document. As the order of elements is not essential to the meaning of the message, graph G_2 is a simplified graph that strips away information idiosyncratic to the source document.

4.1.3 Conversion to a FIX AE message

The FIX AE message is equivalent to the auth.016 in meaning and purpose. The G_3 graph represents the meaning of the auth.016 sample message in a vocabulary based on the FIX tag-value standard. A FIX tag such as `35` is written with the predicate `fix:f35` where `fix` is a chosen namespace, `f` means "field" and `35` is the tag number. As with the XML document, the tree structure of the FIX message is represented as a tree of RDF nodes.

The G_3 graph was created from the G_2 graph by matching patterns in G_2 to equivalent patterns in G_3, a process that can be performed in either direction. The process of creating G_3 does most of the work to creating a FIX message equivalent to A_{16}. The remaining task to write a tag-value FIX message is to write tag-value pairs from G_3 the right order, calculate a checksum, append headers and attend to other details based on information from FIX Orchestra.

4.1.4 Graphs aligned with message and business components

Although the G_2 graph captures enough meaning to translate the message, it is unsuitable for RDFS and OWL inference. For example, A_{16} represents the price at which a trade took place such as:


```
<Pric>
     <Pric>
          <MntryVal>
               <Amt Ccy="EUR">13.5</Amt>
          </MntryVal>
     </Pric>
</Pric>
```

Specifically, the element `<Pric>` embeds another `<Pric>` element inside itself. auth.016 allows the outer price element to contain either a `<Pric>` or a `<NoPric>` element depending on whether a price is specified. As a result, in the G_1 and G_2 graphs the `ns:Pric` predicate is used in two different parts of the graph with two different meanings. This is allowed in RDF (SPARQL queries can be written against such a graph) but RDFS and OWL expect that the same meaning is intended wherever a predicate is used.

The G_4 graph is structurally similar to the G_2 graph and continues to represent the complete content of the message. The main difference is that G_4 is written in a vocabulary based on the message components, which, unlike the elements of the XML document, have a context-independent meaning. The same vocabulary is used in the M_c OWL schema compatible with the G_4 graph.

The G_4 graph represents messages accurately at the cost of:

a) not sharing concepts between messages;

b) having little semantic information about the meaning of message parts.

For instance, multiple messages in the e-Repository concern concepts such as financial transactions and assets. In the message components, different objects are used to represent different financial transactions in different messages. As such, every message part can be said to have precisely the meaning it is expected to have in that message.

Some objects from the e-Repository (such as message definitions, message components and choice components) are represented as OWL classes, while other objects (such as message attributes, message association ends and message building blocks) are represented as OWL properties. While translating and XML document, properties are used to represent XML elements and attributes while classes represent the data types contained inside elements and attributes.

The G_5 graph writes the auth.016 message in terms of the business components, which are more general than message components and can be reused between messages. Aligned with the M_c graph, the G_5 graph continues the process of discarding idiosyncrasies of the message in favour of the essential meaning of the message. The cost of this, however, is that the business components are insufficient to capture 100 % of the meaning of the message. (Ideally the G_5 graph can be used for message processing tasks such as creating the G_3 graph to write a FIX message, however, the lossless nature of G_2 made translation G_2 directly to G_3 a safe approach to the translation task.)

NOTE The e-Repository contains data objects (such as such as Business Area and a Message Definition Identifier) which are not naturally represented as properties and classes records and classes. Objects of this type can be represented as RDF instances conforming to classes and properties imported from the metamodel. Objects that map to OWL schema objects have properties such as XML tag names and registration status that do not exist in the OWL vocabulary, but these can be added to the OWL ontology in the form of annotation properties to capture 100 % of the content of the e-Repository.

Annex A considers an alternative approach to representing the business components in which message-related objects from the e-Repository are exclusively and consistently represented as objects as opposed to the alternating classes and properties in the G_4 graph. In this representation, a single property, denoting that one message part directly contains another message part, is the only property used to build message instances. At the cost of introducing additional nodes, this representation can capture the sequencing of message parts in both the schema and instances by adding sequence numbers to the RDF nodes.

5 Methods

5.1 Multiple local RDF graphs

Much of the work described in this document was done using in-memory graphs from the Python[2] rdflib library. Like a client-server triple store (e.g. OpenLink Virtuoso, Ontotext GraphDB, AllegroGraph[3]), an in-memory graph can be queried with SPARQL. The in-memory graph also allows direct interaction with individual triples and nodes which can be helpful when working with list and tree structures. The Jena framework provides similar in-memory graphs which are used internally with the Protégé ontology viewer and editor.

In-memory graphs scale well for graphs in the range of 1 to 10 million triples and are adequate for handling schemas the size of the ISO 20022 e-Repository; a system that processes instance messages can have a few large graphs that for schemas and large number of small graphs which contain individual instance messages.

Unlike the client-server triple store, in-memory graphs are lost when the application shuts down. If the process for creating the graph is repeatable, it can be constructed from the source material as necessary. Alternately, the graph can be serialized to a Turtle[1] or other RDF format file and later restored or exported to other RDF tools. Client-server triple stores can handle much larger data sets than in-memory stores (billions of triples) and can persist triples for long periods of time. Most client-server triple stores work on

2) Python is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

3) AllegroGraph is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

RDF data sets, which can contain a number of named RDF graphs, a mechanism to represent a collection of multiple RDF graphs.

Working with structures involving blank nodes can be awkward with a client-server triple store. One issue is that complex structures are traversed in a way that requires many round trips between the client and server with long latencies compared to an in-memory graph. Although most client-server databases provide facilities to work with blank nodes, those facilities are not completely portable and can require adaptation to work with various triple stores.

A client-server triple store applies to a situation where multiple parties are making changes to a graph through an API: for instance, a system that provides a web interface to edit an ontology.

5.2 Open linked data

Open linked data is a major trend in the RDF world, which centres around the use of http-based RDF resources such as `<https://dbpedia.org/resource/Leipzig>` which corresponds to the Wikipedia[4] page for the German city. It is possible to retrieve this document using an http client such as curl. In that case, the DBpedia web server returns a document with facts extracted from Wikipedia about that city in a format such as Turtle or RDF/XML[2] as specified in the `Accept` http header.

Linked data provides a simple way to browse a data set: instead of downloading all facts in a graph (which can contain billions of facts as in the case of DBpedia) it is possible to traverse the graph subject-by-subject and selectively retrieve only the necessary facts out of a large database. A global view can be had efficiently by downloading the entire database and loading it into a triple store, but linked data are practical when it is necessary to traverse just a small part of the graph.

RDF ontology projects can interact with linked data in two ways:

a) a project can incorporate data published in linked data form;

b) it can publish linked data.

This document is based on data entirely from the e-Repository, sample messages and FIX Orchestra and did not require additional linked data.

Linked data are a potential information source for research and development. For instance, concepts from the e-Repository can be aligned to concepts in generic databases such as DBpedia.

Linked data poses risks for applications, such as production software that works with financial message. Linked data, for instance, depends on a working internet connection and linked data services being online. Linked data can be updated without warning, and such changes can break an application. The continuing function of an application depends on the availability of data and requires that the linked data publishers maintain their service indefinitely. Linked data can be curated together with other data to produce operational graphs which are inspected, tested, versioned and proven fit for purpose.

The graphs described in this document are not published on the web and thus primarily use URIs that are non-resolvable, such as

```
<urn:iso:std:iso:20022:tech:xsd:auth.016.001.01/>
```

The urn scheme is a global namespace with prefixes uniquely registered amongst organizations, as does the http scheme. Unlike the http scheme, however, there is no expectation that a urn URI be resolvable.

If and when an official and stable RDF translation of the e-Repository is ready, that translation can use http URIs and have the facts published on the web. In fact, this option is open to any organization which wishes to publish their own version of the ISO 20022 e-Repository, or that wishes to publish their own instance documents using vocabulary defined in the e-Repository.

4) Wikipedia is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.