Gellish Syntax for Formal Languages

Definition of the Gellish Expression Format

including a definition of

the Gellish Contextual Facts

Version 6.3

March 2016

Author: Dr. Ir. Andries van Renssen
Gellish.net
http://www.gellish.net
e-mail: info@gellish.net

The Gellish family of formal languages is defined in two parts:

1. This document describes its syntax.
2. The Gellish Taxonomic Dictionary-Ontology of Formal English, or one of the formal Gellish Taxonomic Dictionaries-Ontologies in other formalized natural languages, which defines the semantics.

In addition to that the language definition is described in the book 'Semantic Information Modeling in Formalized Languages' and its application is described in the book Semantic Information Modeling Methodology.

The latest version of the Taxonomic Dictionary-Ontology of Formal English and Formal Dutch can be purchased or licensed via the web shop: http://www.gellish.net/
The Taxonomic Dictionary-Ontology contains definitions of concepts, including also definitions of kinds of relations. The definitions include definition models as well as textual definitions expressed in natural English. The concepts are arranged in a subtype-supertype hierarchy (taxonomy). This is the semantics part of the definition of the formal language
Formal language expressions can be stored in Universal Semantic Databases and can be exchanged between systems using Message files or Queries, all expressed in a Gellish Expression Format. This document defines the structure of expressions (the definition of the syntax) in such databases, message files and queries.

Change history

Version	Date	Modification relative to previous version
4.0	March 2003	Definition of subset models defined.
5.0	August 2007	Gellish database description clarified.
5.1	June 2008	Queries and table header definition extended.
5.2	February 2011	Implementation guidelines and unique key discussion added. Subset Business Model extended.
6	December 2012	Contextual facts description improved. Universal Gellish Expression Format tables described as integration of Naming Tables and Fact Tables. UID of intention, inverse indicator, author of copy and addressee added.
6.1	March 2014	Restructured
6.2	Nov 2015	Extent, probability and exponent added as auxiliary facts
6.3	March 2016	Phrase type UID added as language independent indicator of first and second role player columns.

Table of Content

1. Introduction

Semantic modeling and semantic databases and messages are based on the principle that the meaning of expressions is explicitly represented in the models and not as separate documentation about the models, databases or messages. Semantic modeling thus requires a semantic modeling language that provides mechanisms for making meaning explicit. The definitions of the concepts, especially the definitions of kinds of relations and the syntax (the sentence structures) of that language determine its semantic expression capabilities. It appears that a formal language can be defined with a universal basic semantic structure for all its expressions that is necessary and sufficient for the expression and interpretation of any expression in that language. That language enables to make statements that express any kinds of knowledge, requirements and definitions as well as any opinion or fact about individual things, and also to express questions and answers, promises, commands, etc. This universal basic semantic structure enables to define universal semantic databases as well as a universal structure for data exchange messages (see http://www.gellish.net/semantic-modeling.html).
This document specifies such a universal semantic database, exchange messages and queries structure.

Semantic models should be unambiguously interpretable by computers. Therefore, the terminology that is used in semantic expressions that make up semantic models should not be free, but need to be predefined in the form of semantic definition models. And the formalized terms should be available in a formal electronically dictionary. Such a dictionary should contain such semantic definition models of concepts. Such definition models require that the dictionary is not an ordinary dictionary, but that the concepts are related to each other as in a taxonomy and an ontology. Thus for English, a Formal English taxonomic dictionary-ontology is required to define the Formal English language. The Gellish Taxonomic Dictionary is such a formal language defining dictionary-ontology. When in this document the term 'the dictionary' or 'the formal dictionary' is used, this Taxonomic Dictionary-ontology is meant or an equivalent one of another formalized natural language, such as the taxonomic dictionary of Formal Dutch ('Taxonomisch Woordenboek van Formeel Nederlands').

One of the possible syntaxes for formal languages is defined in this document. Formal expressions can be exchanged between computers as messages in formal language and they can be stored in semantic databases. Formal languages are standardized structured subsets of natural languages that can be understood by humans as well as by computers. They are not a programming language, nor a data definition languages. Formal languages use ordinary words, terms and expressions from natural languages. They do not invent their own terminology, such as Esperanto did. However, formal languages restrict the way in which sentences (expressions) may be made and therefore they standardized a large number of phrases and terms. For example, the Taxonomic Dictionary of Formal English includes ordinary words, such as car and wheel, length and color, buying and transporting, etc. and also includes a large number of standard English phrases, such as 'is a part of', 'is located in' and 'can have as part a'. With those phrases sentences can be made, such as: a car can have as part a wheel. The standard phrases are defined and documented in the Upper Ontology section of the Taxonomic Dictionary.

Taxonomic dictionaries of formal languages are special and powerful, because they not only specify the meaning (semantics) of the components (words and phrases) in formal language expressions, but they also relate synonym terms, abbreviations and codes to the unique identifiers (UID) of their common concepts and they distinguish homonyms (terms with different meanings) by allocating different UIDs to them. The dictionaries can also include pictures to support or illustrate definitions. The concepts in such a dictionary are arranged in a subtype-supertype hierarchy that is compliant with their definition and the dictionary relates concepts to other concepts when that can be derived from their definition. Therefore the Taxonomic Dictionary is also called a Taxonomy.

Semantic expressions are not only expression of *main units of communication (main idea's, facts or opinions),* but each unit of communication needs to be accompanied by a number of *contextual facts* in order to provide a context for interpretation. Main unit of communication (or 'main facts' for short*)* are the core units of meaning that one want to store or communicate. For example, the fact that 'the Eiffel tower is located in Paris'. Contextual facts are facts that provide the context of a unit of communication which adds details about the main unit of communication, such as the intention with which it is communicated, its status, who expressed the fact, when that was done, in which context it is valid, etc., etc.

This document specifies how main units of communication *(such as facts and opinions)* are expressed in semantic models and it specifies a standard collection of obligatory and optional contextual facts: *the Gellish set* of *kinds of contextual facts*. Main units of communication (main facts) and their contextual facts may be stored and exchanged in Gellish Expression Tables or in any public or proprietary syntax.

Most conventional database and message data structures can only store a subset of formal expressions and are not able to contain the rich semantic expression capabilities of the complete formal language. Therefore, this document also specifies universal data structures for Semantic Databases and Data Exchange Messages. Databases and messages that are based on that specification are suitable to store or exchange units of communication of any kind, such as facts about Products, Knowledge representations, E-Business messages and Database queries and responses.

Conventional databases typically consist of some or many tables, each of which is composed of a number of columns. The definition of those tables and columns determine the storage capabilities of the database, whereas the relations between the columns define the kinds of facts that can be stored in such a database. Those columns and relations determine the database structures that defines the expression capabilities of the databases. Similar rules apply for information that is exchanged between systems in electronic data files.

This conventional database technology has some major limitations:

- When data was not covered during the database design and thus is not included in the data model, then such data cannot be stored in the database nor exchanged via such a data file structure.
- Different databases have different data structures, which causes that data in one database cannot be integrated with data from other databases nor exchanged between databases without dedicated data conversion.
- A database modification or extension requires a redesign of the database table structures and modification of the software, which is a complicated and costly exercise.

New Semantic Web technologies, such as RDF or OWL based data storage and data exchange have more flexibility. They allow any kind of relation and have no constraints on terminology, thus they do not standardize the language that is used in the databases and messages. This means that those technologies require additional standards to define a common language for use in different databases and between parties. Formal languages are designed to act as such common languages, which can be used on their own, or in combination with ‘meta languages’ such as RDF and OWL.

Semantic databases and message files for data exchange that apply formal languages do not have the limitations of conventional databases, nor the unconstrained conventions of languages such as RDF, XML and OWL. Formal languages provide flexibility by specifying primarily the general semantics of expressions and they standardize the languages, by providing standard relation types and definitions of concepts and terminology, together with guidelines for (proprietary) extensions. In addition to that this document specifies a number of contextual facts about each main unit of communication (also called meta data about each main ‘fact’). These semantic expressions can be implemented in any meta language, including for example RDF or OWL. However, formal languages can also be implemented in the powerful tabular Gellish Expression Formal data structure, in which any opinion or fact can be expressed, together with their contextual facts, as is specified in this document.

This document specifies how units of communication, such as facts and queries should be expressed in formal languages. It defines a syntax for *formal expressions* as well as the *Gellish set of kinds of contextual facts,* presented in the form of the definition of the *Gellish Expression Format* for databases, message files and queries that can be filled with data written in a *formal language.*

1.1 Gellish data structures for formal languages

Gellish expressions and semantic interpretation rules have a universal data structure and syntax, which implies that information, knowledge and requirements that are expressed in a formal language can be presented in multiple files that all have the same data structure, such as table columns and column definitions. The format is suitable for Messages as well as Databases and Queries. That universal format is called:

- *The Gellish Expression Format*

The Expression Format basically consists of one kind of table: the *Expression Table* or an equivalent form for expressing the same content, such as in case of the *Gellish RDF Expression Format*. The Expression Format enables the use of multiple aliases and languages for the same concepts and enables to keep track of two which alias or translation is used in which expression. Therefore we will explain the format as if it consists of two kinds of tables (that are combined into one *Expression Table*):

- *Naming Tables*
  Tables that are intended to contain the relation between UIDs and terms and aliases, such as synonyms, abbreviations, codes, etc. in multiple languages and language communities.
- *Fact Tables*
  Tables that contain main facts and contextual facts, only using UIDs and no names.

Queries can be expressed using the same Gellish Expression Format, extended with a few extra fields, thus defining a:

- *Query Format*
  Queries can be expressed using the same format extended with a few query fields.

Formal language databases may be implemented as collections of Expression Tables or, when there is no need to keep track of which term is used in which expression, they may be implemented as combinations of Naming Tables and Fact Tables. The collections of tables may be either centrally managed or may be distributed, whereas the distributed sections may be independent of each other, provided that the allocation of UIDs and the usage of UID ranges is coordinated. This coordination process is described in the Gellish Modeling Methodology Part 1A, General Principles and Guidelines.

This document defines the expressions of main facts and contextual facts as well as their presentation in the form of the above mentioned kinds of Gellish Expression Tables.

The Gellish Expression Tables have three fundamental characteristics:

1. The Gellish tables enable to store information about *any kind of thing*. Individual things of any kind are enabled through explicit classification of the individual things by any of the kinds of things that are defined in the extensible Gellish Dictionary. This differs from conventional databases that predefine object types by a limited number of entity types and attribute types as are predefined in a conventional data model.
2. The Gellish tables enable to store *any kind of possible fact* about things or kinds of things. This is enabled through the expression of possible facts as (collections of) binary relations between things, whereas each relation is explicitly classified by relation types that can either be selected from the standardized relation types that are defined in the Dictionary or from relation types that are added to the Dictionary as proprietary extensions. Furthermore, the expressions can express any kind of intentions, because each expression of a fact can be accompanied by an intention (illocutionary force), which indicates whether the expression is a statement, a question, an answer, a promise, a command, etc.
3. The Gellish tables enable to store standard *kinds of contextual facts* about each main fact, such as its language, validity context, intention, scale, status, date of start or end of validity, author, etc.

As a consequence, Universal Semantic Databases or Message files do not need to be modified or extended when the scope of an application changes. Furthermore, different Universal Semantic Databases or files can be combined, merged and integrated, and act as one distributed database, whenever required without a need for data conversion, provided that the unique identifiers are managed.

The Gellish data structure is simple and even human readable. Furthermore, it supports the simultaneous use of multiple languages, because it contains the option to express in which language a fact is expressed. This is supported by the fact that various natural language specific versions can be defined that share the same unique identifiers (UIDs) for the same concepts. For example, Formal English is a formalized subset of natural English and thus uses natural English terminology, whereas Formal Dutch is a formalized subset of natural Dutch and thus uses natural Dutch terminology.

1.2 Natural language variants and automated translation

In principle, there is a variant language for each natural language, depending on the availability of a translation of the terms by which the concepts are denoted. For example, The Formal English Dictionary defines Formal English and the 'Formeel Nederlands Woordenboek' defines Formeel Nederlands (Formal Dutch). Dictionaries in other languages are in preparation. International terminology (such as most units of measure and mathematical concepts) is included in the Dictionary as International language.

Gellish formal languages use a unique identifier (UID) for each thing. This includes also user objects, concepts that are defined in the Dictionary as well as facts and relation types. This enables the use of synonyms and homonyms. It also enables that a computer can automatically translate an expression in a certain language into an expression in another language, provided that dictionaries for those formal language variants are available. Such an automated translation is possible because of the fact that the meaning of a formal expression is captured as a relation between the unique identifiers, so that the meaning is language independent.
This adds power to cooperation, such as via the Semantic Web, because a message can be created e.g. in Formal English whereas it can be presented in another formal language variant, while the computer software has invisibly done the translation.

1.3 Documentation

Gellish formal languages are defined in databases that are themselves written in a formal language. Gellish formal languages are a further development of the technology on which the ISO standards ISO 10303-221 and ISO 15926 are based and are compliant with ISO 16354 – Guidelines for Object Modeling and Knowledge Modeling. A Universal Semantic Database is suitable to contain at least all information that can be expressed according to the ISO 10303-221, ISO 15926 and ISO 12006-3 standards.

Guidance on how to use Formal languages for the expression of information, knowledge and requirements, can be found in the following documentation:

- The Gellish website at http://www.gellish.net/.
- The books 'Semantic Information Modeling in Formalized Languages' and 'Semantic Information Modeling Methodology'.

1.4 Formal languages syntax and semantics

The definition of formal languages includes a semantic part and a syntax part:

- **Semantics**
  The definition of the *semantics* (the meaning) that is needed to interpret formal expressions consists of two universal basic semantic structures, together with definitions of concepts and the specification of terms (names, codes, numbers, synonyms, etc.) and phrases to denote those concepts. This is described in [Ref. 1]. Each concept is represented in Gellish by a unique identifier (UID). Each definition consists of a definition model. A definition model

comprises a collections of expressions of facts (including main facts as well as contextual facts). The names and phrases that denote the concepts determine the terminology that may be used in Formal expressions, apart from the terms that users attach to the UIDs of the individual things that they define. The semantics of concepts (the definition models) is contained in the Dictionary. The basic collection of definition models is provided in the Upper Ontology section.
This document defines the kinds of contextual facts that may accompany expressions of main facts. The collection of kinds of contextual facts is called the *Gellish set of kinds of contextual facts*[1].

- **Syntax**
  The definition of the structure of formal expressions (*syntax*) includes a structure in the form of Gellish Expression Tables. Those Tables can be implemented in any tabular form, such as in SQL or in Spreadsheet form or even in CSV form. The base facts and contextual facts that are defined for the Gellish Expression Tables can also be expressed in other format. For example as collections of Triples (as in RDF/XML or OWL) or Quads (as in Trix) as described in ISO 15926-11. This document provides the definitions of the Gellish syntax in the form of Gellish Expression Tables for databases, messages and queries with their RDF and XML equivalent representations.

For an extended description of formal languages and its capabilities see the Gellish Wiki (http://www.gellish.net/index.php/gellishwiki/view,plugin.html) and the book ‘Semantic Information Modeling Methodology’.

[1] The Gellish set of kinds of contextual facts about main facts is comparable with the Dublin Core of meta data (contextual facts or attributes) about ‘resources’, where a ‘resource’ typically is a file (document) or web page, but usually not a fact. In Gellish, references to files and documents and facts about them are included in expressions of main facts.

2. Expression of idea's

A semantic model requires relations for the expression of the core idea (the topic with the intention of the expression, also called the main fact), the roles that are played by the related objects, relations to express queries and relations to express contextual facts for each main fact. All those relations for each main fact relate a number of 'expression components'. The following paragraphs discusses those expression components and the relations between them. Precise definitions of the expression components and kinds of relations between them are provided in chapter 5.

2.1 Expression of core ideas (main facts)

A semantic model requires the following relations for the expression of an idea (a main fact) and the contextual facts that complete the expression of its meaning (the semantics):

1. A *binary relation* that relates two objects that are involved in a main fact.

   The two related things are: a player of the first role that is required by the relation (usually[2] the left hand object) and a player of the second role that is required by the relation (usually the right hand object). The objects are represented in the relation by their respective unique identifiers (UIDs).

2. A *classification* of the relation.

   This is a fact that classifies the relation by a (standard) kind of relation. This contextual fact is represented by a pair of things: the UID of the Fact and a UID of the kind of relation that classifies the relation.

3. A qualification or quantification of the extent in which a statement is the case.

   This is an auxiliary fact that specifies the fraction for which the main expression is the case. Typically a fraction (possibly expressed as a percentage weight or volume) that expresses the extent in which a part in a composition relation is a fraction of the whole, or the fraction of a mixture for which a classification by substance holds.

4. A classification of a quantification relation by a scale (a *unit of measure*) – when applicable.

   This is an auxiliary fact that classifies a relation by a (standard) kind of scale (also called a unit of measure). Typically this classifies a quantification relation, but it may also classify an extent in which a composition relation or a classification relation is the case (see below on 'extent'). This auxiliary fact is represented by a pair of things: the UID of the fact and a UID of the kind of scale that classifies the relation.

5. A qualification of the expression by the *intention* with which the expression is communicated.

   This is an auxiliary fact that expresses with which intention the main fact is communicated. For example, it may express that the fact is communicated as a statement or as a denial, a confirmation or a question.

Any expression of a main fact in a formal language should therefore consists of the above relations, whereas those relations relate six component. Those expression components can be represented by UIDs that are independent of any natural language. The components are presented in Table 1.

Component ID (column ID)	**Description of object**
1	UID of a fact (idea)
2	UID of a left hand object
60	UID of a relation type
15	UID of a right hand object

[2] In Formal English expressions it is allowed that inverse phrases for kinds of relations are used, but in a Fact Table, inverse phrases may not be used. Thus in a Fact Table the left hand object is always the player of the first role.

30	UID of an extent
66	UID of a scale (unit of measure)
5	UID of an intention

Table 1, The core elements of an expression

When those six component are arranged in a syntactic structure that defines the above describes relations between them, then that structure forms an expression. Thus the expression of a main fact is an arrangement of six components, consisting of six UIDs, in a syntactic structure.

2.1.1 Fact Table core

A tabular syntactic structure is a possible implementation of an expression as the relations between the columns in the table define the relations between the components of the expression. Therefore the core of a Fact Table can represent the above expression components and their relations and thus enables the expression and interpretation of the meaning of main facts. That core of a Fact Table is therefore defined by six columns, identified by a column ID. Each row in a Fact Table represents a combination of the six components, each represented by its own UID.

For example, Table 2 is a core of a Fact Table that illustrates the expression of fact 201.

1	2	60	15	66	5
UID of a fact	**UID of a left hand object**	**UID of a relation type**	**UID of a right hand object**	**UID of a UoM**	**UID of an intention**
201	101	5026	102	570423	491285

Table 2, A Fact Table with the identification and expression of one main fact

Each object that is represented in column 2 of Table 2 is called a left hand object and denotes the player of a ‘first role’ in a relation, as defined in the definition model of the relation types (see the upper ontology in TOPini). By analogy column 15 denotes a right hand object, which is the player of the ‘second role’ in the relation of the specified kind.

The UIDs in Table 2 represent objects (things). The terms (names, etc.) of those objects in natural language are specified in expressions of separate contextual facts, which are called naming relations, as is specified in the paragraph 2.5. Those naming expressions are typically provided in a Naming Table (see par 2.4.1.1). Replacing or accompanying UIDs by columns with terms that denote the UIDs delivers a human readable equivalent for Table 2.

Usually a Naming Table will contain various synonyms terms for the objects that are denoted only by a UID in a Fact Table. Therefore, in principle it is necessary to record which term is used in which expression. This can be done by creating a separate Term Usage Table that contains a specification of which terms are used for each UID that appears on a row in a Fact Table. However this issue is solved by using integrated Expression Tables (see par. 4).

2.2 Expression of roles of role players

Each object that is involved in a relation plays a role of a particular kind in that relation. Thus each binary relation implies two roles played by the related objects.
Those roles often may remain implicit in expressions, because the relation type implies particular kinds of roles. For example, the relation type <is a part of> (a composition relation) implies the kinds of roles ‘part’ and ‘whole’. The definition of those kinds of roles follows from the *definition* of the kinds of relations (as specified in the TOPini section of the Dictionary).

In some cases it is required to model those roles explicitly. This especially holds for the modeling of constraints, when those constraints are applicable only when objects of a kind play a role of a particular kind.
Roles can be made explicit by modeling a role as a separate object in either of two ways:

1. By treating a role in the same way as a role player. This means that a fact that an object plays a particular role in a relation, is explicitly expressed by two relations:
   - A relation between the object and the role that is played by that object.
   - A relation between a relation and the role of the kind that is required by that relation.

These kinds of expressions of facts about roles can be stored in the same way as all other expressions of facts. It only requires the recording of all the roles as separate objects and explicit classification of those roles and defining kinds of roles in their own subtype-supertype hierarchy (taxonomy).

2. By inserting a left hand and right hand kind of role in an orderly expression. This implies that for each main fact four contextual facts are defined: two that specify that the objects play roles of those kinds and two that specify that those kinds of role roles are subtypes of the kinds of roles that are by definition required by a relation of such a kind.

In a tabular form this means that a Fact Table is extended with two additional columns; one for the left hand kind of role and another for the right hand kind of role, whereas also the names of the kinds of roles need to be defined in a Naming Table. The kinds of roles classify the played roles and are implied subtypes of the kinds of roles that are required by the relation type. This is implemented in a Fact Table as follows:

1	2	72	60	74	15	66	5
UID of a fact	UID of a left hand object	UID of a left hand kind of role	UID of a relation type	UID of a right hand kind of role	UID of a right hand object	UID of a scale	UID of an intention
201	101	301	5026	401	102	570423	491285

Table 3, Fact Table core extended with explicit roles

Thus the explicit modeling of the roles implies an extension of the core of a fact table with the following columns:

Component ID (column ID)	Description of object
72	UID of a left hand kind of role
74	UID of a right hand kind of role

Table 4, Extension of the core of a Fact Table with roles

2.3 Expression of queries

The modeling of a dialogue (being a human activity) typically requires modeling the various communication activities as separate occurrences. The questioning, answering, confirmation, etc. are modeled as activities that are classified by kinds of activities. However, with or without modeling the dialogue itself, it is also required to model a question or query as a message. The general model of a query message is discussed in [Ref. 1].

The three components of an expression that express contextual facts for terms in queries are specified in Table 5.

Component ID (column ID)	Description of object
80	Left hand string commonality
81	Right hand string commonality
82	Relation type string commonality

Table 5, String commonality columns in a Query Table

These components imply additional relations between these components and the left hand term (character string), the right hand term and the name of the relation type respectively that should be interpreted from the syntactical structure of the expression.

A tabular implementation enables to interpret the relations from the definition of relations between the columns. Therefore, a query can be implemented as a Query Table. Such a Query Table is an

Expression Table that contains three additional columns (80, 81 and 82) in which the commonality criteria for the left hand and the right hand term and the name of the relation type can be specified.

2.4 Expression of contexts

A proper interpretation of the meaning of an expression (or proposition) requires not only that the main fact (the topic) is expressed with the terms in the user preferred language and language community, but it also requires that the expression includes information about the context in which an expression is made. Therefore, semantic modeling not only requires expressions of the main facts themselves, but it also requires that each expression is accompanied by additional expressions of facts about the main fact. Such additional facts are called 'contextual facts' about a main fact.

Contextual information is also required for the management of information. For example, for proper interpretation as well as for proper information management it should be recorded who has created an expression, when that was done, what the status of the expression is, since when it is outdated or replaced by another expression of a fact, in what language it is expressed, etc.

Each expression of a fact shall be accompanied by such contextual facts. Standard kinds of contextual facts are discussed below.

2.4.1 Language and language community contexts

Terms (names) of things and phrases are language and language community context dependent. In a formalized language that uses natural language independent UIDs to represent things, facts and ideas as well as the components in expressions of those facts and ideas are represented by UIDs. Thus the expressions are also language independent.

The relations between language independent UIDs and natural language and language community dependent terms and phrases (names) are specified in naming relations.

Thus, a naming relation is a relation between a term (possibly being a phrase) that is used in any expression and the thing (UID) that is denoted by that term, whereas the naming relation is classified by a relation type. Every naming relation requires a relation with a language community that is the contexts in which the naming relation is valid. Every term requires a relation between the term and a natural language, which specifies that the term belongs to the vocabulary of that language. Those four relations form a collection of contextual binary relations or triples (also called a ‘graph’) that forms a naming model, being a model of the naming of anything.

The components of a naming model that includes the language and language community contexts are given in Table 6.

Component ID (column ID)	Description of object
69	UID of a natural language
71	UID of a language community
101	Term (name, phrase, abbreviation, code, URI, number or symbol)
60	UID of relation type (of the naming relation)
2	UID of a thing that is denoted by the ‘term’ in the language, as originated in the language community

Table 6, Components for the expression of language and language community context

The relations between these components are implemented through the syntactical structure or format of expressions.

The definitions of the language and language community are given in the following paragraphs. The definition of the components 101, 60 and 2 were already provided in par. 2.1)

2.4.1.1 Naming Table

In a tabular implementation the relations between the components are defined by the definition of the relations between the columns in the table. For example, each of such a collection of contextual relations can be represented in tabular form as one Naming Table, provided that the relations between the columns in that table represent the kinds of relations for that collection.

Such a Naming Table therefore has the following table header:

69	71	101	60	2
UID of a language	UID of a language community	Term	UID of relation type	UID of a named thing

Table 7, Header of a Naming Table

The columns 69, 71 and 101 together form a unique key, which means that a combination of those three items may occur only once in the table..

The columns have their own column ID's that uniquely identifies the columns, independent of a natural language. This enable that the column titles are free text descriptions that can vary per language or user preference.

69	54	71	16	101	60	2
UID of the language of the term	**Name of the language**	**UID of the language community**	**Name of the language community**	**Term (name)**	**UID/Name of relation type**	**UID of named thing**
910036	English	193263	engineering	pump	5117/is a name of	130206
910037	Dutch	193263	engineering	pomp	5117/is a name of	130206
910038	German	193263	engineering	Pumpe	5117/is a name of	130206
910036	English	190668	linguistics	German	5117/is a name of	910038
910038	German	190668	linguistics	Deutsch	5117/is a name of	910038
910037	Dutch	190668	linguistics	Duits	5117/is a name of	910038
910036	English	193259	ontology	assembly relation	5117/is a name of	1190
910036	English	492015	Gellish	is a part of	1981/is a synonym of	1190
910036	English	492015	Gellish	has as part	1986/is an inverse of	1190
910036	English	492015	Gellish alternative	is a whole of	1986/is an inverse of	1190

Table 8, Naming Table with UIDs and names of a concept in various languages

The use of a Naming Table is illustrated in Table 8 on three examples:

1. The concept represented by UID 130206 is denoted in English as pump, in German as Pumpe and in Dutch as pomp. The language community where these names originate is 'engineering'. Table 8 illustrates how those various names in those three languages are allocated to the concept that is denoted by UID 130206.
2. Table 8 also illustrates an example of how names of languages differ in various languages. For example, the name of the German language, expressed in German is Deutsch and in Dutch it is Duits. Table 8 illustrates how the various names of the German language are related to the concept that is denoted by UID 910038.
3. The third example gives the names of a relation type, its denoting phrase as a synonym, its inverse phrase and an alternative for the phrase.

Table 8 should be interpreted as follows:

- The table has two header rows. The numbers in the first row, 69, 54, 71, 16, 101, 60 and 2, are standardized natural language independent identifiers of the table columns. They refer to standard columns in Expression Tables as is described later in this document. The texts on the second line are not-standardized names of those columns.
- The second and the fourth columns (54 and 16, in red) are added for clarification, but are semantically superfluous and are not part of a standard Naming Table.

- The UID of the language in the first column (69) specifies the language in which the term in column (101) is expressed. Thus the number 910036 on the first row, which is the Gellish UID of the English language, specifies that the term ‘pump’ is an English term for concept 130206. Similarly, UID 910037 denotes the Dutch language and UID 910038 denotes the German language.
  Note that the fourth line specifies that the term ‘English’ is the English name of the language that is represented by the UID 910036.
- Column 60 denotes a UID of the relation type. In order to facilitate the readability of the example table the name of that relation type is given in addition, although that name is superfluous and does not belong to a Naming Table. Note that the UID of the relation type could also indicate other relation types, such as ‘is an abbreviated name of’ or ‘is a code for’ and some other variations. If the UID is 1986, then the ‘name’ consists of a phrase that denotes that in a relation the left and right terms are switched to express the same fact as when other phrases are used.
- The columns 69, 71 and 101 together form a unique key for the table.

Not only all dictionary concepts, but also each user-defined concept or individual thing (user defined object[3]) that is used in formal expressions of main facts shall have a Gellish UID. Each user defined object UID shall be unique and shall be allocated conform the rules for allocation of UIDs (as described in [Ref. 1].

2.4.2 Contextual facts

Each main fact is accompanied by a number of prime and secondary contextual facts. Together that collection of facts is called the *expression context*, which is a set of kinds of contextual facts. Each of the contextual facts (which are specified below) is expressed as a binary relation that relates a pair of objects and a classification of that relation. The classification relation and the classifying relation type that classifies the relation may remain implicit in implementations (for example in a tabular implementation where they are defined by the definitions of the columns and the relations between the columns that make up an expression). However it depends on the kind of implementations whether the contextual relations can be interpreted from these relations and thus whether they should be made explicit in order to enable semantic interpretation. The latter is for example the case in RDF implementations.

2.4.2.1 Prime contextual facts

The objects that are specified in the following table imply relations that express prime contextual facts. Definitions of these contextual facts as well as those in the next table are given in the following paragraphs.

Component ID (column ID)	Description of object
44	A pair of left hand object minimum and maximum simultaneous cardinalities.
45	A pair of right hand object minimum and maximum simultaneous cardinalities.
76	A UID of the accuracy of a quantification.
70	A UID of a pick list for the qualification of aspects.
19	A UID of the validity context for a fact.
65	A partial definition in natural language of a concept or individual thing.
4	A full definition in natural language of a concept or individual thing.
42	A textual description of a main fact.

[3] In this document the unqualified term ‘object’ is used as synonym for the term ‘anything’.

14	Remarks on the expression of a main fact.
8	Approval status of the expression of a main fact.

Table 9, Prime contextual facts

The definitions of these components of an expression are given in the following paragraphs.

2.4.2.2 Secondary contextual facts

The secondary contextual facts are facts that do not directly contribute to the semantic interpretation of the facts, but are added for administrative reasons. They include the facts in the following table.

Component ID (column ID)	Description of object
24	Reason for latest change of status.
67	UID of the successor of the fact, in case the fact has the status 'replaced'.
11	UID of creator of fact.
9	Date-time of start of validity of the fact.
23	Date-time of start of availability of the expression.
22	Date-time of creation of copy.
10	Date-time of latest change of the expression.
6	UID of author of latest change of the expression.
78	UID of addressee of the expression.
13	References.
53	UID of the expression of the fact. (Line UID)
50	UID of a collection of facts to which the fact belongs.
0	A sequence in which the expressions are presented. (Presentation sequence)

Table 10, Secondary contextual facts

Uniqueness constraints are implementation constraints that intent to prevent that a database contains identical expressions in which also the contextual facts are identical. It depends on the scope of a database which expressions including context are considered to be identical. For example, in an extreme situation two identical expressions about the same fact, thus semantically having the same meaning, but expressed by different persons (originators), may be considered to be two different expressions in one context, whereas they are considered to be the same expression in another context. This means that it might be required to add the originator to the uniqueness constraint. Similarly, when a requirement is stated to be valid in multiple validity contexts, then this means that there are multiple requirements, each with its own 'fact UID'. This implies that the 'validity context UID' should be added to the second uniqueness constraints.

2.5 Naming relations for objects in expressions of facts

In principle, every UID that is used in an expression of a main fact, or in an expression of a contextual fact, is denoted in a human readable expression by a term (name, etc.), or by more than one term in case of synonyms. The terminology is recorded in naming relations between UIDs and terms.

In Databases all the naming relations of UIDs can be recorded in a separate Naming Table. However, it is also possible that they are included in an integrated Expression Table (see par. 4). In an integrated Expression Table the UIDs as well as the terms are included in the table itself.

Table 11 specifies all the names that imply naming relations (expressions of additional contextual facts) that are required to allocate names (terms) to the UIDs that are used to express main facts and contextual facts.' Note that 'name' stands for a character string that can be a term, a code, a phrase, a number, a URI, etc.

Component ID (column ID)	Description of object
101	The name of a left hand object.
201	The name of a right hand object.
3	The name of a relation type.
31	The name of an extent (typically a number)
7	The name of a scale (UoM).
54	The name of a language.
16	The name of a language community.
73	The name of a left hand role.
75	The name of a right hand role.
43	The name of an intention.
12	The name of an author of latest change.
77	The name of an accuracy of quantification.
20	The name of a pick list.
68	The name of a collection of facts.
79	The name of an addressee of the expression.
83	The name of a creator of a fact.

Table 11, Naming columns in an Expression Table

3. Subsets of expression components & context

Expressions in messages or databases may consist of the full set of expression component, facts and contextual facts as defined in this document. It may also consist of a subset of them.

The definition of these subsets implicitly also define subset Expression Tables.

Depending on the application, users may decide to use a flexible subset or one of the predefined standard subsets of the collection of contextual facts.

The following subsets of facts are defined, each with its equivalent subset Expression Table:

- Subset Minimum subset
- Subset Flexible subset
- Subset Nomenclature
- Subset Dictionary
- Subset Taxonomy
- Subset Product Model
- Subset Business Model (recommended)
- Query tables

These standard subsets are defined in the following paragraphs.

The subsets require the presence of all elements that are specified for the chosen subset and the elements shall be arranged in the indicated sequence, with as only exception the Flexible subset.

The default subset is the *Business Model*.

3.1 Subset: Minimum subset

A *Minimum subset* is intended to express messages in nearly natural language, whereas still using a formal taxonomic dictionary and the standard formal language relation types.

A Minimum subset has limited expression capabilities and therefore only suitable for usage in simple applications in small communities. For example, the subset does not provide a mechanism for the explicit distinction between homonyms nor for the explicit distinction between languages. It is neither suitable to express intentions such as questions, denials, but is only intended to express statements. It does not provide for contextual facts, such as an approval status, source and timing information about the expressed facts.

Users of Minimum subsets should ensure that the terms (names) of objects in the messages are unique or that the distinction between homonyms is apparent from the context in which the terms are used and that synonyms are explicitly declared to be synonyms.

A Minimum subset consists of only the core of an expression of a main fact, expressed in formalized natural language terms. Such a minimum subset consists of the following three expression components:

Component ID (column ID)	Description
3	A name of a relation type (= formal language phrase)
101	A name of a left hand object
201	A name of a right hand object

Table 12, Minimum subset

Minimum subsets may be expressed (implemented) in various ways (syntactic structures or formats). For example in the form of functions, such as:

Relation type (left hand object, right hand object)

Another implementation may be in the form of a Minimum subset Expression Table, which contains only the three columns: 101, 3 and 201. An example of such an Expression Table is:

101	**3**	**201**
Name of left hand object	**Name of relation type**	**Name of right hand object**
the Eiffel tower	is located in	Paris

Table 13, Minimum subset Expression Table

Minimum subset expressions are triples of expression components that are directly compliant with the Notation 3 RDF (N3) form of the RDF standard of the World Wide Web Consortium (W3C).

Note: A more elaborate Expression Table with additional columns can also be represented as collections of triples and can also be expressed in RDF or Notation 3 RDF as is described in the last chapter.

Minimum subset+ is a in extension of Minimum subset with the fact UID, which enables the management of facts.

3.2 Subset: Flexible subset

A **Flexible subset** is a subset that contains at least the non-optional expression components. The non-optional components are: 2, 101, 1, 60, 3, 15, 201, 8, 9 and 10 as described in Table 14.

Component ID (column ID)	Description
2	UID of left hand object
101	Name of left hand object
1	UID of main fact
60	UID of relation type
3	Name of relation type
15	UID of right hand object
201	Name of right hand object
8	Approval status
9	Date-Time of start of validity
10	Date-time of latest change
etc	Free choice of additional columns (in any sequence)

Table 14, Minimum expression components for flexible subset

Note: Expressions that consists of more than three expression components can be represented as collections of triples. For example, when they are expressed in RDF or Notation 3 RDF extended with an indicator for the collections (such as in TRIX). Such a format is described in ISO 15926-11.

The selection of additional optional columns as well as the sequence of the columns is free. The sequence of the columns in an Expression Table is semantically irrelevant, because the columns shall be uniquely identified by their column identifiers and the relations between the columns are defined independent of their position in the table.

A Flexible subset may even include non-standard additional columns, which columns are then treated as comment from a formal language perspective.

3.3 Subset: Nomenclature, Lexicon or Vocabulary

A *Nomenclature subset*, *Lexicon subset* or *Vocabulary subset* (Nomenclature for short) is intended to specify terminology. A specification of terminology implies names, synonyms, codes, abbreviations, translations, etc. that are used to denote something that is represented by a UID.

A Nomenclature subset represents a list of particular terms as ‘names’ of things and their unique identifier, together with the language in which the names are expressed and the language community in which the term for the thing originates.

A Nomenclature list typically includes names of concepts, but may also include names of individual things such as countries and other standard geographical objects. Organizations or projects will often maintain the nomenclature of individual things or collections of individual things. For example as represented in equipment lists, line lists, inventories, etc.

A Nomenclature subset includes contextual facts as well. For example the approval status and date-time values, sources, etc. A Nomenclature subset consists of the following expression components in the indicated sequence:
0, 69, 54, 71, 16, 2, 101, 1, 8, 67, 9, 10, 12 and 13. These expression components are given in Table 15.

Component ID (column ID)	Description
0	Presentation key
69	UID of natural language
54	Name of natural language
71	UID of language community
16	Name of language community
2	UID of left hand object
101	Name of left hand object
1	UID of main fact
8	Approval status
67	UID of succeeding fact
9	Date-Time of start of validity
10	Date-time of latest change
12	Name of author of latest change
13	UID of creator of fact

Table 15, Expression components for a vocabulary

A collection of such expression components require a syntactical structure to define the relations between the components. For example, a tabular implementation implicitly defines as contextual fact a naming relation between the UID and a term (name of thing) in the vocabulary. This relation is of the type ‘is called’ (or ‘is referenced as’). For example:

130206 is called pump.

Such a table also expresses a contextual fact that defines the language context in which the naming is done. This fact is of the type ‘is presented in’ (English).

The Nomenclature subset also allows defining the language community (sub-culture) where a name originates (component 71 and 16). For example, the name ‘pump’ may be declared to originate in the ‘mechanical engineering’ domain.

Misspellings and a pointer to the correct spelling can also be recorded in the nomenclature table. Misspellings can be indicated by a status (column 8) ‘replaced’ as well as an ‘identifier of successor of main fact’ (column 67), which refers to the Fact UID that defines the correct spelling.

Preferred terms are terms which use is preferred in a particular language community. When an organization wants to specify its own list of preferred terms it might specify them within their own language community, even specifying terms that are identical to terms that are already specified for another language community.

When a Nomenclature (or Lexicon or Vocabulary) is represented in tabular form it can be represented in a Nomenclature subset of an Expression Table. Table 16 is an example of the main columns in a Nomenclature table.

54	16	2	101	1	8	67
Language	**Language community (discipline)**	**Gellish UID**	**Name of thing**	**UID of fact**	**Status**	**UID of successor of main fact**
English	mechanical technology	130206	pump	201	accepted	
Deutsch	Maschinenbau	130206	Pumpe	202	proposed	
Nederlands	werktuigbouwkunde	130206	pompe	203	replaced	204
Nederlands	werktuigbouwkunde	130206	pomp	204	accepted	

Table 16, Nomenclature subset example

Table 16 illustrates that the same concept, represented in the formal language by UID 130206 is denoted in English as ‘pump’ and in other languages by different terms, whereas the spelling ‘pompe’ in Dutch is a misspelling that should be replaced by ‘pomp’.

3.4 Subset: Dictionary

A *Dictionary subset* is intended to provide textual definitions of things, especially of concepts, as an addition to the Nomenclature and Taxonomy subsets. This implies a relation between the thing and the text that defines the thing.

The following is an example of the core columns in a Dictionary subset of an Expression Table.

54	2	101	1	4	8
Language	**UID of defined thing**	**Name of thing**	**UID of fact**	**Textual definition**	**Status**
English	130206	pump	205	is a rotating equipment item intended to increase pressure in a liquid.	accepted
Nederlands	130206	pomp	206	is een apparaat met roterende delen dat bedoeld is om de druk in een vloeistof te verhogen.	accepted

Table 17, Dictionary subset core example.

A full *Dictionary subset* consists of a Vocabulary subset (Table 15) plus two additional components: the full definition and an option for adding remarks.

Component ID (column ID)	Description
4	Full definition (natural language text)
14	Remarks

Thus a Dictionary subset comprises the following components in the indicated sequence:
0, 69, 54, 71, 16, 2, 101, 1, **4**, **14**, 8, 67, 9, 10, 12 and 13.

Note 1: It is possible to record definitions for the same concept in multiple languages.

Note 2: *Definition models* are definitions that are expressed as collections of relations between concepts. Those relations require at least a Product Model subset.

Note 3: *Verbal (spoken) or pictorial definitions* require a relation to a sound or picture (or combination of them). However the textual definition (column 4) is meant for a string in ASCII or Unicode only. Therefore, such other definitions require at least a ‘Product model’ subset, as described below.

3.5 Subset: Taxonomy

A *Taxonomy subset* is a specialization hierarchy of concepts, also called a subtyping hierarchy (sometimes erroneously called a classification hierarchy). This implies that there are subtype-supertype relations between the concepts. A subtype concept is a specialization of a supertype concept. The inverse of that relation expresses the same fact in another way, namely that a supertype concept is a generalization of a subtype concept.

Table 18 illustrates the core columns in a Taxonomy table.

54	2	101	1	15	15	8
Language	**UID of left hand object**	**Name of left hand object**	**UID of fact**	**UID of right hand object**	**Name of right hand object**	**Status**
English	130206	pump	7	130227	rotating equipment item	accepted
Nederlands	130206	pomp	7	130227	apparaat met roterende delen	ignore duplicate

Table 18, Taxonomy subset example

A specialization relation implies that the subtype concept inherits all the aspects that are intrinsic to the supertype concept.

Note that the left hand object name and the right hand object name, as well as the language, are strictly speaking superfluous, but they are added to support the readability of the table. If they are ignored it becomes clear that the two lines in the above example define the same fact, which is the reason why the UIDs of the facts are identical and the status of the latter one is set at 'duplicate'.

A *Taxonomy subset* is an extension of a Dictionary subset by including expression components for the UIDs and names of supertype concepts.

Component ID (column ID)	Description
15	UID of right hand object
201	Name of right hand object

Thus a Taxonomy subset consists of the following expression components in the indicated sequence:

0, 69, 54, 71, 16, 2, 101, 1, **15**, **201**, 14, 8, 67, 9, 10, 12 and 13.

3.6 Subset: Product Model

A *Product Model subset* is intended for use in practice of data exchange to describe individual objects (including occurrences) during their lifecycle as well as knowledge about kinds of things.

A Product Model subset consists of the following expression components in the indicated sequence:

0, 69, 54, 71, 16, 2, **44**, 101, 1, **60, 3**, 15, **45**, 201, **65, 4, 30, 31, 66, 7**, 14, 8, 67, 9, 10, 12, 13, **50** and **68**.

The expression components are presented in Table 19.

Component ID (column ID)	Description
0	Presentation key
69	UID of natural language
54	Name of natural language
71	UID of language community
16	Name of language community
44	Left hand object cardinalities

2	UID of left hand object
101	Name of left hand object
1	UID of main fact
60	UID of relation type (kind of relation)
3	Name of relation type
45	Right hand object cardinalities
15	UID of right hand object
201	Name of right hand object
65	Partial definition
4	Full definition
30	UID of extent
31	Name of extent
66	UID of unit of measure
7	Name (symbol) of unit of measure (UoM)
14	Remarks
8	Approval status
67	UID of succeeding fact
9	Date-Time of start of validity
10	Date-time of latest change
12	Name of author of latest change
13	UID of creator of fact
50	UID of collection of facts
68	Name of collection of facts

Table 19, Expression components of a Product Model subset

For definitions of the components and implied relations see par. 5.

3.7 Subset: Business Model

A *Business Model subset* is intended for use in practice of data exchange to describe propositions. This includes business communication about both designs (imaginary objects) as well as real world objects (observed individual objects) during their lifecycle and about enquiries, answers, orders, confirmations, etc. This subset is a superset (indicated in **bold**) of the Product Model subset, so it can also be used for storage and exchange of knowledge about kinds of things.

A Business Model subset is a subset that consists of the following expression components in the indicated sequence:
0, 69, 54, 71, 16, **39**, 44, 2, 101, **72**, **73**, **5**, **43**, **19**, **18**, 1, **42**, 60, 3, **74**, **75**, 45, 15, 201, **34, 35,** 65, 4, 30, 31, **32, 33,** 66, 7, **76, 77, 70**, **20**, 14, 8, **24**, 67, 9, **23, 22**, 10, **11, 83, 6**, 12, **78, 79,** 13, **53**, 50, 68.

The expression components in a Business Model are presented in Table 20.

Component ID (column ID)	**Description**
0	Presentation key
69	UID of natural language
54	Name of natural language
71	UID of language community
16	Name of language community
39	Reality
44	Left hand object cardinalities
2	UID of left hand object
101	Name of left hand object
72	UID of left hand kind of role
73	Name of left hand kind of role

5	UID of intention
43	Name of intention
19	UID of validity context
18	Name of validity context
1	UID of main fact
42	Description of main fact
60	UID of relation type (kind of relation)
3	Name of relation type
74	UID of right hand kind of role
75	Name of right hand kind of role
45	Right hand object cardinalities
15	UID of right hand object
201	Name of right hand object
34	UID of exponent
35	Name of exponent
65	Partial definition
4	Full definition
30	UID of extent
31	Name of extent
32	UID of probability
33	Name of probability
66	UID of unit of measure
7	Name (symbol) of unit of measure (UoM)
76	UID of accuracy of quantification
77	Name of accuracy of quantification
70	UID of pick list
20	Name of pick list
14	Remarks
8	Approval status
24	Reason
67	UID of succeeding fact
9	Date-Time of start of validity
23	Date-time of start of availability of expression
22	Date-Time of creation of this copy of expression
10	Date-time of latest change
11	UID of creator of fact
83	Name of creator of fact
6	UID of author of latest change
12	Name of author of latest change
78	UID of addressee of expression
79	Name of addressee of expression
13	References
53	UID of expression
50	UID of collection of facts
68	Name of collection of facts
82	Name of file in which facts reside

Table 20, Expression components for a Business Model

The above-indicated sequences of expression components are defined as a handy sequence for human interpretation of a tabular content. There is no semantic meaning in that sequence, because the semantics of the relations between the components are defined explicitly in chapter 5.

3.8 Query subsets

A *Query subset* consists of one of the other subsets, extended with expression components for the specification of string commonality criteria.

In a tabular form a Query subset is a subset that is extended with the expression components 80. 81 and 84.

Component ID (column ID)	**Description**
80	Left hand string commonality
81	Right hand string commonality
84	Relation type string commonality

4. Implementation in a Universal Format (Syntax)

All semantic expressions, of any 'arity', can be expressed in various syntaxes. For example in RDF triples, although such triples should be extended with a method to recognize collections of triples, which are usually called 'graphs', to represent the idea's, terms as well as contextual facts. This can be done for example by using TRIX as is specified in ISO 15926-11. A more direct powerful and efficient implementation is the tabular Expression Format syntax. Such a table can be used to describe any facts and ideas as well as queries about individual things or occurrences, requirements for things or knowledge about things in general. The various standard kinds of relations that are used to classify the relations and the indication of the intentions and the contextual facts determine the categories of the expressions.

Typically a statement or question about an individual thing is modeled by a relation that is classified by a kind of relation (a relation type) that is denoted by a phrase that starts with "is" or "has". A requirement phrase starts with "shall" and must specify a validity context (in column 18). A statement that expresses knowledge typically uses a relation type that is denoted by a phrase that starts with "can have" or "can be". This is illustrated in Figure 1.

101	**18**	**1**	**3**	**45**	**201**
Left hand object name	**Validity context for main fact**	**UID of fact**	**Relation type name**	**Cardina lities**	**Right hand object name**
I-1		101	is a part of		P-1
impeller	handover to operations	102	shall have as aspect a		diameter
centrifugal pump		103	can have as part a	1,n	pump impeller
impeller		104	has by definition as part a	2,n	vane

Figure 1, Example of Product data, a Requirement and Knowledge in one Expression Table

The example in Figure 1 illustrates four kinds of statements. The first one states that a particular impeller is a part of a particular pump. The second one states that information about any (model of an) impeller that is handed over to operations shall include a diameter. The third statement describes the general knowledge that any centrifugal pump can have (and at least has) one impeller. The minimum and maximum number of simultaneous instances (individual impellers for individual pumps) is indicated by the cardinalities. The last expression states that an impeller has by definition 2 or more vanes. Figure 1 demonstrates that all such kinds of statements can be expressed in the same table or in tables that have the same columns and have a single common definition.

4.1 Universal Databases and Messages – Expression Format

Messages that are exchanged between systems will each consist of one or more expressions of facts (ideas). In order to support readability for verification and human communication they should also contain the names of the things that are referred to by the UIDs. This combination enables that different parties use their own terminology as synonyms, whereas a corresponding party can verify the terminology, while replacing it by his own terminology. Therefore, messages shall use Expression Tables, each of which is a combination of a Fact Table that uses UIDs and a Naming Table. This combination makes the table human readable and makes it suitable for the exchange of long as well as relative short messages, without the need to exchange and combine different tables.

Universal Semantic Databases should also use these integrated Expression Tables or their equivalents in another syntactic structure (format) in order to keep track of the variation in use of terminology.

The core of an Expression Table thus consists of a combination of the content of a Fact Table (Table 2) and Naming Table (Table 11). A partial combination is illustrated in Table 21.

54	43	1
Language	**Intention**	**UID of fact**
English	statement	201
English	statement	202

2	101	60	3	15	201	4
UID of left hand object	**Name of left hand object**	**UID of relation type**	**Name of relation type**	**UID of right hand object**	**Name of right hand object**	**Full definition**
101	P-1	1225	is classified as	102	cycle-pump	for office-1
102	cycle-pump	1146	is a specialization of	130206	pump	intended to inflate cycle tires.

Table 21, Example of an Expression Format table (selection of columns)

Databases can also consist of a combination of various Expression Format tables, each with the same structure, whereas such tables may be stored in a distributed way, thus forming distributed databases.

Expression Format tables allow that different naming conventions (synonyms) for the same objects are used in the same table or in a combination of Expression Format tables. This has the advantage that each organization can keep using its own terminology, provided that they use common UIDs while the dictionaries explicitly specify the synonyms.

When an organization does not allow the use of synonyms, then an Expression Format table may be considered to include redundancy, as it then (re)specifies (uses) the names of things multiple times. For such database implementations it is possible to eliminate the redundancy by implementing only Naming Tables and Fact Tables.

A data exchange message will contain one or more Expression Tables that are sent to another party as a file (a message, embedded in an 'envelope') in some chosen format. The format that may be chosen is not prescribed by the formal language, the only constraint is that the format represents the same meaning as the expressions in a tabular structure such as in Expression Format tables, without constraints on the characters in the cells, except for a tab, and that the receiving party possesses software to read such a format.

An Expression Format table is a neutral (software independent) tabular format that can be implemented in the structure of various proprietary or open file or database formats. For example, an Expression Format table for data exchange can be implemented as a neutral ASCII or Unicode text file (.txt) or may be implemented as a spreadsheet table (.xls), provided that single tabs separate the fields. A semantic database is preferably implemented as an ISO standard SQL database file format (as in Oracle or DB2), or as an MS-Access database table (.mdb), or as collections of RDF triples.

An Expression Format table can conform to various defined subsets of expression components, each with its own application area and corresponding number of columns. Those subsets are defined in chapter 3.

4.2 Expression Format table - definition

4.2.1 The Expression Format table header definition

Each Expression Format table file has in principle a table header that defines table columns (with defined and implied relations between the columns) and a body that contains rows with fields that represent cells for values of the expression components. Each row represents an expression of an idea or fact and accompanying contextual facts as described in chapter 2.
An Expression Format table can consist either of a complete set of columns (according to a combination of a Business Model subset and a Query subset) or of one of the pre-defined subsets of expression components as defined in chapter 3.

Each column has a column ID and a column name (which are the same as the expression component ID and name). A value in a column field is a value for the expression component. Most expression components imply a contextual fact, which means that the value has implied relations with one or more values in other columns. *Those relations define the facts about the objects!*

An Expression Format table body shall be preceded by header information, which consists of three collections of ‘fields’. Each collection of fields may be implemented on a separate line (row) at the top of a table (such as in a spreadsheet table), or may be included in a database definition.

1. The first collection of fields consists of a sequence of fields A1 through An. The first seven fields meanings are position based, the later ones are name based. The fields shall contain the following content:

   A1 = The table type, being ‘Expression Table’ or ‘Naming Table’ or ‘Fact Table’ or ‘Query Table’ or just ‘Gellish’. This specifies that it conforms to the Gellish Expression Format table header and column definitions.
   A2 = The language in which the table is expressed, although left hand terms and right hand terms may include terms in other languages. Thus e.g. ‘English’ or ‘Nederlands’ or any other name of a formalized natural language that is used for the terms (vocabulary) in the table as a whole.
   A3 = The character string ‘Version:’ or the equivalent term in the natural language indicated in field A2.
   A4 = The version of the formal language defining ontology (dictionary) that is required for the interpretation of the expressions in the file. (especially the version of the upper ontology section).
   A5 = The date of the release of the collection of expressions in this table (optional).
   A6 = The name of the collection of expressions in the table (optional), possibly preceded by a path to the location where the source of the table is located in a network (such as the Internet).
   For example: http://example.gellish.net/TOPini.
   A7 = Free text. Typically used for a name or description of the content of the table.
   A8 = LowerObjUID=n: The lower limit of the range within which (by default) new UIDs can be allocated to new objects within this set.
   A9 = UpperObjUID=n: The upper limit of the range within which (by default) new UIDs can be allocated to new objects within this set.
   A10= LowerRelUID=n: The lower limit of the range within which (by default) new UIDs can be allocated to new relations (ideas, facts) within this set.
   A11= UpperRelUID=n: The upper limit of the range within which (by default) new UIDs can be allocated to new relations (ideas, facts) within this set.
   A12= RefIRIs=(name$_1$[,name$_i$]): References (IRI’s) to zero or more Expression Format tables that are required to be used in combination with this table as a prerequisite for a proper interpretation.

2. The second collection of fields contains the sequence of expressions component ID’s (column ID’s), whereas each ID is a standard number that denotes a particular defined expressions component.
   Note that the ID numbers are arbitrarily chosen. These ID’s allow the expressions components (table columns or parameters) to be presented in a different sequence without loss of meaning (the numbers in the table below correspond to those expressions component ID’s).

3. The third collection contains human readable text for every expressions component (column field) in the second collection, providing (short) names of the table columns. These names are free text. They are typically expressed in the natural language that is indicated in field A2.

If an Expression Format table is implemented in a spreadsheet or CSV, ASCII or Unicode file, then the table starts with a header of three lines that represent the above three collections in the same sequence.
If an Expression Format is implemented in a parameter driven form, then the header consists of three header lines: header-1, header-2 and header-3, each with a set of parameter values conform the above three collections.

4.2.2 The Expression Format table columns

The lines in Expression Format tables are independent of each other and thus the lines may be sorted in any sequence, without loss of meaning (different sequences should be semantically identical). For example time dependency should be modeled explicitly and should not be inferred from line sequences.

Each line (row) in the body of an Expression Format table (which in a spreadsheet, ASCII or Unicode table starts on the fourth line) expresses a group of facts, which consists of a *main fact* and a number of *contextual facts*.

Depending on the type of main fact (the main relation and its relation type) slightly different contextual facts can be distinguished and thus slightly different conventions are used to fill in the fields in the corresponding table row as is indicated in the table below.

Several columns contain unique identifiers (UIDs). Each UID should preferably be represented by a 64-bit integer (8-byte, Int64 or bigint**),** whereas only positive values shall be used. It is not recommended to use an unsigned integer (which only allows positive values) because SQL only enables the bigint datatype, which is signed.

Most other columns contain character *string values*. For database implementations it is indicated whether they have a fixed or variable length (*nvarchar* of *varchar*) or whether the string is externally stored (data types *ntext* and *text*). In addition to that it is indicated whether the cells may contain Unicode.

Fields in columns that are indicated as optional may be left empty, in which case the indicated default value is applicable. Otherwise a field value is obligatory.

The data type, optionality and default value of each expression component (or table column in an Expression Format table) are specified in Table 22. Note: the Expression component numbers (Comp ids) correspond with the column IDs in an Expression Format table.

Comp id	Expression component name (name of table column)	Data type, Optionality, Default value
0	Presentation key (Sequence)	string (optional), default null
69	UID of natural language (LanguageUID)	integer (optional), 64 bit, default null
54	Name of language of left hand object (Language)	string (optional), Unicode, nvarchar(255), default null
71	UID of language community (UID-7) (LHContextUID)	integer (optional), 64 bit, default null
16	Name of language community (LHContextName)	string (optional), Unicode, nvarchar(255), default null
39	Reality (LHReality)	string (optional), Unicode, nvarchar(255), default null
2	UID of left hand object (UID-2) (LHObjectUID)	integer, 64 bit
44	Left hand object cardinalities (LHCardinalities)	string (optional), non-Unicode, varchar(32), default null
101	Name of left hand object (LHObjectName)	string, Unicode, nvarchar(255), default = ‘nameless’
72	UID of left hand kind of role (LHRoleUID)	integer (optional), 64 bit, default null
73	Name of left hand kind of role (LHRoleName)	string (optional), Unicode, nvarchar(255), default null
5	UID of an intention (IntentionUID)	integer (optional), 64 bit, default ‘491285’
43	Name of intention (Intention)	string (optional), non-Unicode, varchar(255), default ‘statement’

19	UID of validity context (ValContextUID)	integer (optional), 64 bit, default null
18	Name of validity context (ValContextName)	string (optional), Unicode, nvarchar(255), default null
1	UID of main fact (UID-1) (FactUID)	integer, 64 bit
42	Description of main fact (template text) (FactDescription)	string (optional), Unicode, nvarchar(255), default null
60	UID of relation type (RelTypeUID)	integer, 64 bit
3	Name of relation type (RelTypeName)	string, Unicode, nvarchar(255)
74	UID of right hand kind of role (RHRoleUID)	integer (optional), 64 bit, default null
75	Name of right hand kind of role (RHRoleName)	string (optional), Unicode, nvarchar(255), default null
15	UID of right hand object (UID-3) (RHObjectUID)	integer, 64 bit
45	Right hand object cardinalities (RHCardinalities)	string (optional), non-Unicode, varchar(32), default null
201	Name of right hand object (RHObjectName)	string, Unicode, nvarchar(255), default = ‘nameless’
65	Partial definition (PartialDefinition)	string (optional), Unicode, ntext, default null
4	Full definition (FullDefinition)	string (optional), Unicode, ntext, default null
30	UID of extent (ExtentUID)	integer, 64 bit
31	Name of extent (ExtentName)	string, Unicode, nvarchar(255)
66	UID of Unit of measure (UoMUID)	integer (optional), 64 bit, default null
7	Name of Unit of measure (UoM) (UoMName)	string (optional), Unicode, nvarchar(32), default null
76	UID of accuracy of quantification (AccuracyUID)	integer (optional), 64 bit, default null
77	Name of accuracy of quantification (AccuracyName)	string (optional), Unicode, nvarchar(255), default null
70	UID of pick list (DomainUID)	integer (optional), 64 bit, default null
20	Name of pick list (DomainName)	string (optional), Unicode, nvarchar(255), default null
14	Remarks (Remarks)	string (optional), Unicode, ntext, default null
8	Approval status of main fact (ApprovalStatus)	string, non-Unicode, varchar(64)
67	UID of succeeding fact (SuccessorUID)	integer (optional), 64 bit, default null
24	Reason (Reason)	string (optional), Unicode, ntext, default null
9	Date-time of start of validity (EffectiveFrom)	Date-time, stored as a real value in the ‘1900 date system’[4]
13	UID of creator of fact (CreatorUID)	integer (optional), 64 bit, default null
10	Date-time of latest change (end of validity) (LatestUpdate)	Date-time, stored as a real value in the ‘1900 date system’

[4] See http://support.microsoft.com/kb/q180162/

6	UID of author of latest change (AuthorUID)	integer (optional), 64 bit, default null
12	Name of author of latest change (Author)	string (optional), Unicode, nvarchar(64), default null
22	Date-time of creation of copy (CopyDate)	Date-time, stored as a real value in the ‘1900 date system’
23	Date-time of start of availability of expression (AvailabilityDate)	Date-time, stored as a real value in the ‘1900 date system’
78	UID of addressee of expression (AddresseeUID)	integer (optional), 64 bit, default null
79	Name of addressee of expression (AddresseeName)	string (optional), Unicode, nvarchar(64), default null
13	Reference (Reference)	string (optional), Unicode, nvarchar(255), default null
53	UID of expression (UID-5) (ExpressionUID)	integer (optional), 64 bit, default null
50	UID of collection of facts (UID-4) (CollectionUID)	integer (optional), 64 bit, default null
68	Name of collection of facts (CollectionName)	string (optional), Unicode, nvarchar(255), default null
80	Left hand string commonality (LHCommonality)	string (optional), non-Unicode, nvarchar(64), default null
81	Right hand string commonality (RHCommonality)	string (optional), non-Unicode, nvarchar(64), default null
82	File name	string (optional), Unicode, nvarchar(64), default null

Table 22, Expression components in Expression Format tables

5. Definitions of expression components and implied relations

5.1 Natural language

A UID of a natural language (69) is the unique identifier of the natural language in which the name of the left hand object (see column 101) and the name of the relation type (see column 3) is spelled and, if present, in which the definition (see column 63 and 4) is spelled. The language is a context for the origin of the referencing relation between the UID and the string that is the name.

A name of a natural language (54) of the left hand object name indicates the name of the language for which a UID is given in column 69 and that is a context for the name of the left hand object (see column 101) and the name of the relation type (see column 3). If the relation type name is not available in that language, it may be given in English.
The allowed values for 'language name' are the names defined in the Formal Dictionary (or a private extension). Currently the dictionary contains names of natural languages and of (artificial) programming languages.

For example:

- natural language has as qualitative subtype English, French (francais), German (Deutsch), etc.
  The language 'International' shall be used to indicate strings that are natural language independent, such as codes.

This column represents a fact that can be expressed as a relation between the UID of the language (69) and a name of that language (54).
For example:

910036	is called	English

5.2 Language community

The language community UID (71) provides the uniqueness context within which the left hand object *name* (101) is a unique reference to the object id in column 2, in addition to the language context (see component 69 and 54).

The context is superfluous (and is for human clarification only) on all lines other than lines with a specialization, a qualification a classification or an alias relation and their subtypes, because only there the left hand objects, identified by their UID, are *defined* to have a name. If no context is given on a definition line, then the name for the left hand object is unique in the whole (natural) language (column 54) and no homonyms are then allowed (in the Dictionary).

A name of a language community (16) for the associated name of the left hand object is a name for the uniqueness context of which the identifier is given in column 71.

The name is optional (and is for human clarification only) because the context UID in column 71 shall be a reference to a context that is defined on another line, where its UID and name appears in columns 2 and 101 respectively.

This column represents a fact that can be expressed as a relation between the UID of the language community (71) and a name of the community in which a term originates (16).

Example: 1193707	is called	engineering

5.3 Main fact

A UID of a main fact (1) is an identifier of the *main* fact that is represented on the line (such as an association or possession relationship). This main fact is of the type as indicated in column 3 'relation type name'.

A description of the main fact (42) is a character string that is a description of a main fact (1), which is meant to be presented to a user of an application system. The text is intended as an aid for human interpretation of the meaning of the main fact in its context and may imply an instruction to a user for what should be filled in (typically for filling in a value for the left and/or right hand term) or what

should be selected from a pick list in order to finalize a fact or group of facts. The text might appear on a user interface (e.g. a fill-in-the-blanks form or data sheet) and supports human understanding of the meaning of the fact(s) and the intention of the object in column 15 and 201 and optionally the UoM in column 7.
For example: the text 'temperature of the fluid at inlet' suggests that a value and a unit of measure should be supplied.

This column represents a fact that can be expressed by a relation between a UID of a fact (1) and its description (42). The fact is basically only used to provide a textual description as is specified for a user interface template.

Example: 201 is described as Length of the pipe

5.4 Validity context

The UID of a validity context (19) for a main fact identifies the context within which the fact id, given in column 1, represents a valid fact. If not given, the fact is valid in all contexts.

This column represents a fact that can be expressed by a relation between a main fact (1) and a UID of a validity context (19). The fact is basically only used to express in which context a requirement is applicable. Thus it is only applicable for ‘shall be...’ and ‘shall have...’ relations.

Example: 201 is valid in context 202 (e.g. ISO 16739)

The validity context name (18) provides a name of the context that is identified in column 19.

5.5 Intention

A UID of an intention (5) is the UID of which the name is specified in column 43.
An intention is the intention with which the expression of the main fact is communicated. It indicates the extent to which the main fact is the case according to the author of the proposition. An intention includes also a level of conviction about the truth of the possible fact. If a line expresses a proposition or communicative fact, then the intention qualifies the proposition. If a line expresses an opinion about a possible fact, then the intention indicates whether the expression of an idea is one of the following intentions:

- *statement,*
- *assertion* (declaration of truth),
- *denial,*
- *request (question),*
- *confirmation,*
- *promise,*
- *declination,*
- *denial,*
- *probability*
- *acceptance*

Default = ‘statement’, which means a qualification of the expression: this fact “is the case” according to the opinion of the author of latest change (see below).

A name of an intention (43) represents a fact that can be expressed as a relation between the UID of an intention (5) and a name of an intention with which an expression is communicated (43).

Example: 491285 is called statement

5.6 Reality

The reality (39) of left hand object is a classification of the left hand object, being either

- imaginary or
- real (= materialized)

This indicates that the object is either a product of a mind or an object whose existence is based in the physical world, either as natural or as artificial object.

If not specified, then the reality shall be interpreted from the context or from an explicit classification fact. Kinds of things (classes) are by definition imaginary. For example, a design activity of a pump will create an imaginary (although realistic) object; a fabrication process will create a real (observable) object. Note that an object cannot be imaginary and real. An installation relation or a materialization relation relates an imaginary object to a real object.

5.7 Left hand object

A UID of left hand object (2) is the identifier of the main object about which the line defines a fact. That main fact is an association between two objects mentioned in column 2 and 15. The external identifier (name) of the object in column 2 can be given in column 56 with its text attribute in column 101 'name of left hand object'.

A UID is an artificial sequence number, provided it is unique in a managed context. For example, the UID 4724 is a reference number of a telephone extension in the context of my company in The Hague. An identical number may refer to a different object in a different context, such as the extension with UID 4724 in the context of your company. The uniqueness context is given in column 16 (subject area). Such a context itself is defined on a separate line in an Expression Format table.

Note, that a fact represented by an association or relationship is also an object.

A name of a left hand object (101) is a string, which is a term such as a textual name, phrase, code, number, URI, etc. that denotes the object identified in column 2 and associated with it via an "is called" relation in a language context referred to in column 69 and a language community context referred to in column 71.

For example, a tag name or some other code or proper name or class name.

The string may also consist of a phrase consisting of multiple terms or a sequence of terms, each separated by a term separator, such as a comma or semicolon followed by a space. For example, a string may consist of a list of numeric values. It may also consist of a function name with a sequence of UIDs and terms as arguments in brackets.

The name is intended as a human reference or computer readable file name or address to denote the object represented by a UID in column 2. The name facilitates when the lines are sorted in a different sequence later. Normally the *name* has no UID (the object has one), but if the string would have a UID, then this name can be regarded an attribute of the encoded information identified in column 56.

Nameless objects are allowed, which implies that there is no instance in column 56 and the name in column 101 for the object in column 2 should be 'nameless'. Note, a nameless object (with a UID) can be uniquely referenced indirectly. For example it can be referenced by a combination of its kind and the assembly of which it is a part. For example, the impeller of P-1201 can be uniquely referenced as a nameless thing that is classified as an impeller and is a part of P-1201.

This column represents a fact that can be expressed as a naming relation between a UID of a left hand object (2) and a name of the left hand object (101).

Example:	301	is called	P-1

5.8 Left hand kind of role

A UID of left hand kind of role (72) identifies the kind of role that classifies the role that is played by the left hand object in column 2. This kind of role is implicitly a subtype of the first or second kind of role that is required by the kind of relation in column 60.

A name of left hand kind of role (73) is the name of the kind of role in column 72.

This column represents a fact that can be expressed as a relation between the UID of a left hand role (72) and a name of the kind of role (73).

Example:	401	is called	vessel assembly

5.9 Left hand cardinalities

For relations between kinds of things this column contains the *simultaneous cardinalities for the left hand object kind of thing*. This means that it indicates the minimum and maximum number of individual things of the specified kind that can or may be related at the same time with an individual thing of the kind specified as the right hand.
The cardinalities may be specified by:

- A comma separated list of two integers that indicate the lower and upper limit cardinalities. The upper limit may be the character ‘n’ to indicate that the upper limit is unlimited.

The table column represents a fact that can be expressed as a relation between the UID of the main fact (1) and the left hand cardinalities (44).

Example:	201	has as left hand cardinalities	1, n

5.10 Relation type (kind of relation)

A UID of a relation type (60) is unique ID for the kind that qualifies the fact in column 1, whereas a name of the type of relation is given in a formal language in column 3.

A relation type name or phrase (or fact type name) (3) is a name of one of a subtypes of relation or one of its base phrases or one of its inverse phrases. The preferred phrase is indicated by the naming context, such as ‘Gellish Formal English’ (see column 71 and 16).

5.11 Phrase type

A phrase type UID (85) is a language independent UID of a kind of phrase, being either base phrase (UID 6066) or inverse phrase (UID 1986) that represents the reading direction of the language. The UID 6066 indicates that a first role player UID and name is given in columns (2 and 101) and that a second role player is given in columns (15, 201), whereas UID 1986 indicates the inverse. Thus, when a base phrase is used then the left hand object (column 2, 101) shall be the object that is the player of the first role (role-1) according to the definition of the relation type. Then the right hand object (column 15, 201) shall be the player of the second role (role-2). When an inverse phrase is used, then the left hand object shall be the player of role-2 and the right hand object shall be the player of role-1. The fact that is expressed is independent of the phrase that is used, provided that the sequence of the objects is in line with the phrase.

This column represents a fact that can be expressed as a relation between the UID of the relation type (60) and a name (3).

Example:	1225	has as base phrase (6066)	is classified as a

5.12 Right hand kind of role

A UID of right hand kind of role (74) identifies the kind of role that classifies a role that is played by the right hand object in column 15. This kind of role is implicitly a subtype of the first or second kind of role that is required by the kind of relation in column 60.

A name of right hand kind of role (75) is the name of the kind of role in column 74. Typically the name of a kind of right hand role is a concatenation of the right hand name, the string “ of a “ and the left hand name. For example, in the expression “pump <has as part a> bearing”, the right hand role can be called “bearing of a pump”. Similarly, in the expression “pipe <has as aspect a> diameter”, the right hand role name can be called “diameter of a pipe”.

This column represents a fact that can be expressed as a relation between the UID of a right hand kind of role (74) and a name of the kind of role (75).

Example 1:	501	is called	vessel part
Example 2:	502	is called	member of A1

5.13 Right hand object

A UID of a right hand object (15) is the UID of the object that is related to the object in column 2. The name of this right hand object can (optionally) be given as right hand term in column 201. The name of an object that has a name is defined only on a line where the fact type indicates a referencing association to the object. On other lines a filled in name is only meant to support human readability.

For dates in column 15/201 the Gellish convention includes that the UIDs for dates between the year 1500 and 3000 are integer numbers that are concatenations of four digits for the year, two for the month and two for the day, whereas two zero's are used for the month when a whole year is meant and two zero's for the day when a whole month is meant. For example, January 2006 has UID 20060100.

For numbers the Gellish convention includes usage of the *formalized scientific notation* as is described in chapter **Fout! Verwijzingsbron niet gevonden.**, Appendix A of the book 'Semantic Information Modeling in Formalized Languages' [Ref. 1]. This convention implies that column 15 and 201 are used either for the whole number or for the integer significand only whereas in the latter case column 34 and 35 are optionally used for the corresponding exponent.

A name of a right hand object (201) is a character string, which is a term such as a textual name, phrase, code, number, URI, list of numbers separated by semicolons, etc. that denotes the object identified in column 15. It is associated with the object in column 2 that has a name in column 101.

For example, a tag name or some other code, numeric value, class name or a description that also is a name. A string that may appear in column 101 (left hand object name) may also appear in this column (201, right hand object name).

This column represents a fact that can be expressed as a naming relation between a UID of a right hand object (15) and a name of the right hand object (201). This fact in an expression requires a consistency check, because every (right hand) object shall appear (as a left hand object in a classification or specialization relation where is receives its name in the language and language community context (or is nameless).

Example:	302	is called	Length of P-1

5.14 Right hand cardinalities

Right hand object cardinalities (45) for relations between concepts are a pair of values, separated by a comma, that specify the *simultaneous* cardinalities for the right hand object concept. This means that it specifies the minimum and maximum number of individual things of that kind that can or may be associated with an individual thing of the left hand object kind at the same time. The cardinalities may be specified in the same way as the cardinalities for the left hand object.

The column represents a fact that can be expressed as a relation between the UID of the main fact (1) and the right hand cardinalities (45).

Example:	201	has as right hand cardinalities	1, n

5.15 Partial definition

A partial definition (65) of the left hand object is a description in natural language that together with the relation type name (column 3) and the right hand object name (column 201) forms a full definition of the left hand object as presented in column 4. A partial definition is only useful as an intermediate to generate a full definition in column 4.

This column represents a fact that can be expressed by a relation between a left hand object (2) and its partial textual definition (65) that complements a classification or specialization relation. The fact is basically only used to provide a textual definition on a line that specifies a classification or a specialization relation.

Example:	101	is partially defined as	that ...

5.16 Full definition

A full definition (4) of the left hand object is a textual description in natural language of the characteristics that define the left hand object or things of the kind specified by the left hand object concept. Typically this is a concatenation of three components: the term "is a" or "is an", the right hand object name and the text in column 65 (partial definition), separated by spaces. A full definition is only applicable on lines where the left hand object is an individual thing that is classified or a kind of thing that is defined to be a subtype of another kind of thing (i.e. with relation types <is classified as a> or <is a specialization of> or one of their subtypes.

This column represents a fact that can be expressed by a relation between a left hand object (2) and its full textual definition (4). The fact is basically only used to provide a textual definition on a line that expresses a classification or a specialization of the left hand object.

Example:	101	is defined as	a circular hollow profile that ...

5.17 Extent of being the case

A UID of an extent (30) specifies the extent to which the main fact (1) is the case. Typically in a relation between two individual things it specifies the fraction on a scale (e.g. expressed on a scale in weight percentage, %wt) of a part in a composition relation with a whole. Then the fraction is the fraction of the whole individual thing that forms the component individual thing. In a classification of a part relation it specifies a fraction or concentration of a component of a kind or a substance of a kind in a mixture for which a classification is the case. Then the fraction is the fraction of the whole that is classified by the kind.

A 'name' of an extent (31) is a denotation for the extent value, typically being a decimal encoded number.

The UID column represents a fact that can be expressed as a relation between the UID of the main fact and the UID of the extent.

Example:	Sample-1 of seawater	has as part	Sample-1 of salt	3.0	wt%
	Sample-1 of seawater	is classified by substance as	water	97	wt%

The extent can also be an upper or lower limit value for an extent. In such cases a subtype of the relation type should be used to express how the value should be interpreted. For example the relation types <has with a minimum ratio as part> (6089) and <is classified by substance as at least> (6085) indicate that the extent value is a lower limit value.

5.18 Probability of being the case

The UID of a probability of being the case (32) represents a qualitative value or a quantitative value on a scale that indicates the probability that the main fact in column 1 is the case. The character string that denotes the probability value can be expressed in column 33. If a probability is quantified then the scale or unit of measure (66) classifies the quantification of the probability as a value on that scale, except when the main fact is about a quantification relation. In that latter case a quantitative probability shall be in percentage. For example, the probability may specify a % chance that the main fact is true. A denotation ('name') of a probability (33) is typically a string in decimal notation.

For example, the specification that some state, called State-1, is open is 97 %. This can be expressed as follows:

State-1 <is qualified as> open 97 %

5.19 Unit of measure (UoM)

A UID of a unit of measure (66) identifies the scale used for interpretation of the numeric value of a property in column 201. In case column 201 contains a concept of property name, the indicated UoM UID in column 66 indicates the default.

A name of a unit of measure (7) is a name of the scale used for interpretation of the numeric value of a property in column 201. In case column 201 contains a concept of property name, the indicated UoM in column 7 is a name of the default.

This column represents a fact that can be expressed as a relation between the UID of the scale (66) and a name or symbol (7).

Example:	570423	is called	mm

5.20 Accuracy of a quantification

A UID of an accuracy of a quantification mapping (76) identifies a numeric range that is defined by two tolerances. The tolerances are defined relative to a common pivot value. The range defines the accuracy of the quantification mapping relation between a property and a numeric value on a scale, where the numeric value has a role as pivot value. For example, a range that is defined from –0.3 to +0.4 can be used to indicate the accuracy of a diameter. When the diameter maps on scale as equal to 5 mm with an accuracy that range around the pivot value (5), then this means that the diameter is between 4.7 mm and 5.4 mm.

The column represents a fact that can be expressed by a relation between a main fact (1) and a UID of an accuracy of mapping (76).

Example:	201	has as accuracy	590001

A name of an accuracy of quantification (77) is a name for the concept in column 76. For example, an accuracy of –0.3 and +0.5 around a pivot value.

This column represents a fact that can be expressed as a relation between the UID of an accuracy of quantification (76) and a name of a range that expresses the accuracy or tolerance (77).

Example:	590001	is called	-0.3, + 0.5

5.21 Picklist

The unique identifier for the collection of objects or pick list from which values for instances of the right hand term may be selected in the context of an instance of the left hand term. This holds within the validity context (if specified).

Note, this column (together with column 20) is meant as a short-cut for subtyping a (right hand) aspect type in the context of the left hand object and adding an additional line which defines that the value for a subtype “shall be one of the” pick list collection of aspect values.
For example,

model X	shall have a color from the list of	model X colors

is a short cut for:

model X	shall have a	color of model X
color of model X	is a specialization of	color

and

color of model X	shall be one of the	model X colors

The column represents a fact that can be expressed by a relation between a main fact (1) and a UID of a domain (70) that is a collection of qualitative aspects from which aspect values may be selected. The fact is only used to denote a list of qualitative values.

Example:	201	has as pick list	590002

The name of a pick list or domain identified by the Picklist UID in column 70. The name of the pick list shall be unique in the same context as the context for the right hand term (column 201) as defined in column 16 on the line where the right hand term is defined and occurs as a left hand term.

This column represents a fact that can be expressed as a relation between the UID of a pick list (70) and a name of a pick list (20).

Example:	590002	is called	Model X colors

5.22 Remarks

A remarks (14) component is intended for comments related to the fact or the existence of the left hand object, its definition or status.
Formally this is qualitative information that is a qualitative subtype of ‘remark’.

A remarks column represents a fact that can be expressed by a relation between a UID of a fact (1) and a Remark (14).

Example:	201	has as remark	To be checked with John

5.23 Approval status (Status)

An approval status (8) indicates the status of the expression of the main fact. The status of the other facts on a line can be derived from the status of the main fact. A status can be any of the qualitative subtypes of the concept ‘approval status’ in the Dictionary. For example: proposed, issue, deleted, proposed to be deleted, ignore, agreed, accepted, accepted association (= only the main fact is accepted), history, or replaced (see also the book ‘Semantic Information Modeling Methodology’). The status ‘replaced’ indicates that the main fact is deleted and that a succeeding fact (see column 64) exists. The reason of the status may be clarified in the remarks column (see column 14).

This column represents a fact that can be expressed by a relation between a UID of a fact (1) and an Approval Status (8).

Example:	201	has approval status	accepted

5.24 Succeeding fact

A UID of a succeeding fact (67) is a UID of the fact by which this line, and especially the main fact which UID is given in column 1, is replaced when the status in column 8 is "replaced". It indicates that there exists a succession relation between the two facts that points to the succeeding fact.

Note: If the relation type is the last classification relation or specialization relation for the left hand object, then the life of the left hand object is terminated and replaced by the left hand object of the succeeding relation.

This column represents a fact that can be expressed as a relation between a UID of a fact (1) and a UID of another fact (67). The fact specifies that the fact (1) is succeeded by the fact (67). When there is a succeeding UID mentioned, then the status of fact (1) should be ‘replaced’ and vice versa.

Example:	201	is succeeded by	301

5.25 Reason

A reason (24) describes the reason why an expression of a fact is created and changed and including the reason why the approval status became what it currently is. Typically a reason why a status became qualified as ‘deleted’ or ‘replaced’ or ‘historic’.

A reason column represents a fact that can be expressed as a relation between a UID of a fact (1) and a reason for latest change of status (24).

Example:	202	is changed for reason	duplicate of fact 201

5.26 Date-time of start of validity

A date-time of start of validity (9) is the moment at with the period of the validity of the main fact begins. It is implicitly associated with the main fact via a “valid since” relation. The ‘1900 date system’ enables very accurate timestamps, for example for the recording of moments of measurement (See http://support.microsoft.com/kb/q180162/). If no date-time value is given, it is assumed that the main fact has been valid always.

This column represents a fact that can be expressed by a relation between a UID of a fact (1) and a number that indicates a date and time (9) registered according to the 1900 time system that specifies when the fact became valid. This is either the time of measurement, when the measured value is recorded or it specifies an (approximate) date and time that the fact was invented, created or proposed for the first time.

Example: The fact that The Eiffel Tower is located in Paris is a fact that is created probably in the year 1887. This can be expressed as follows:

201	has as date of start of validity	1887

5.27 Creator of fact

A UID of a creator of a fact (11) specifies the person who invented, created or (first) proposed the fact. That person may differ from the one who created or changed the expression. The latter is the author of the latest change of the expression (6).

This column represents a fact that can be expressed as a relation between a UID of a fact (1) and a UID of a person who invented (or created or first proposed) the fact (11). The name of the creator can be recorded in column id (83).

Example: the fact that The Eiffel Tower is located in Paris is a fact that is created probably by Gustave Eiffel. The expression in a particular database may be created by John Doe, indicated by UID 123457 (see author of latest change (6)).

5.28 Date-time of latest change

A date-time of latest change (10) specifies the latest change of one of the contextual facts. If the status in column 8 is ‘deleted’, ‘replaced’ or ‘history’, then the date of latest change indicates the ***date-time of the end of the validity*** of the main fact. Then it is assumed to be related to the main fact by an ’is valid until’ relation.

This column represents a fact that can be expressed by a relation between a UID of an expression (53) and a number that represents a date and time registered according to the 1900 time system that specifies when the expression of the fact was modified for the last time. When no change of the expression took place yet, this date is the same as the date-time of start of validity.

Example:

222001	has as date of latest change	August 1, 2011

5.29 Author of latest change

A UID of the author of the latest change (6) is the party which name is given in column 12.

This fact is expressed by a relation between the UID of an expression (53) and the UID of a party (6) that is responsible for the changed version of the expression. When the expression is in its original unchanged state, then the UID of the author (party) shall be the same as the ‘UID of creator of fact’. This author (party) is taken as the issuer (sender) of the message line, making the statement, or asking the question, or commanding the command, etc.

Example:	222001	is changed by	401

A name of the author of a latest change (12) is a name of the person who is the originator of the proposition or of the expression of the fact and who has at least some responsibility for the content of the line; especially its latest change. It is good practice to provide this information, although strictly speaking it is optional. It is recommended that the allowed names are limited to unique references to persons of which the UID and additional information is in the database.

This column represents a fact that can be expressed as a relation between the UID of a party (6) and a name of a party that is author of the latest change (12).

Example:	491286	is called	John Doe

5.30 Date-time of creation of copy

A date-time of creation of copy (22) is a creation date-time that specifies when this copy of the expression was created. It is distinguished from the date of start of validity of fact. The latter specifies when a fact became the case, whereas this date-time specifies when this expression was recorded.

This column represents a fact that can be expressed as a relation between the UID of an expression (the line UID, 53) and a number that represents a data and time registered according to the 1900 time system, that specifies when this copy of the expression was created (22). It is distinguished from the date of start of validity of fact. The latter specifies when a fact became the case, whereas this date-time specifies when this expression was recorded.

Example, the fact that The Eiffel Tower is located in Paris is a fact that is valid since 1889, its date of start of validity of the fact. This fact may be recorded in a particular fact database e.g. at September 12, 2011, its date-time of creation of the (second) copy of the formal expression.

Example:

201	has as date of creation of copy	August 26, 2011

5.31 Date-time of start of availability of expression

This availability date-time (23) specifies when the expression of the fact was incorporated in a system for the first time. Other copies of the expression may be included in other data sets at later dates (see 22).

This fact is expressed as a relation between a UID of an expression (a Line UID, 53) and a number that represents a date-time (23) registered according to the 1900 time system that specifies when the expression of the fact was incorporated in a system for the first time. Other copies of the expression may be included in other data sets at later dates (see 22).

Example:

201	has as date of start of availability	August 26, 2011

5.32 Addressee of expression

A UID of an addressee of an expression (78) is the UID of the party that is the intended addressee for the expression. When the expression is not changed yet, the UID of the author (party) shall be the same as the ‘UID of creator of fact’. This author (party) is taken as the issuer (sender) of the message line, making the statement, or asking the question, or commanding the command, etc.

This fact is expressed by a relation between the UID of an expression (53) and the UID of a party (78) that is the intended addressee for the expression. When the expression is not changed yet, the UID of the author (party) shall be the same as the ‘UID of creator of fact’. This author (party) is taken as the issuer (sender) of the message line, making the statement, or asking the question, or commanding the command, etc.

Example: 222001 is addressed at 402

A name of the addressee of an expression (79) is the name of the party which UID is specified in column 78.

This column represents a fact that can be expressed as a relation between the UID of a party (78) and a name of a party that is the addressee of the expression (79).

Example: 491286 is called John Doe

5.33 References

References (13) are one or more names of organizations, persons or positions in organizations or (parts of) documents that act as a source or point of reference for the main fact. If more than one reference is provided the references are separated by semicolons. It is good practice to provide this information, although strictly speaking it is optional. It may include URI strings. It is recommended that the allowed names are limited to unique references to parties or documents, preferrably of which the UID and additional information is contained in the database.

This column represents a fact that can be expressed by a relation between a UID of a fact (1) and a character string (13) that describes one or more references that form supporting evidence for a fact or its description or expression.

Example: 201 has as reference ISO 1000

5.34 Complete expression

A UID of an expression (53) is the identifier for a single row in an Expression Format table. It indicates the collection of (contextual) facts (or ‘cloud’ of related things) in which the main fact and the contextual facts on one single line in an Expression Format table are included. It may be used to distinguish different expressions of the same fact (with the same fact UID (see column 1). For example, to distinguish the same fact expressed in different languages.

This column represents a fact that can be expressed as a relation between a UID of a fact (1) and a local UID of an expression (53), which expressions including the expression of the accompanying contextual facts.

Example: 201 is expressed by 222001

5.35 Collection of facts

A UID of a collection of facts (50) is a unique identifier of a collection of facts in which the fact as identified in column 1 is included. This column is intended to indicate a collection of which the elements are facts that are identified by the above mentioned unique main fact identifiers (UID-1). A plural fact identifier is typically used as an *identifier* of a *model* or *(sub) template* or *view*.
When a plural fact identifier is filled-in, it implies the existence of an inclusion relation (<is an element of>) between the main fact on this line identified in column number 1 and the collection of facts identified in column number 50. The name of the collection may be given in column 68. Collections may appear as left hand or right hand objects in main facts, for example in the definition of larger collections.

This column represents a fact that can be expressed as a relation between a UID of a fact (1) and a UID of a collection of facts (50) that specifies that the fact administratively belongs to the collection.

Example: 201 is an element of 601

A name of a collection of facts (68) specifies a name of the collection of main facts to which this fact belongs. This collection is identified by the UID in column 50. The facts in the collection might be managed together. The main fact on the line is an element of the collection. The collection may indicate for example an area of responsibility of a peer group, the content of a table or the facts on a data sheet.

This column represents a fact that can be expressed as a relation between the UID of a collection of facts (50) and a name of a collection of facts (68) of which the main fact is an element.

Example: 590003 is called Facts about buildings

5.36 Left hand string commonality

A left hand string commonality (80) specifies for a query in which way a search string has or shall have commonality with a target string that is a left hand object name.

A string commonality may have one of the allowed values that are specified as qualifications of a string commonality in the Dictionary. The allowed string commonality values are defined in the Dictionary. They include:

- case sensitive identical
- case insensitive identical
- case sensitive partially identical
- case insensitive partially identical
- case sensitive front end identical
- case insensitive front end identical
- case sensitive different
- case insensitive different
- equal
- unequal
- less than or equal
- great than or equal

The default in a query is ‘case sensitive partially identical’

This column represents a fact that can be expressed as a commonality relation between a name of a left hand object (101) (the search string) and the names of objects in Expression Format tables that are searched and that match the expression of the fact.

Example:
P has commonality case sensitive partially identical

5.37 Right hand string commonality

A right hand string commonality (81) specifies for a query in which way a search string has or shall have commonality with a target string that is a right hand object name. The allowed values are the same as for the left hand string commonalities.

This column represents a fact that can be expressed as a commonality relation between a name of a right hand object (201) (the search string) and the names of objects in expressions that are searched and that match the expression of the fact.

Example:

1	has commonality	case sensitive partially identical

5.38 Relation type string commonality

A relation type string commonality (84) specifies for a query in which way a search string has or shall have commonality with a target string that is a relation type name. The allowed values are the same as for the left hand string commonalities.

5.39 File name

A file name (82) specifies the name of the file from which the expression originates. A file name is recommended to include the date or date-time of the latest change of the file (or when not available the date-time of import of the file). For example, a CSV file called 'Model' with a latest modification date of 30 jan 2015 might be: Model 30jan2015.cvs.

Note: This component is only applicable for databases and not for data exchange message files.

5.40 Presentation key

A presentation key indicates a position or field in a presentation structure, such as a spreadsheet or a list of lines. It can support *sorting* the content of an Expression Format table. It has no contribution to the meaning of the facts represented on the line. The presentation key does not affect the meaning of the lines. This column can be arbitrarily filled-in for use in a specific context. Identical strings indicate that there is no preference in presentation sequence of the lines.

This fact is expressed as a relation between a UID of an expression (53) and a presentation sequence (0).

6. Implementation of formal languages

A Universal Semantic Database or Data Exchange Message consists of one or more expressions of main facts and their accompanying contextual facts. Those facts shall be stored in Gellish Expression Format tables (e.g. in SQL) or in one of their equivalent formats, such as collections of RDF triples. Each of those formats shall contain at least the obligatory facts (table columns) that are defined in this document and the definitions of those facts shall be compliant with the definitions in this document.

A database in which the content of several files with Gellish Expression Format tables are combined into one Gellish Expression Format table data store shall be extended with an additional column in which the file name is recorded from which an expression (a line) originates. The database application shall manage the addition of new data and the consistency of the various data sets as well as the consistency of data stores in a family of data stores.

6.1 Unique keys

Typically in databases the uniqueness of expressions is managed by determining a 'unique key'. That raises the question: when are expressions considered to be duplicates and what makes an expression unique?
Typically each main fact (a possible fact plus an intention) is unique and thus has a unique Fact UID. However, one and the same fact may be expressed in multiple languages. This means that in a multi-language Database it is allowed that the Fact UID is not unique, but then the combination of language UID, language community UID and fact UID is unique. Within the same database this may be equivalent with a unique Expression UID (line UID, 53). For particular kinds of applications, such as discussion forums where several people issue proposals for concepts, facts and their definitions, it may even be allowed that in one language multiple expressions about the same fact are allowed, provided that that status of the expression is 'accepted'. In such a database the author of latest change and the status shall be added to the unique key.

So, depending on the objective of a database the unique key may be:

1. Single language database: Fact UID
2. Multi-language database: language UID, language community UID and fact UID
3. Development database: for status 'accepted' the same as for multi-language database. For non-accepted status: language UID, language community UID, fact UID and author of latest change.

6.2 Distributed Semantic Databases

A Distributed Semantic Database consists of several data stores, whereas each data store is a Semantic Database.

Different data stores shall use the same formal language definition and shall use the same core of (main and contextual) fact definitions. In addition to that data stores may also use one or more of the optional contextual facts. Preferred collections of contextual facts are defined in chapter 3, Subsets, in which subsets of contextual facts are defined. When data stores are based on the same set of definition of kinds of contextual facts it enables that data from different data stores can be easily combined, merged or integrated, provided that the other conventions of the formal language are also adhered to. This also enables for example to combine the results of a query to various independent data stores, which then act as a distributed database.
This is illustrated in Figure 2.

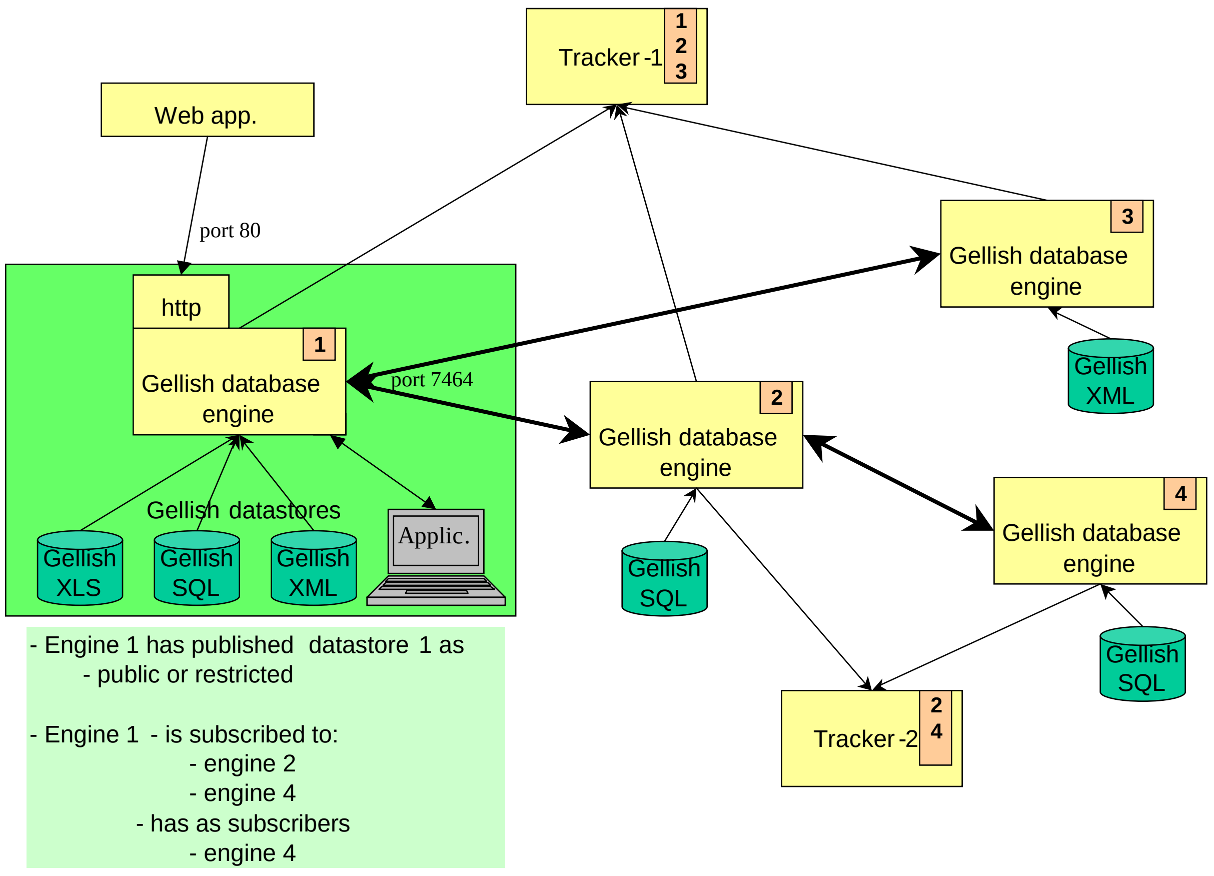

Figure 2, A Distributed Universal Semantic Database

The left hand of Figure 2 illustrates a database engine that manages three Data stores (in different formats) that can be read and updated via an Application Software system or via a web application (API). The engine can also communicate with other engines in the network to issues queries and to provide answers on queries from those other engines. Engines can be made known to engine trackers. An engine can publish a data store either as public or as restricted to subscribers. Other engines can subscribe to data stores. The engines exchange formal language messages, for example according to the SOAP protocol, which consists of a standard Header and a Message Body. The structure of the message bodies (queries as well as answers) shall include all required contextual facts that are defined in this document.

6.3 Formal languages in other syntaxes (formats)

A collection of formal expressions (expression of main facts and their contextual facts) can be implemented in several standard, system independent formats:

1. The standard *Gellish Expression Format table*, implemented in any SQL-based database tables (e.g. MySQL, MS-Access, Oracle or DB2), or in a spreadsheet table format, such as an MS-Excel table format (XLS).
2. The *Gellish Tab delimited ASCII file format*, or the *Gellish Unicode file format*, which are character string representations of a Gellish Expression Format table, where fields are separated by a tab as a delimiter and the characters are encoded either in ASCII code or in Unicode. One of the ways to create these table formats is by exporting a Gellish Expression Format table in MS-Excel format through saving the file in the ASCII or Unicode format.
3. The *Gellish STEPfile format*, an ISO 10303-21 implementation format of a Gellish Expression Format.
4. The *Gellish XML format*, an XML implementation format of a Gellish Expression Format, defined in an XML Schema (see http://gellikx.com/2009/ns/2.0/GellishSchema/).

5. The *Gellish RDF* (e.g. as defined in *ISO 1526-11) possible in combination with TRIX (as defined in ISO 15926-11) or in RDF(S)/OWL.*

The above-mentioned formats are semantically equivalent. In other words, the meaning of all five ways of expressions of facts is identical and defined unambiguously by the semantics of the formal language.
This separation between form and content definition gives a freedom to choose their preferred or the most appropriate format in their context. It also enables computer software to interpret and process a message content automatically and unambiguously in whatever form it is.

The Dictionary itself is documented using a Gellish Expression Format table, as it can be downloaded in the form of Microsoft Excel tables. The integration of the various tables forms one large Gellish Expression Format table (seen either as one or as a collection of tables), in which various subsets can be distinguished as collections of facts.

6.4 Software and support

Various suppliers may deliver formal languages enabled software. Such software should be able to process any data in a standard neutral format, as far as the software can store and retrieve such data. Such software forms a component in the Semantic Web. Such software can also be made to search for product data in a semantic database or to browse the Dictionary or knowledge that is expressed in a formal language. An example of such an application independent browser is the *Gellish Search Engine* (which can be acquired via www.gellish.net/), which supports the creation, import and export, validation and browsing of any data in a Gellish Expression Format table.

7. Gellish Expression Format implementations

7.1 A Gellish Expression Format table

A Gellish Expression Format table can be implemented directly in any tabular file format. For example it can be implemented in a spreadsheet or an SQL based database table, such as in XLS format of MS-Excel, in MDB format of MS-ACCESS or in an Oracle or DB2 database table. And as such it can be exchanged.

7.2 The Gellish STEPfile format

The Gellish STEPfile format is a way to express the content of a Gellish Expression Format in a form that is compliant with the STEP physical file standard (ISO 10303-21), also called a “part 21” file format. A file in this format is indicated by file extension ‘.G21’.
ISO 10303-21 requires that the entities that are instantiated in a STEP compliant file are defined in a data model, written in EXPRESS (ISO 10303-11). This is defined in the following paragraph.

7.2.1 Gellish Expression Format subset Product Model defined in EXPRESS

The *Gellish Expression Format subset Product Model* as defined in EXPRESS is presented in the 3rd column of Table 23.

		SCHEMA Gellish_Data_Table_subset_Product_Model;
		ENTITY gellish_fact;
0	Sequence	presentation_sequence_key: OPTIONAL string;
54	Language	language_name: string;
71	LHContextUID	context_UID_for_left_hand_object_name: OPTIONAL integer;
16	LHContextName	context_name_for_left_hand_name: OPTIONAL string;
2	LHObjectUID	left_hand_object_UID: integer;
44	LHCardinalities	left_hand_cardinalities: OPTIONAL LIST(2) of integer;
101	LHObjectName	left_hand_object_name: string;
1	FactUID	fact_UID: integer;
60	RelTypeUID	relation_type_UID: integer;
3	RelTypeName	relation_type_name: string;
15	RHObjectUID	right_hand_object_UID: integer;
45	RHCardinalities	right_hand_cardinalities: OPTIONAL LIST(2) of integer;
201	RHObjectName	right_hand_object_name: string;
65	PartialDefinition	definition: OPTIONAL string;
4	FullDefinition	full_definition: OPTIONAL string;
66	UoMUID	uom_UID: OPTIONAL integer;
7	UoMName	uom_name: OPTIONAL string;
14	Remarks	remarks: OPTIONAL string;
8	ApprovalStatus	status: string;
67	SuccessorUID	successor_of_fact_UID: OPTIONAL integer;
9	EffectiveFrom	date_of_creation: real;
10	LatestUpdate	date_of_latest_change: real;
12	Author	originator_of_change: OPTIONAL string;
13	Reference	source: OPTIONAL string;
50	CollectionUID	collection_of_facts_UID: OPTIONAL integer;
68	CollectionName	collection_of_facts_name: OPTIONAL string;
		UNIQUE
		ur1: fact_UID;
		ur2: left_hand_object_name, right_hand_object_name, relation_type_name;
		END_ENTITY;
		END_SCHEMA;

Table 23, The Gellish subset Product Model defined in EXPRESS

The first column in the figure refers to the column number in a Gellish Expression Format table. The second column provides standard column names for database implementations.

A row in a Gellish Expression Format table corresponds directly with an instance of this “gelish_fact” entity.

The following example is an illustration of the body of a G21 file in ISO standard format for subset Product Model. The fact expresses that P-101 is classified as a centrifugal pump.

```
#1   gellish_fact(,‘english’,,’project A’,10000001,,,’P-101’,11000001,’is classified as’,
                    130058,,,’centrifugal pump’,,,,,,’accepted’,,20Feb2003,20Feb2003,’AvR’,’AvR’,)
```

When this is represented in a Gellish Expression Format table, not showing the empty columns and the last four columns, it becomes:

54	16	2	101	1	3	15	201	8
Language	**Language community for left hand object name**	**UID of left hand object**	**Name of left hand object**	**UID of fact**	**Name of relation type**	**UID of right hand object**	**Name of right hand object**	**Status of fact**
English	project A	101	P-101	201	is classified as a	130058	centrifugal pump	accepted

7.2.2 Gellish Expression Format table subset Business Model data model

A *Gellish subset Business Model* of a Expression Format table, as defined in EXPRESS, is presented in the third column of Table 24.

ID	Column name	Long name, optionality, type and default (EXPRESS schema)
		SCHEMA Gellish_Message_Table_Format_subset_Business_Model;
		ENTITY gellish_fact;
0	Sequence	presentation_sequence_key: OPTIONAL string; default null;
69	LanguageUID	language_UID: OPTIONAL integer; default null;
54	Language	language_name: OPTIONAL string; Unicode; nvarchar(255); default null;
71	CommunityUID	community_UID_for_community_name: OPTIONAL integer; default null;
16	CommunityName	community_name_for_LHObject_name: OPTIONAL string; default null;
39	LHReality	reality_of_left_hand_object: OPTIONAL string; default null;
2	LHObjectUID	left_hand_object_UID: integer;
44	LHCardinalities	left_hand_cardinalities: OPTIONAL LIST(2) of integer; default null;
101	LHObjectName	left_hand_object_name: string; default nameless;
72	LHRoleUID	left_hand_role_UID: OPTIONAL integer; default null;
73	LHRoleName	left_hand_role_name: OPTIONAL string; default null;
5	IntensionUID	intention_UID: OPTIONAL integer; default null;
43	Intention	intention: OPTIONAL string; default null;
19	ValContextUID	validity_context_UID: integer; default null;
18	ValContextName	validity_context_name: string; default null;
1	FactUID	fact_UID: integer;
42	FactDescription	description_of_main_fact: OPTIONAL string; default null;
60	RelTypeUID	relation_type_UID: integer;
3	RelTypeName	relation_type_name: string;
74	RHRoleUID	right_hand_role_UID: OPTIONAL integer; default null;
75	RHRoleName	right_hand_role_name: OPTIONAL string; default null;
15	RHObjectUID	right_hand_object_UID: integer;
45	RHCardinalities	right_hand_cardinalities: OPTIONAL LIST(2) of integer; default null;
201	RHObjectName	right_hand_object_name: string; default nameless;
65	PartialDefinition	definition: OPTIONAL string; default null;
4	FullDefinition	full_definition: OPTIONAL string; default null;
66	UoMUID	uom_UID: OPTIONAL integer; default null;
7	UoMName	uom_name: OPTIONAL string; default null;
76	AccuracyUID	accuracy_UID: OPTIONAL integer; default null;
77	AccuracyName	accuracy_name: OPTIONAL string; default null;
70	DomainUID	domain_UID: OPTIONAL integer; default null;
20	DomainName	domain_name: OPTIONAL string; default null;
14	Remarks	remarks: OPTIONAL string; default null;
8	ApprovalStatus	status: string;
24	Reason	reason: string; default null;
67	SuccessorUID	successor_of_fact_UID: OPTIONAL integer; default null;
9	EffectiveFrom	date_of_creation: real;
10	LatestUpdate	date_of_latest_change: real;
6	AuthorUID	originatorUID: OPTIONAL integer; default null;
12	AuthorName	originator_of_change: OPTIONAL string; default null;
78	AddresseeUID	addresseeUID: OPTIONAL integer; default null;
79	AddresseeName	addressee: OPTIONAL string; default null;
13	Reference	source: OPTIONAL string; default null;
53	LineUID	line_UID: OPTIONAL integer; default null;
50	CollectionUID	collection_of_facts_UID: OPTIONAL integer; default null;
68	CollectionName	collection_of_facts_name: OPTIONAL string; default null;

		UNIQUE
		ur1: fact_UID, language_UID, intention, originator_of_change;
		ur2: left_hand_object_UID, right_hand_object_UID, relation_type_UID, language_UID, intention, originator_of_change;

		END_ENTITY;
		END_SCHEMA;

Table 24, The Gellish subset Business Model defined in EXPRESS

The second column provides standard column names for database implementations. Yellow marked columns indicate that those columns do not appear in a Gellish Fact Table.

The above example expressed as a STEP Physical File, compliant with GTF subset Business Model and ISO 10303-21 is as follows:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((),'2;1');
FILE_NAME('gellish_table_format_subset_business_model','2003-05-02T23:18:26',('B.J.H. de
Boer'),('TLO Holland Controls b.v.'),'EXPRESS Data Manager version 20020107',$,$);
FILE_SCHEMA(('GELLISH_TABLE_FORMAT_SUBSET_BUSINESS_MODEL'));
ENDSEC;

DATA;
#1= GELLISH_FACT($,'english',$,'project A',$,10000001,$,'P-101',$,11000001,'is classified
as',$,130058,$,'centrifugal pump',$,$,$,$,$,'accepted',$,300000.,300000.,'AvR','AvR',$);
ENDSEC;

END-ISO-10303-21;
```

7.2.3 Subset Extended Model data model

The *Gellish Expression Format subset Extended Model* in EXPRESS is presented in Table 25.

		SCHEMA Gellish_Data_Table_subset_Extended_Model;
		ENTITY extended_gellish_fact;
0	Sequence	presentation_sequence_key: OPTIONAL string;
69	LanguageUID	language_UID: OPTIONAL integer;
54	Language	language_name: string;
71	LHContextUID	context_UID_for_left_hand_object_name: OPTIONAL integer;
16	LHContextName	context_name_for_left_hand_name: OPTIONAL string;
17	LHUniqueContext	uniqueness_context_left_UID: OPTIONAL string;
50	PluralFactUID	plural_fact_UID: OPTIONAL integer;
38	LHObjectType	left_hand_object_type: OPTIONAL string;
39	LHReality	reality_of_left_hand_object: OPTIONAL string;
2	LHObjectUID	left_hand_object_UID: integer;
56	LHTermUID	left_hand_term_UID: OPTIONAL integer;
44	LHCardinalities	left_hand_cardinalities: OPTIONAL LIST(2) of integer;
101	LHObjectName	left_hand_object_name: string;
43	Intention	intention: string;
19	ValContextUID	validity_context_UID: integer;
18	ValContextName	validity_context_name: string;
1	FactUID	fact_UID: integer;
60	RelTypeUID	relation_type_UID: OPTIONAL integer;
3	RelTypeName	relation_type_name: string;
52	RHUniqueContext	uniqueness_context_right_UID: OPTIONAL string;
15	RHObjectUID	right_hand_object_UID: integer;
45	RHCardinalities	right_hand_cardinalities: OPTIONAL LIST(2) of integer;
55	RHUnContextName	uniqueness_context_right_name: OPTIONAL string;
42	FactDescription	description_of_main_fact: OPTIONAL string;
201	RHObjectName	right_hand_object_name: string;
65	PartialDefinition	definition: OPTIONAL string;
4	FullDefinition	full_definition: OPTIONAL string;
66	UoMUID	uom_UID: OPTIONAL integer;
7	UoMName	uom_name: OPTIONAL string;
70	DomainUID	domain_UID: OPTIONAL integer;
20	DomainName	domain_name: OPTIONAL string;
14	Remarks	remarks: OPTIONAL string;
8	ApprovalStatus	status: string;

67
9
10
12
13
53
50
68
	SuccessorUID
	EffectiveFrom
	LatestUpdate
	Author
	Reference
	LineUID
	CollectionUID
	CollectionName
		successor_of_fact_UID: OPTIONAL integer;
		date_of_creation: real;
		date_of_latest_change: real;
		originator_of_change: OPTIONAL string;
		source: OPTIONAL string;
		line_UID: OPTIONAL integer;
		collection_of_facts_UID: OPTIONAL integer;
		collection_of_facts_name: OPTIONAL string;
		
		UNIQUE
		  ur1: fact_UID, language_UID, intention;
		  ur2: left_hand_object_name, right_hand_object_name, relation_type_name, language_UID, intention, originator_of_change, date_of_creation;
		END_ENTITY;
		END_SCHEMA;

Table 25, The Gellish subset Extended Model defined in EXPRESS

The second column provides standard column names for database implementations.

The second uniqueness rule expresses that a person can express a proposition more than once. If somebody expresses the same proposition twice (e.g. on different moments), then these expressions are considered to be different propositions (with different fact_UID).

7.3 The Gellish XML format (GXL)

An XML representation of a Gellish Expression Format may according to Ref [2], but may also be conform ISO 10303-28. The latter means that it is defined as a representation of a Gellish data model in EXPRESS as defined in the previous paragraph, although presented in XML, compliant with the conversion rules defined in ISO 10303 part 28.

An automated conversion procedure can converts a Gellish Expression Format implemented as an Excel spreadsheet table (XLS) into an XML file.

8. References

1 Andries van Renssen,: Semantic Information Modeling in Formalized Languages, Gellish.net, ISBN 978-1-304-51359-5, http://www.lulu.com/commerce/index.php?fBuyContent=14146523

2 http://www.gellish.net

3 http://gellikx.com/2009/ns/2.0/GellishSchema/

9. Index