34 Chemical Reactions

Reaction data elements can be included in any document of any document type that allows it. Common document scenarios where reactions occur may be

- Laboratory notebook entry
- Research article
- Reaction database entry
- Patent disclosure
- Manufacturing planning and specification

Specific such use cases would define specific document types with their own document code and constraints as to the sections and other data elements.

This implementation guide section defines how the SPL data elements are used to represent chemical reactions and related information in general from a general chemical and logical perspective, it does not specify a lot of constraints and conformance criteria for specific limited use cases beyond what is logically sensible. For example, if an organization was to use reaction SPL to catalog reactions they would likely require some minimum data elements and on the other hand limit the expressivity to only the features that that organization is interested in.

34.1 Recapitulating Documents and Sections

For general guide about the use of the document and section XML structure, see the implementation guide Section 2. The following is an example of the start of a document that carries reactions:

```
<document xmlns="urn:hl7-org:v3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="urn:hl7-org:v3
                    https://www.accessdata.fda.gov/spl/schema/spl.xsd">
  <id root="6ef20731-e18a-4f16-b952-6e0bc95f5931"/>
  <code code="C47902" codeSystem="1.3.6.1.4.1.32366.1.1"
        displayName="Article"/>
  <title>A New Method of Synthesizing Gold from Mercury</title>
  <effectiveTime value="20211016220549"/>
  <setId root="c5b792f0-f5b0-47e8-b4eb-c2a4bddcbc38"/>
  <versionNumber value="1"/>
```

Documents have a document (version) set id and an integer version number, and a document (version) id. The set id is the id of a set of document versions, the document id or document version id changes every time a new document version is created. These ids are UUIDs (also called GUIDs) and are presented in lower case (not upper case and not case-insensitive).

The document effective time is generally the date (and time) this document becomes “effective”, this is usuall the time it is finalized and released. This uses the timestamp

(TS) data type which is in the form “YYYYMMDD[HH[MM[SS[.FFF*]]]”, with at least the precision of day (“DD”) and optional hour (“HH”), minute (“MM”), second (“SS”) and if desired any fractional seconds. This is a constrained version of the ISO date and time format without any delimites like dashes, colons and the “T” between date and time.

The example continues with author, of which there may be one or more, which might be a person and the primary organization to which they are affiliated in their authorship, or only the organization if it specifications or database entries published where the individual authors are not disclosed:

```
<author>
  <assignedEntity>
    <assignedPerson>
      <name>Jean Baptista van Helmont</name>
    </assignedPerson>
    <representedOrganization>
      <name>University of Leuven</name>
    </representedOrganization>
  </assignedEntity>
</author>
```

Then comes the body of the document. A document has one body, which contains the toplevel sections:

```
<component>
  <structuredBody>
    <component>
      <section>
        <id root="211b8a92-efd3-4c6c-8622-46082d921b7d"/>
        <code code="C0600678" codeSystem="1.3.6.1.4.1.32366.1.1"
              displayName="Abstract"/>
        <title>Abstract</title>
        <text>Lorem ipsum …</text>
        <effectiveTime value="20211016220549"/>
```

There can be any nesting of sections down to any level.

```
      <section>
        <id root="211b8a92-efd3-4c6c-8622-46082d921b7d"/>
        <code code="C0600678" codeSystem="1.3.6.1.4.1.32366.1.1"
              displayName="Abstract"/>
        <title>Abstract</title>
        <text>Lorem ipsum …</text>
        <effectiveTime value="20211016220549"/>

        <component>
          <section>
            <id root="3451b8a92-efd3-4c6c-8622-46082d921b7d"/>
            <title>Introduction</title>
            <text>Lorem ipsum …</text>
```

Section codes are optional. If a document template with standardized sections exist, their codes can be used. If the sub-section type is unspecified, then no code element needs to be included.

Any time there is a heading, there is a section, and the heading becomes the title and the body becomes the text and optional further nested sub-sections.

34.2 Basic Chemical Reaction Structure

A chemical reaction structure can be placed under any section:

```
<section>
  <id .../>
  <code .../>
  <title>...</title>
  <text>...</text>
  <effectiveTime value="..."/>
  <subject2>
    <specification>
      <code nullFlavor="NI"/>
      <component>
        <processStep>
```

<subject2> is simply the tag for the subject of the section. There are different kinds of subjects, and chemical reactions would appear as "<subject2>". Notice that chemical substances (see implementation guide Section 14) are under a tag names <subject>, without the "2" at the end. There is no other special meaning of this "2" and there is no <subject1> nor a <subject3> tag.

<specification> exist because of the context in which this schema was first developed, i.e., as detailed manufacturing processes under a detailed specification of a manufactured compound. It turns out that this structure is wholly suitable to describe reaction schemas as it is to describe reaction processed reduced into practice and elaborated further. At this point, just cut through this specification layer, and provide the code with this nullFlavor="NI" as simply hard coded structure. The <component> of the <specification> is the reaction, represented by the tag <processStep>.

<processStep> is what represents any reaction. It is called process step from it's origin in manufacturing specifications, and any process step that changes any molecules is ultimately a chemical reaction. There is a continuum between stating a reaction schema and stating a very specific process with specific vessels, reactors, reagents, temperature, pressure, kinetic characteristics, equilibrium constants, reaction enthalpies, etc. All of that can be expressed in as much detail as required, and it is all done under this <processStep> tag.

The tag is called process step, because any process can be thought of as one step, or it can be decomposed into multiple steps. It could have just be called <process> but process step emphasizes this composability (and de-composability). See more about this in the subsection 34.4 Multi-Step further below.

34.3 Reaction Participants – Interactors

A chemical reaction equation notation contains inputs (reactants), outputs (products), and catalysts and solvents and reagents that remain unchanged in the reaction. Here is an example of two reactants and one product:

```
<processStep>
  <interactor typeCode="CSM">
    <functionCode code="reactant" codeSystem="1.3.6.1.4.1.32366.1.1"/>
    <identifiedSubstance>
      ... substance definition elements molfile, InChI, etc. ...
    </identifiedSubstance>
  </interactor>
  <interactor typeCode="CSM">
    <functionCode code="reactant" codeSystem="1.3.6.1.4.1.32366.1.1"/>
    ...
  </interactor>
  <interactor typeCode="PRD">
    <functionCode code="product" codeSystem="1.3.6.1.4.1.32366.1.1"/>
    ...
  </interactor>
</processStep>
```

Reactants and reagents are specified with interactor participations elements of typeCode “consumable” (CSM). A functionCode can further say what some people call “role” in the reaction, such as “substrate” vs. “other reactant”.

Products are specified with the interactor participations elements of typeCode “product” (PRD). A functionCode can label the main intended product vs. waste products (if they are even specified to balance the reaction.)

Any other agents in the reactions (that are typically written above or below the arrow, including catalysts and solvents, are specified with the interactor participations elements of typeCode “catalyst” (CAT), even if, in the case of a solvent, we would not consider that a “catalyst” (although, generally speaking, it is). Again, the functionCode can be used with domain specific terminology to say “catalyst” in the narrower sense vs. “solvent”, or any other more specific designation. In practical terms, one can think of typeCode values as:

- CSM - left side of arrow
- PRD - right side of arrow
- CAT - above and below the arrow
- DIR - intermediate structures

The classCode “DIR” stands for “direct participant”, i.e., a thing that is directly physically involved in some way, but in this case not specified whether it is input or product or catalyst.)

Now the <identifiedSubstance> elements can be specified fully using the substance specification in the SPL implementation guide Section 14 on “Substance Indexing”.

This allows capturing a huge range of substances from small molecules all the way to complex hybrid bio-macromolecules. Here, for example, is pyruvate as a reactant:

```
<processStep>
  <interactor typeCode="CSM">
    <identifiedSubstance>
      <id extension="PYR" root="6ef20731-e18a-4f16-b952-6e0bc95f5931"/>
      <identifiedSubstance>
        <code code="PYR"
              codeSystem="6ef20731-e18a-4f16-b952-6e0bc95f5931"/>
        <name>pyruvate</name>
        <moiety>
          <partMoiety/>

          <subjectOf>
            <characteristic>
              <code code="C103240" displayName="Chemical Structure"
                    codeSystem="2.16.840.1.113883.3.26.1.1"/>
              <value xsi:type="ED"
                     mediaType="application/x-mdl-molfile"><![CDATA[
  GS-rxn2rspl

  6  5  0  0  0  0           999 V2000
    4.5981   -0.2500    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.7320    0.2500    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.8660   -0.2500    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.7320    1.2500    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    2.0000    0.2500    0.0000 O   0  5  0  0  0  0  0  0  0  0  0  0
    2.8660   -1.2500    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  2  0  0  0  0
  3  5  1  0  0  0  0
  3  6  2  0  0  0  0
M  CHG  1   5  -1
M  END
]]></value>
            </characteristic>
          </subjectOf>

          <subjectOf>
            <characteristic>
              <code code="C103240" displayName="Chemical Structure"
                    codeSystem="2.16.840.1.113883.3.26.1.1" />
              <value xsi:type="ED"
                     mediaType="application/x-inchi"
                >InChI=1S/C3H4O3/c1-2(4)3(5)6/h1H3,(H,5,6)/p-1</value>
            </characteristic>
          </subjectOf>
```

See the substance indexing specification for details. But in the above example we can see molfiles and InChI both being used for specifying the pyruvate.

Since writing out the substances fully defined with all details is costly, one can write them out once and refer to them by a simple document-internal identifier scheme:

```
    <identifiedSubstance>
      <id extension="PYR" root="Document ID"/>
      <identifiedSubstance>
        <code code="PYR" codeSystem="Document ID"/>
        <name>pyruvate</name>
```

Here we have defined the symbol “PYR” in the local “namespace” identified by the present document’s id. And this has occurred twice. This seems to be a redundancy, as redundant indeed as having two tags <identifiedSubstance> nested one inside the other. The reason for this is in the underlying generic data model, in which Act objects (here, reaction) have multiple participants (here, the interactors), which link not directly to an entity (here, the substance entity, represented by the inner <identifiedSubstance> tag), but through a Role, which in turn connects a player and a scoper, but might only have a player.

The best example for Act – Participation – Role – (player + scoper) Entity is seen around the <author> tag of the <document>. The <document> is conceptualized as an Act (yes, because it is not the physical embodiment of a copy of that document which defines the document, but the information content, or communication which it conveys. A document is therefore an Act, also known as “Speech Act”). The <author> is a Participation of the <document> Act (other participations might be <verifier> or <legalAuthenticator>). Then that Participation links the Act with a Role, here <assignedEntity> which has both an <assignedPerson> player of the Role, and a <representedOrganization> scoper of that Role. The meaning of <assignedEntity> is any association, employee, contractor, or agent relationship of the player Entity (here, an individual Person) with a scoper (here, an Organization).

Other examples of this Act – Participation – Role – (player + scoper) Entity is the transportation (Act) of a payload (Participation) which is a chemical (player) contained in (Role) a vessel (scoper). This pattern of the general model underlies all nesting of the tags. And when in any particular case the scoper of a Role is not of special interest, therefore omitted, it can appear as if the Entity (here <identifiedSubstance>) is unnecessarily wrapped inside a Role (here also named <identifiedSubstance>). The reason those two are named the same thing is a coincidence of the type of Role being an “identified entity” with an identifying scoper and an identified player Entity called “Substance”. The Role player and scoper tag-names are formed based on the Role classCode (here IDENT) and it’s standard naming patterns, that includes the prefix “identified” before the name of the Entity class, “Substance” leading to “itentifiedSubstance”.

Finally the duplication of the “PYR” symbol as an id of the Role as well as a “code” of the Substance, is handy, because in all future references to the PYR participant, we can simply include only the outer <identifiedSubstance> tag with the <id> as follows:

```
<processStep>
  ...
  <interactor typeCode="PRD">
    <identifiedSubstance>
      <id extension="PYR" root="6ef20731-e18a-4f16-b952-6e0bc95f5931"/>
    </identifiedSubstance>
  </interactor>
```

That “PYR” is also defined as a Substance <code> is for a different purpose, where we may want to refer to the Substance Entity and connect it to a different Role, such as the generalization – specialization (is-a) Role or a part – whole Role.

In summary, the reacting substances can be specified in full detail, but also labeled and then re-used by their label.

This might suggest that document authors, and possibly those who define more constrained reaction document templates, might suggest that all substances used in the document shall be specified in separate sections, a materials list, and the reaction or process description sections shall refer to these substancees only by their labels given in those sections. There is no need to first define the substance labels and then use them, but "forward references" are completely fine.

For example, a reasonable document collection for a manual of standard practical reaction to obtain all sorts of substances, might use a standard style guide, where it is stipulated that the first section shall describe the product in detail, then a second section the source materials, and then the reaction and process description, only followed in the end, by a "Reagents, Solvents and Catalyst" section.

Another use of reaction documents might decree that common substances may be referred to by some well defined code system. Such as the FDA's UNII code system, or the Enzyme Commission EC numbers, or the Chemical Entities of Biological Interest (ChEBI) identifiers, or any other which the community exchanging those documents deem a well-known code system. In that case the identifiedSubstances might be given only by reference. For example, here we do not need to specify the structure of pyruvic acid, but we refer to the FDA substance registration system's (and substance indexing SPL file) with the UNII code "8558G7RUTR":

```
<identifiedSubstance>
  <id extension="PYR" root="Document ID"/>
  <identifiedSubstance>
    <code code="8558G7RUTR" codeSystem="2.16.840.1.113883.4.9"/>
    <name>pyruvic acid</name>
```

We are still also giving the label "PYR", so this is a good example to show why this apparent "duplication" of this outer <identifiedSubstance> Role <id> and the inner <identifiedSubstance> Entity <code> is sometimes (and perhaps very often) quite useful.

There are even cases where the complete structure of a substance is not known yet, such as in Orphan Enzymes Project [orphanenzymes.org] where the structure of the hydroxypyruvate decarboxylase (EC 4.1.1.40) is listed as unknown. To describe the hydroxypyruvate decarboxylase reaction in reaction-SPL, we therefore need to refer to the catalyst only by EC number and giving a local label:

```
<identifiedSubstance>
  <id extension="PYR-OH-deCOOH" root="Document ID"/>
  <identifiedSubstance>
    <code code="4.1.1.40" codeSystem="1.3.6.1.4.1.32366.1.1.24"/>
    <name>hydroxypyruvate decarboxylase</name>
```

34.4 Multi-Step Reactions

From the basic reaction building blocks we can build complex multi-step reactions. This can often make sense in the case of multi-step reaction schemas, but is even more relevant when these reaction schemas are reduced into practice and detailed instructions of how to carry out this reaction are provided.

The following example is a 3 step process whereby the second step has multiple sub-steps that occur in parallel:

```
<processStep>
  ...
  <component>
    <sequenceNumber value="1"/>
    <processStep .../>
  </component>
  <component>
    <sequenceNumber value="2"/>
    <processStep>
      ...
      <component>
        <sequenceNumber value="1"/>
        <processStep .../>
      </component>
      <component>
        <sequenceNumber value="1"/>
        <processStep .../>
      </component>
    </processStep>
  </component>
  <component>
    <sequenceNumber value="1"/>
    <processStep .../>
  </component>
</processStep>
```

When the sequenceNumber is increasing, that means the sub-processes are executed in that sequence. When two sequence numbers are equal, that means they may be performed in parallel.

There are many additional features to control process steps in the HL7 Reference Information Model underlying the SPL schema, including repeated execution of a sequence of steps (e.g., wash and rinse cycles) as well as conditionals and controls of how parallel processes get spawned and joined together. All these features are already defined and can easily be brought into the SPL schema either by future versions of SPL ratified through the HL7 process, or by a fork of the present SPL schema maintained by the community interested in utilizing these functions. Fortunately, for a specification of multi-step reactions schemas this enhancement of the SPL schema is not necessary.

34.5 Process Step Types and Control Variables

To specify laboratory process steps not only input and output substances and reagents need to be specified but also the actions performed on them. And then actions may have different parameters, such as temperature, pressure, and other conditions and settings. The following example specifies stirring while maintaining the temperature of the system between 100 °C and 110 °C:

```
<processStep>
  <code code="stir" codeSystem="1.3.6.1.4.1.32366.1.1.997"
displayName="stir"/>
  <text>stir at 100 to 110℃ for 4 hours</text>
  <effectiveTime>
    <width value="4" unit="h"/>
  </effectiveTime>
  <controlVariable>
    <observation>
      <code code="temperature" codeSystem="1.3.6.1.4.1.32366.1.1.998"
            displayName="temperature"/>
      <value xsi:type="IVL_PQ">
        <low value="100" unit="Cel"/>
        <high value="110" unit="Cel"/>
      </value>
    </observation>
  </controlVariable>
</processStep>
```

These activity and system codes have not all been specified, but a code system OID has been set aside where these codes are specified symbolically. Implementers should make a list of English words that concisely define the action, and this list can be compiled and provided with definitions, and then harmonized should there be any conflicts occurring.

Table 17: Laboratory Process Steps (gleaned from XDL)

Code	Description	Participations	Parameters	Control Variables
transfer	Move contents from one vessel into another	subject source vessel destination vessel	effectiveTime duration (fast, slowly)	
stir	Agitate a dispersion	subject vessel or system	effectiveTime duration	temperature
heatchill	Actively change the temperature by heating or chilling	subject vessel or system		temperature
dissolve	Disperse a material into a solvent	dissolved material solvent	effectiveTime duration	temperature
clean-vessel		subject vessel or system		
precipitate	Cause precipitation by optionally adding a reagent, then changing temperature and stirring.	subject vessel or system		

crystallize	Crystallize dissolved solid by ramping temperature to given temp over given time.		
purge	Purge liquid by bubbling gas through it.		
evacuate-refill	Evacuate vessel and refill with inert gas.		
filter	Filter a dispersion to separate undissolved particles from liquid		
wash-solid	Wash solid by adding solvent and filtering.		
dry	Allow remaining liquid to evaporate from solid		
separate	Separate a dispersion by mixing two different solvents, typically polar/aqueous and aliphatic (e.g. chloroform) and then allow to separate in two phases, the product will end up in one of the two phases.		
evaporate	Allow or promote solvent to evaporate bringing dissolved substance into solid phase.		temperature pressure (negative)
lyophilize	Freeze dry under negative pressure.		
distill	Evaporate a liquid phase with lower evaporation temperature from other liquid with higher evaporation temperature.		
irradiate	Expose to radiation.		dosis, wavelength