I2B2 Cell Messaging

Natural Language Processing (NLP) Cell (1.1)

1. Table of Contents

2. Document Version History

Date	Version	Description	Author(s)
11/26/2007	1.1	Version 1.1	Sergey Goryachev

3. Introduction

This document gives an overview of I2B2 cell messaging as well as a more detailed description of message formats specific to the Natural Language Processing (NLP) Cell.

3.1 The I2B2 Hive

Informatics for Integrating Biology and the Bedside (I2B2) is one of the sponsored initiatives of the NIH Roadmap National Centers for Biomedical Computing (http://www.bisti.nih.gov/ncbc/). One goal of I2B2 is to produce a comprehensive set of software tools to enable clinical investigators to collect and manage their project-related research data, including clinical and genomic data; that is, a software suite for the modern clinical research chart. Since different applications from different sources must be able to communicate with each other, a distributed computing model is needed, one that integrates multiple web-based applications in a standardized way.

The I2B2 hive and associated web services are the infrastructure used to create this integration. The hive is comprised of a collection of cells representing unique functional units. Cells in the hive have an array of roles, such as data storage, data analysis, ontology or identity management, natural language processing, and data conversion, derivation or de-identification. Each cell is a self-contained modular application that communicates with other cells via XML web services. A common I2B2 messaging protocol has been defined to enable the cells to interact with each other, sharing business logic, processes and data.

3.2 I2B2 Messaging Overview

All cells in the I2B2 hive communicate using standard, pre-defined I2B2 XML request and response messages.

A request message is sent from a client to a service and contains information, inside the top-level <request> tag, that allows the service to satisfy the request. The <request> tag contains a <message_header>, <request_header> and <message_body> as shown, below, in Figure 1.

The service sends back a response message, inside a top-level <response> tag, which informs the client about the status of the request and may also contain the actual results. The <response> tag contains its own <message_header>, <response_header> and <message_body> and it may optionally echo the request's <request_header> as shown, below, in Figure 1.

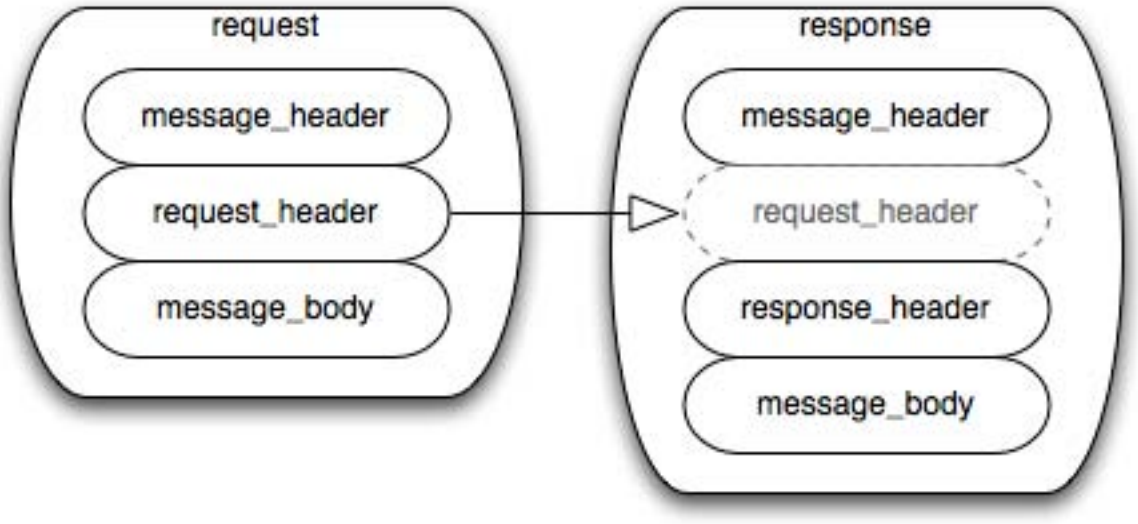


Figure 1: The basic structure of a request and response message. The request_header in the request can be echoed in the response.

3.2.1 Message Header

All requests are sent using a <request> tag and responses are returned using a <response> tag. The same <message_header> tag is used for both. Both request and response messages contain this <message_header> tag which has control information such as sending application, receiving application and message type.

3.2.2 Request Header

The request must contain a <request_header> tag which includes information about how to process a request such as the amount of time it is willing to wait for a response. The <request_header> tag may optionally be echoed back in the response.

3.2.3 Response Header

The response must include a <response_header> tag which includes general information about the response such as status and error messages or where to look for the results if they are not included with the response.

3.2.4 Message Body

Both request and response messages contain a <message_body> tag which may contain any well-formed xml. Individual cells may define cell-specific XML that will be put inside <message_body> tag. This cell-specific XML need not extend the I2B2 message schema since the I2B2 schema will allow insertion of tags from any namespace into the <message_body> tag.

3.3 I2B2 XML Schema Definitions

The I2B2 XML schema consists of three XSD files:

- **i2b2.xsd** This schema defines the type for the <message_header> and <message_body> tags. This schema is included in the i2b2_request.xsd and the i2b2_response.xsd.
- **i2b2_request.xsd** This schema defines the type for the top-level <request> tag and the <request_header> tag. It is used for validating i2b2 request messages.
- **i2b2_response.xsd** This schema defines the type for the top-level <response> tag and the <response_header> tag. It is used for validating i2b2 response messages.

More details about the <request>, <response>, <message_header>, <request_header> and <response_header> tags can be found in a separate document describing the generic I2B2 message. The remainder of this document describes the contents of the <message_body> for the Natural Language Processing (NLP) Cell.

4. The Natural Language Processing (NLP) Cell Messaging Details

The Natural Language Processing cell provides accepts the unstructured text repots from its clients and returns the extracted clinical information.

The cell currently supports three main types of natural language processing operations.

1. Principal diagnoses extraction. The client passes the unstructured text document to the cell and the cell returns the list of extracted principal diagnosis codes. In addition to concept codes, an extra information about each concept is returned: the UMLS CUI and UMLS concept preferred name, the list of semantic types to which the concept belongs, the term in the document that mapped to the concept, the list of modifiers (if any) that were assigned to concept (negation, temporal and family history), and the name and categories of document section where the concept was found.

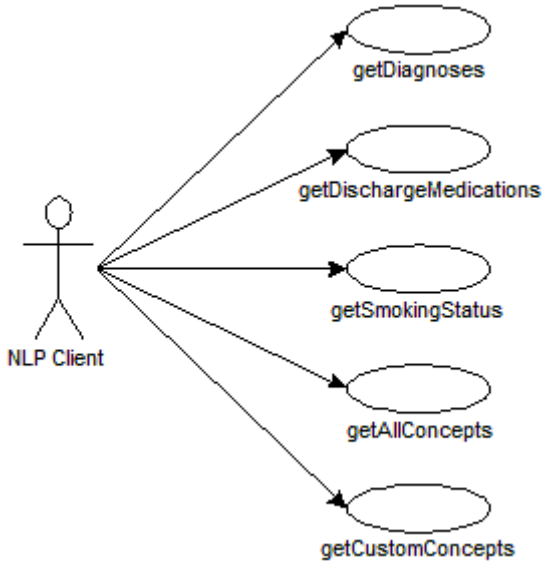
2. Discharge medications extraction. The client passes the unstructured text document to the cell and the cell returns the list of extracted discharge medication codes. In addition to concept codes, extra information about each concept is returned: the name and categories of document section where the concept was found.

3. Smoking status extraction. The client passes the unstructured text document to the cell and the cell returns the list of extracted smoking status codes. In addition to concept codes, extra information about each concept is returned: the name and categories of document section where the concept was found.

The cell also supports two additional operations. First, it supports the **extraction of all available concepts** from a document (i.e., principal diagnoses, discharge medications and smoking status) in one service call. This operation is available for convenience to simplify client programming. Second, the cell supports **custom concepts extraction**. The NLP cell communicates with the NLP core running on the server side. NLP core is configured to run a few standard NLP processing pipelines to support each of the available client operations (1, 2 and 3). Each pipeline consists of NLP components. Each component has its own setup parameters. There are 3 pre-assembled processing pipelines for the 3 standard operations (principal diagnoses extraction, discharge medications extractions and smoking status extraction). However, power users are given an option to (a) change the order of components in a pipeline and (b) change the setup parameters for each component in a pipeline. A custom pipeline allows user to solve new NLP problems such as finding co-morbidities, searching for arbitrary regular expression matches in the document, etc. For example, principal diagnoses extraction is actually a specific case of more general UMLS concepts extraction. Strictly speaking, a principal diagnosis is a UMLS concept found in the document section categorized as “Principal diagnoses related”, that belongs to at least one of semantic types designating medical finding. If we change the section category to “secondary diagnoses related”, we’ll be effectively searching for co-morbidities. Similarly, the discharge medication extraction is a specific case of regular expression matching and can be tweaked to extract other kinds of concepts.

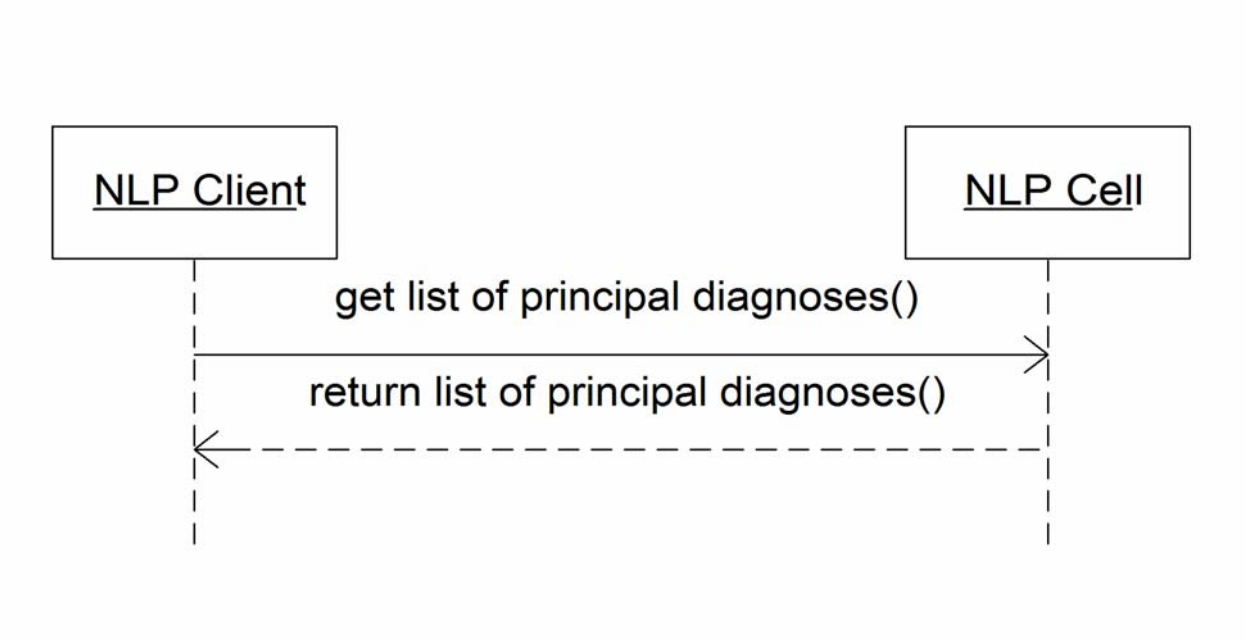
Below is a collection of operations (use cases) supported by the NLP service:

1. **`getDiagnoses`**: invokes the principal diagnoses extraction functionality, as described above.
2. **`getDischargeMedications`**: invokes the discharge medications extraction functionality, as described above.
3. **`getSmokingStatus`**: invokes the smoking status extraction functionality, as described above.
4. **`getAllConcepts`**: invokes all known concepts extraction functionality, as described above.
5. **`getCustomConcepts`**: invokes custom concepts extraction functionality, as described above.



4.1 getDiagnoses

A getDiagnoses message returns a list of principal diagnoses discovered in a text document. The text document body and the type of the text document (e.g., LMR note, BWH or MGH discharge summary) are passed in the message_body by the NLP client.



4.1.1 Get a list of principal diagnoses associated with a document

The getDiagnoses message is sent by the NLP client to retrieve the list of principal diagnoses associated with the text document. Each diagnosis is represented by a code. The sequence of events is as follows:

1. Client requests the list of principal diagnoses by sending a getDiagnoses message containing text report body and the type of the report.
2. The NLP service performs the following steps:
    a. The text report body is extracted from the message_body.
    b. The type of the report is extracted from the message. Currently, three report types are known: LCS-I2B2:c1010c (MGH Discharge Summary), LCS-I2B2:c1011c (BWH Discharge Summary) and LCS-I2B2:c1009c (LMR Note).
    c. The correct NLP pipeline is identified based on the type of the report (LMR note or discharge summary) and the type of the message (getDiagnoses). The principal diagnoses extraction pipelines, one for each report type, are available on the server.
    d. The report is parsed by the pipeline and principal diagnosis codes are extracted.
    e. The response message is generated. Along with a principal diagnosis code, some extra information about each principal diagnosis is included in the message_body of the response: the UMLS CUI and UMLS concept preferred name, the list of semantic types to which the concept belongs, the term in the document that mapped to the concept, the list of modifiers (if any) that were assigned to concept (negation, temporal and family history), and the name and categories of document section where the concept was found.
3. The client displays the list of principal diagnoses in tabular format. If no diagnoses were found (empty list was returned), a warning message is displayed.

4.1.2 getDiagnoses request message

```
<message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation
           download_date="2007-11-26T15:54:21.252-05:00"
           import_date="2007-11-26T15:54:21.252-05:00"
           sourcesystem_cd="RPDR"
           update_date="2007-11-26T15:54:21.252-05:00">
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>LCS-I2B2:c1010c</concept_cd>
          <start_date>2007-11-26T15:54:21.252-05:00</start_date>
          <observation_blob>
                ***Report body goes here***
           </observation_blob>
        </observation>
      </ns2:observation_set>
    </ns2:patient_data>
  </message_body>
```

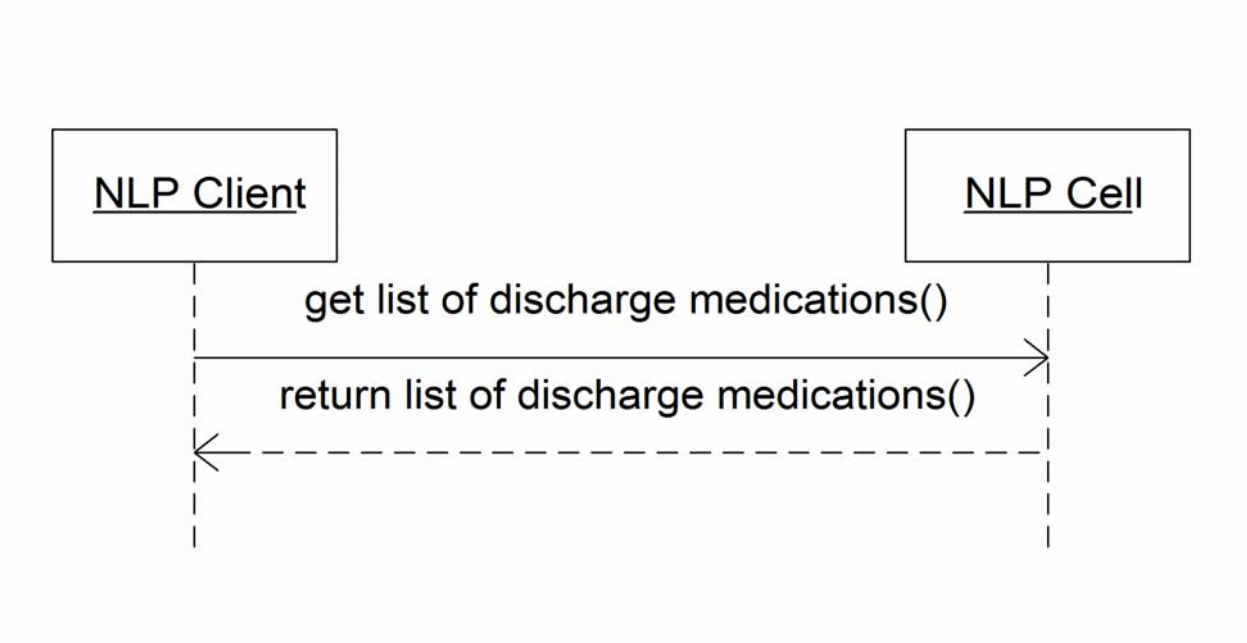
4.1.3 getDiagnoses response message

```
  <message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:c0349790</concept_cd>
          <start_date>2007-11-26T14:53:12.649-05:00</start_date>
          <observation_blob>
            <concept_cd type="umlsConcept">
              <cui>c0349790</cui>
              <umlsName>Exacerbation of asthma (disorder)</umlsName>
              <semanticTypes>
                <semanticType name="Finding" tui="T033" />
              </semanticTypes>
              <mappedTerm><![CDATA[Asthma exacerbation]]></mappedTerm>
              <modifiers>
                <modifier name="negationStatus" value="Actual" />
              </modifiers>
              <section>
```

```
                <sectionName><![CDATA[PRINCIPAL   DISCHARGE   DIAGNOSIS
;Responsible After Study for Causing Admission)]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="DIS" />
                  <sectionCateg name="PRI" />
                </sectionCategs>
              </section>
            </concept_cd>
          </observation_blob>
        </observation>
      </ns2:observation_set>
    </ns2:patient_data>
  </message_body>
```

4.2 getDischargeMedications

A getDischargeMedications message returns a list of discharge medications discovered in a text document. The text document body and the type of the text document (e.g., LMR note, BWH or MGH discharge summary) are passed in the message_body by the NLP client.



4.2.1 Get a list of discharge medications associated with a document

The getDischargeMedications message is sent by the NLP client to retrieve the list of discharge medications associated with the text document. Each discharge medication is represented by a code. The sequence of events is as follows:

1. Client requests the list of discharge medications by sending a getDischargeMedications message containing text report body and the type of the report.

2. The NLP service performs the following steps:
    a. The text report body is extracted from the message_body.
    b. The type of the report is extracted from the message. Currently, three report types are known: LCS-I2B2:c1010c (MGH Discharge Summary), LCS-I2B2:c1011c (BWH Discharge Summary) and LCS-I2B2:c1009c (LMR Note).
    c. The correct NLP pipeline is identified based on the type of the report (LMR note or discharge summary) and the type of the message (getDischargeMedications). The discharge medications extraction pipelines, one for each report type, are available on the server.
    d. The report is parsed by the pipeline and discharge medication codes are extracted.
    e. The response message is generated. Along with a discharge medication code, some extra information about each discharge medication is included in the message_body of the response: the name and categories of document section where the concept was found.

The client displays the list of discharge medications in tabular format. If no discharge medications were found (empty list was returned), a warning message is displayed.

4.2.2 getDischargeMedications request message

Same as `getDiagnoses`.

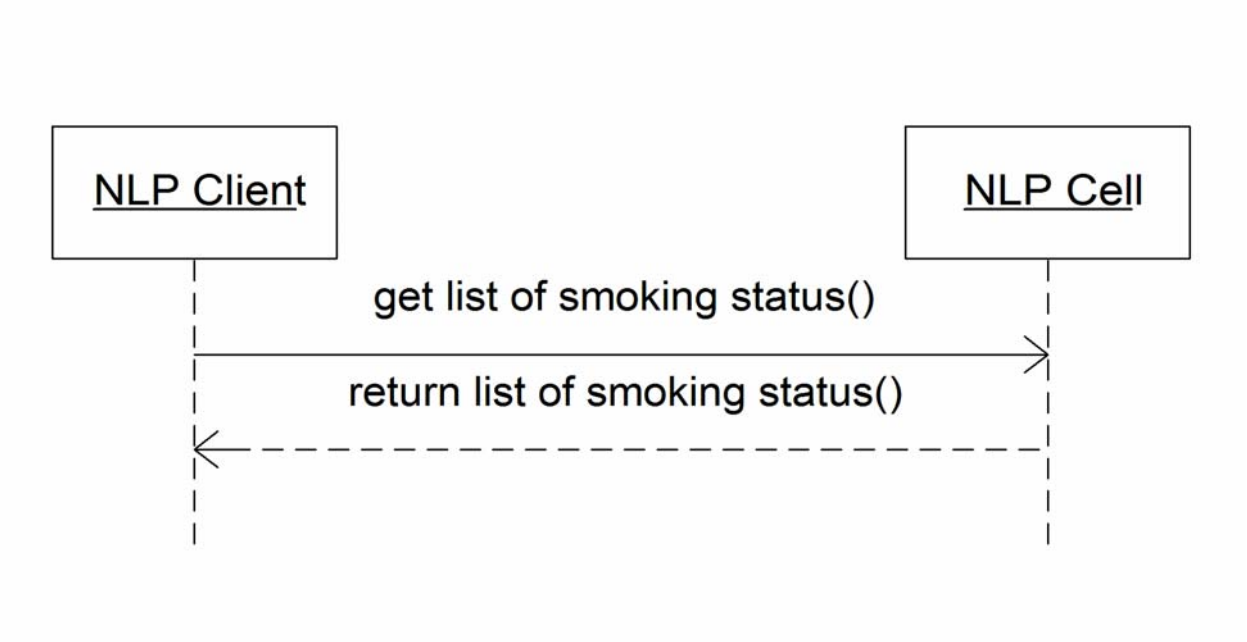
4.2.3 getDischargeMedications response message

```
  <message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:PREDNISONE</concept_cd>
          <start_date>2007-11-26T14:52:18.592-05:00</start_date>
          <observation_blob>
            <concept_cd type="medication">
              <code>DSG-NLP:PREDNISONE</code>
              <section>
                <sectionName><![CDATA[DISCHARGE
MEDICATIONS:]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="DIS" />
                  <sectionCateg name="MED" />
                </sectionCategs>
              </section>
            </concept_cd>
```

```
        </observation_blob>
      </observation>
    </ns2:observation_set>
  </ns2:patient_data>
</message_body>
```

4.3 getSmokingStatus

A getSmokingStatus message returns a list of smoking status codes discovered in a text document. The text document body and the type of the text document (e.g., LMR note, BWH or MGH discharge summary) are passed in the message_body by the NLP client.



4.3.1 Get a list of smoking status codes associated with a document

The getSmokingStatus message is sent by the NLP client to retrieve the list of smoking status codes associated with the text document. The sequence of events is as follows:

1. Client requests the list of smoking status codes by sending a getSmokingStatus message containing text report body and the type of the report.
2. The NLP service performs the following steps:
    a. The text report body is extracted from the message_body.
    b. The type of the report is extracted from the message. Currently, three report types are known: LCS-I2B2:c1010c (MGH Discharge Summary), LCS-I2B2:c1011c (BWH Discharge Summary) and LCS-I2B2:c1009c (LMR Note).
    c. The correct NLP pipeline is identified based on the type of the report (LMR note or discharge summary) and the type of the message (getSmokingStatus). The smoking status extraction pipelines, one for each report type, are available on the server.

d. The report is parsed by the pipeline and discharge smoking status codes are extracted.
e. The response message is generated. Along with a smoking status code, some extra information about each smoking status is included in the message_body of the response: the name and categories of document section where the concept was found.

The client displays the list of smoking status codes in tabular format. If no smoking status was found (empty list was returned), a warning message is displayed.

4.3.2 getSmokingStatus request message

Same as `getDiagnoses`.

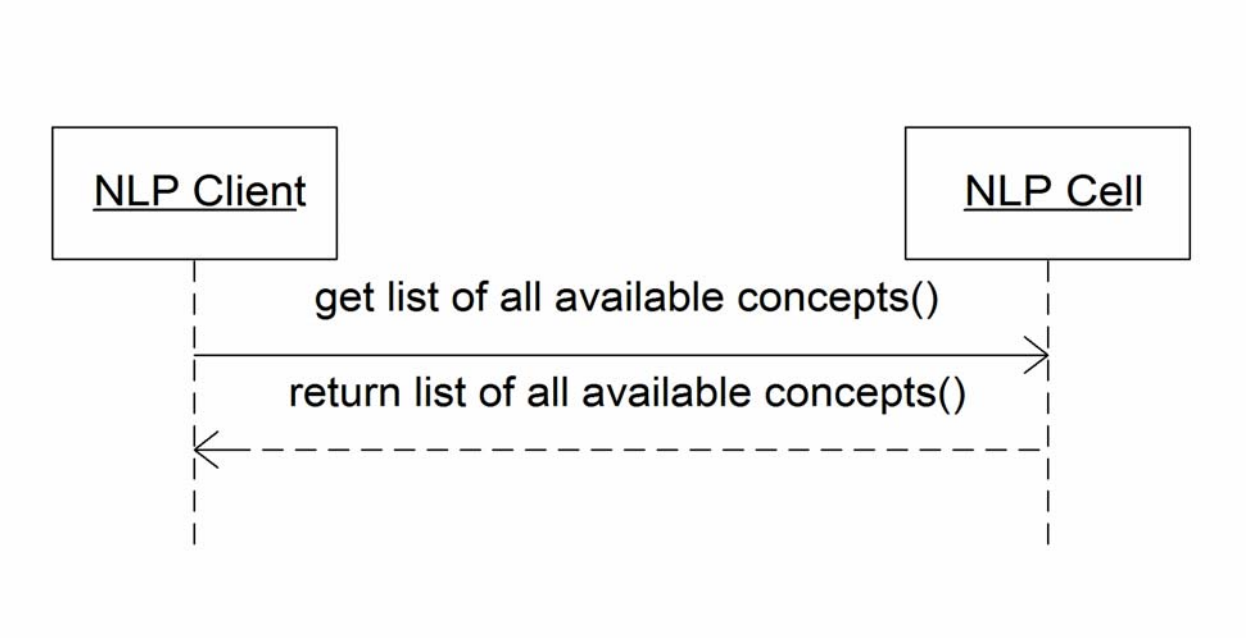
4.3.3 getSmokingStatus response message

```
  <message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:current_smoker</concept_cd>
          <start_date>2007-11-26T14:51:35.156-05:00</start_date>
          <observation_blob>
            <concept_cd type="smokingStatus">
              <code>DSG-NLP:current_smoker</code>
              <section>
                <sectionName><![CDATA[HOSPITAL
COURSE:]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="PROC" />
                </sectionCategs>
              </section>
            </concept_cd>
          </observation_blob>
        </observation>
      </ns2:observation_set>
    </ns2:patient_data>
  </message_body>
```

4.4 getAllConcepts

A getAllConcepts message returns a list of all known concepts discovered in a text document. Currently, this includes principal diagnoses, discharge medications and smoking status. The text

document body and the type of the text document (e.g., LMR note, BWH or MGH discharge summary) are passed in the message_body by the NLP client.



4.4.1 Get a list of all known concepts associated with a text document

The getAllConcepts message is sent by the NLP client to retrieve a list of all known concepts associated with the text document. The sequence of events is as follows:

1. Client requests the list of all known concepts by sending a getAllConcepts message containing text report body and the type of the report.
2. The NLP service performs the following steps:
    a. The text report body is extracted from the message_body.
    b. The type of the report is extracted from the message. Currently, three report types are known: LCS-I2B2:c1010c (MGH Discharge Summary), LCS-I2B2:c1011c (BWH Discharge Summary) and LCS-I2B2:c1009c (LMR Note).
    c. The correct NLP pipeline is identified based on the type of the report (LMR note or discharge summary) and the type of the message (getAllConcepts). The extraction pipelines, one for each report type, are available on the server.
    d. The report is parsed by the pipeline and all known concept codes are extracted.
    e. The response message is generated. Along with each concept code, some extra information about concept is returned, as described in previous sections.

The client displays the list of all discovered concept codes in tabular format. If no concepts were found (empty list of was returned), a warning message is displayed.

4.4.2 getAllConcepts request message

Same as `getDiagnoses`.

4.4.3 getAllConcepts response message

```
  <message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:c0349790</concept_cd>
          <start_date>2007-11-26T16:15:34.984-05:00</start_date>
          <observation_blob>
            <concept_cd type="umlsConcept">
              <cui>c0349790</cui>
              <umlsName>Exacerbation of asthma (disorder)</umlsName>
              <semanticTypes>
                <semanticType name="Finding" tui="T033" />
              </semanticTypes>
              <mappedTerm><![CDATA[Asthma exacerbation]]></mappedTerm>
              <modifiers>
                <modifier name="negationStatus" value="Actual" />
              </modifiers>
              <section>
                <sectionName><![CDATA[PRINCIPAL   DISCHARGE   DIAGNOSIS
;Responsible After Study for Causing Admission)]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="DIS" />
                  <sectionCateg name="PRI" />
                </sectionCategs>
              </section>
            </concept_cd>
          </observation_blob>
        </observation>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:PREDNISONE</concept_cd>
          <start_date>2007-11-26T16:15:34.984-05:00</start_date>
          <observation_blob>
            <concept_cd type="medication">
              <code>DSG-NLP:PREDNISONE</code>
              <section>
                <sectionName><![CDATA[DISCHARGE
MEDICATIONS:]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="DIS" />
```

```
                  <sectionCateg name="MED" />
                </sectionCategs>
              </section>
            </concept_cd>
          </observation_blob>
        </observation>
        <observation>
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>DSG-NLP:current_smoker</concept_cd>
          <start_date>2007-11-26T16:15:34.984-05:00</start_date>
          <observation_blob>
            <concept_cd type="smokingStatus">
              <code>DSG-NLP:current_smoker</code>
              <section>
                <sectionName><![CDATA[HOSPITAL
COURSE:]]></sectionName>
                <sectionCategs>
                  <sectionCateg name="PROC" />
                </sectionCategs>
              </section>
            </concept_cd>
          </observation_blob>
        </observation>
      </ns2:observation_set>
    </ns2:patient_data>
  </message_body>
```

4.5 getCustomConcepts

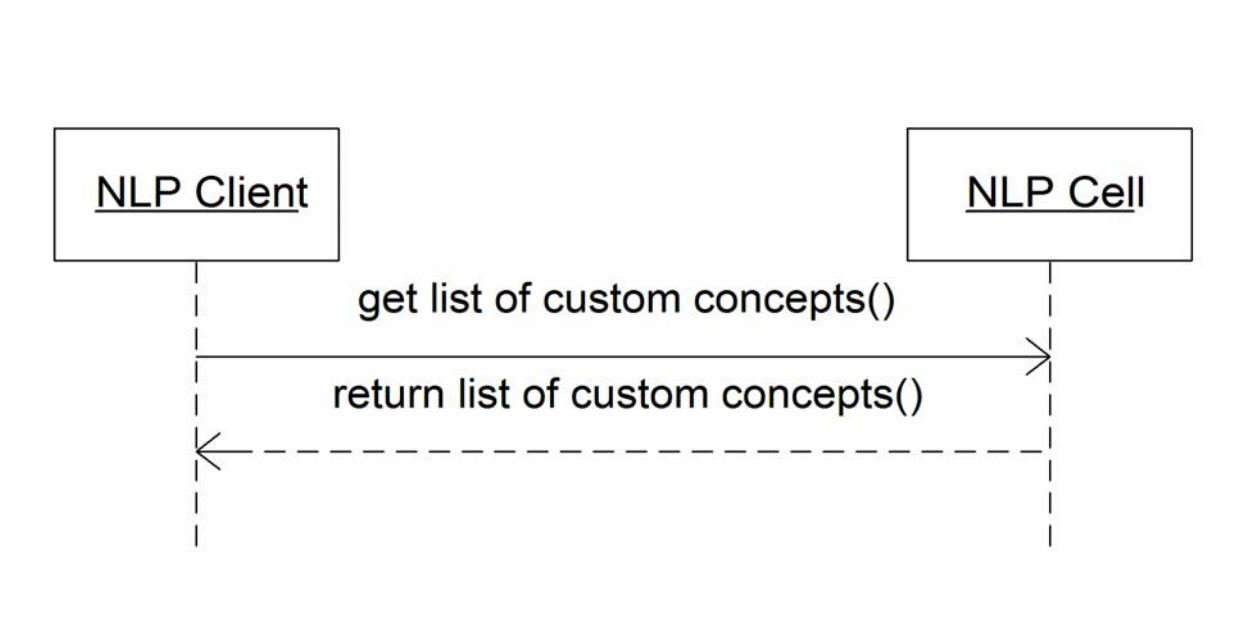
A getCustomConcepts message returns a list of all user-defined concepts discovered in a text document. A user-defined pipeline configuration is passed in the message_body section of the message inside the optional DSG-NLP:param element. The text document body and the type of the text document (e.g., LMR note, BWH or MGH discharge summary) are passed inside this section as well.

Custom pipeline is built on top of a pre-configured pipeline containing all available NLP components as a blueprint. This default pipeline is created on the fly each time a getCustomConcepts request comes in, and is destroyed after the processing is finished. The user can enabled or disable each component in this pipeline, and pass custom configuration parameters to each of the components. Custom parameters can replace the existing parameters, or merge with the existing parameters.

The DSG-NLP:param element contains configuration entries for each component in the pipeline (param elements). It also defines an order of components in a pipeline. Each configuration

entry contains user-specified configuration parameters for this component. They are intended to override the default component configuration. Since these parameters can either replace or merge with an existing component configuration (remember, custom pipeline is built using a complete pipeline as a blueprint), the merge strategy is also defined for in each configuration entry. It is specified as a mergeStrategy attribute, which can take 2 valid values: merge or replace.

The custom configuration entry is removed by the server and is not mirrored back to client. The report text is not mirrored back either.



4.5.1 Get a list of custom concepts associated with a text document

The getCustomConcepts message is sent by the NLP client to retrieve a list of custom (user-defined) concepts associated with the text document. The sequence of events is as follows:

1. Client requests the list of all known concepts by sending a getCustomConcepts message containing text report body and the type of the report.
2. The NLP service performs the following steps:
    a. User-defined configuration is extracted from the message_body section of the request.
    b. The text report body is extracted from the message_body.
    c. The type of the report is extracted from the message. Currently, three report types are known: LCS-I2B2:c1010c (MGH Discharge Summary), LCS-I2B2:c1011c (BWH Discharge Summary) and LCS-I2B2:c1009c (LMR Note).
    d. The default NLP pipeline is identified based on the type of the report (LMR note or discharge summary) and the type of the message (getCustomConcepts). The pipeline is not yet initialized.

e. If user-defined configuration exists and is valid, it is merged with the configuration for the default processing pipeline. The resulting configuration is used to instantiate and initialize the pipeline. Otherwise, the default configuration is used to instantiate and initialize the pipeline.
f. The report is parsed by the pipeline and all known concept codes are extracted. Some types of the concepts may not exist in the output, if user chose to disable parts of the default pipeline. The semantic meaning of the concepts is determined by user. For example, discovered UMLS concepts may designate principal diagnoses or co-morbidities, depending on which section of the documents they were found in. These sections, in turn, may be defined by custom configuration.
g. The response message is generated. Along with each concept code, some extra information about concept is returned, as described in previous sections.
h. The client displays the list of all discovered concept codes in tabular format. If no concepts were found (empty list of was returned), a warning message is displayed.

4.5.2 getCustomConcepts request message

```
<message_body>
    <ns2:patient_data>
      <ns2:observation_set>
        <observation
           download_date="2007-11-26T15:54:21.252-05:00"
           import_date="2007-11-26T15:54:21.252-05:00"
           sourcesystem_cd="RPDR"
           update_date="2007-11-26T15:54:21.252-05:00">
          <event_id>1000001</event_id>
          <patient_id>1234567</patient_id>
          <concept_cd>LCS-I2B2:c1010c</concept_cd>
          <start_date>2007-11-26T15:54:21.252-05:00</start_date>
          <observation_blob>
                ***Report body goes here***
           </observation_blob>
        </observation>
      </ns2:observation_set>
   </ns2:patient_data>
   <DSG-NLP:param
      xmlns:DSG-NLP="http://dsg.harvard.edu/nlp"
      class="hitex.Pipeline"
      description="Pipeline application"
      mergeStrategy="merge" name="PIPELINE">
       <param
            class="hitex.gate.Sectionizer"
            description=""
```

```
     mergeStrategy="merge"
     name="SECTIONIZER" />
<param
     class="hitex.gate.Tokenizer"
     description=""
     mergeStrategy="merge"
     name="TEXT_TOKENIZER" />
<param
     class="hitex.gate.SentenceSplitter"
     description=""
     mergeStrategy="merge"
     name="SENTENCE_SPLITTER" />
<param
     class="hitex.gate.SentenceClassifier"
     description=""
     mergeStrategy="merge"
     name="SMOKING_CLASSIFIER" />
<param
     class="hitex.gate.RegExConceptFinder"
     description=""
     mergeStrategy="merge"
     name="REGEX_CONCEPT_FINDER" />
<param
     class="hitex.gate.POSTagger"
     description=""
     mergeStrategy="merge"
     name="POS_TAGGER" />
<param
     class="hitex.gate.NPSplitter"
     description=""
     mergeStrategy="merge"
     name="NOUN_PHRASE_SPLITTER" />
<param
     class="hitex.gate.UMLSConceptFinder"
     description=""
     mergeStrategy="merge"
     name="UMLS_CONCEPT_FINDER">
     <param
          class="java.lang.Boolean"
          description=""
          mergeStrategy="replace"
          name="filterMappings">true</param>
       <param
          class="java.lang.Boolean"
          description=""
          mergeStrategy="replace"
```

```
               name="stop">true</param>
            <param
               class="java.lang.Boolean"
               description=""
               mergeStrategy="replace"
               name="stem">true</param>
            <param
               class="java.lang.Boolean"
               description=""
               mergeStrategy="replace"
               name="truncate">true</param>
            <param
               class="java.lang.String"
               description=""
               mergeStrategy="replace"
               name="suppress">STRONG_CHV</param>
      </param>
      <param
            class="java.util.List"
            description=""
            mergeStrategy="replace"
            name="COMPONENTS_ORDER">
            <value>SECTIONIZER</value>
            <value>TEXT_TOKENIZER</value>
            <value>SENTENCE_SPLITTER</value>
            <value>SMOKING_CLASSIFIER</value>
            <value>REGEX_CONCEPT_FINDER</value>
            <value>POS_TAGGER</value>
            <value>NOUN_PHRASE_SPLITTER</value>
            <value>UMLS_CONCEPT_FINDER</value>
       </param>
    </DSG-NLP:param>
</message_body>
```

4.5.3 getCustomConcepts response message

Same as `getAllConcepts`.