GELLO REST API - VA Version

A GELLO query can be made to the server at the following address with a HTTP POST message.

1 Formatting
2 Compact JSON type inference
3 Request parameters
4 Reply parameters
5 Requirements
6 Example

```
http://hostname:port/rest/gellov2/patientquery
```

or

```
https://hostname:tlsport/rest/gellov2/patientquery
```

POST request format using JSON.

```
{
"PatientId": string,
"Requests": [
{
"RequestId": string,
"GelloCode": string
},
.....]
}
```

POST reply from server

```
{
"PatientId": string,
"Results": [
{
"RequestId": string,
"wasSuccessful": boolean,
"errors": [string, .....],
"result": { ..... }
},
.....]
}
```

Formatting

Requests and Responses are formatted using JSON. Fields of JSON objects may appear in a different order to that documented.

Compact JSON type inference

OCL types can be inferred from the JSON output. The following rules describe how these OCL types can be inferred.

JSON **number** ---> OCL **"Real"**

JSON **string** ---> OCL **"String"**

JSON **boolean** (true/false) ---> OCL **"Boolean"**

JSON **null** ---> OCL **"Any"** with oclsDefined property = False

JSON **array** ---> OCL **"Sequence(Any)".** Usually the element type can be inferred also however, the array may contain mixed types.

JSON **object** ---> OCL class with class name contained in the **"#type"** class property. If this is omitted or of value "Tuple" it is assumed to be an OCL **Tuple.**

There are some minor exceptions to the above. A **Real** may have the values NaN, +Inf and -Inf, in which case it is represented as a OCL class of type **"Real"** with a sub property of "value" containing the string representation of the exceptional Real value. An **Integer** is descendant of class **Real,** and could be inferred from the additional regex "**\-?[0-9]+/**", however JSON does not make this distinction, and the GELLO engine will perform the necessary conversion if required.

Request parameters

PatientId (required) is a key to retrieve a Patient Context against which all GELLO queries in this request will be made.

Requests (required) is a JSON array containing a list of GELLO queries to be made against the Patient Context

Each request contains the following JSON object

RequestId (required) is a unique Identifier supplied by the user to identify multiple jobs within the request. It should not be empty or null.

GelloCode (required) is a string containing a valid GELLO query to be executed in the HL7 context. It may contain contain white space and line breaks, but these should be escaped according to JSON rules.

Reply parameters

PatientId (required) should be the same key supplied by the request.

Results (required) is a JSON array containing a list of results from the GELLO queries made.

Each result contains the following JSON object. Results may appear out of order but can be collated via the **RequestId** parameter.

RequestId (required) matches one and only one RequestId in the Request.

wasSuccessful (required) contains boolean true or false, representing whether the GELLO query was successful or not.

result (optional) contains the result of the GELLO query in JSON format. The contents of this is dependent on the nature of the GELLO query. It may be missing or null if the GELLO expression failed.

errors (optional) contains a string list of any errors encountered in compiling and/or executing the GELLO query, and should contain at least one item if **wasSuccessful** returned false. It may be missing or empty if the GELLO expression was successful.

Requirements

In order to return results the calling machine must be setup on the Equator as a valid client for Web Access. To add the client computer, in the configuration screen of Equator select Web Access Setup, check the Web Access Allowed check box and then add the computer details to the list of valid addresses.

Example

```
Request
{
"PatientId": "abcd",
"Requests": [
{
"RequestId": "12345",
"GelloCode": "Let msg: String = \"Hello World!!!\" msg"
}
]
}
```

Reply

```
{
"PatientId": "abcd",
"Results": [
{
"requestID": "12345",
"wasSuccessful": true,
"result": "Hello World!!!"
}
]
}
```

Example Gello Version 2 you can try:

```
Let x:Integer = 19
Let y:Real = 100.1011
Let b:Boolean = true
Let z:Real =
If x < 20 and b then
Let a:Real = x *14
in
(a + 20) * y
else
Let a:Real = x *20
in
(a + 25) * y
endif
z
```

Download the Gello Tool here.