Draft **graph URI Scheme**

*The URI scheme name **graph** has been registered with IANA. The registration is provisional, and a specification of this kind will be needed to arrive at a permanent registration.*

Revision record

Version	Date	Status	Comment	Contributors
0.1	18 November 2016	Draft, Neo Internal	First draft	Alastair Green
0.2	20 November 2016	Draft, Neo Internal	State syntax in ABNF, use referenced URIs directly, fix ?/# problem	Alastair Green
0.3	25 July 2017	Draft for CIR	Changed solution to embedded URI (?#) problem; added relative URI; expanded introduction	Peter Furniss

Identifying and addressing graphs

If programs are to access or generate multiple graphs then it is necessary to have a way of identifying any specific graph, and it would be useful to have a way of addressing such a graph.

This document describes a URI scheme that enables the unambiguous identification of a graph, in a way that also permits (but does not mandate) that the URI can be used for location of the graph in a specific graph data store or service.

In the terms defined by [RFC3986] Uniform Resource Identifier (URI): Generic Syntax, this scheme defines the syntax for a graph URI that is a **name** (that only identifies the graph) and can also be a **locator** (specifying how the graph can be accessed).[1]

Since the form (component data structures) of a graph will vary, a graph URI used as a locator must define both where the graph is and how to access it. This differs from most URI schemes, where the scheme itself specifies how the resource can be accessed. (E.g. with the http scheme, HTTP protocol verbs are used on the single resource defined by the URI.) Graph URIs used as locators contain a “locator-scheme” that specifies the form of the graph and an embedded URI as the locator. (E.g. for a graph stored in a relational database, the graph URI might have a locator-scheme that specifies how relational tables/views are used to store graphs, and a jdbc URI that can be used to connect to those tables/views; for a graph stored in a set of HTTP-accessible resources, the locator-scheme could specify the structure of the set of resources and an http URI the top-level or index resource for the set.)

A graph URI may be returned as a graph reference as a result of some operation (e.g. query) on graph located by a graph URI. This URI scheme provides for such a returned graph URI to be relative to the original parent graph URI.

---

[1] **1.1.3. URI, URL, and URN**

A URI can be further classified as a locator, a name, or both. The term "Uniform Resource Locator" (URL) refers to the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). The term "Uniform Resource Name" (URN) has been used historically to refer to both URIs under the "urn" scheme [RFC2141], which are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable, and to any other URI with the properties of a name.

An individual scheme does not have to be classified as being just one of "name" or "locator". Instances of URIs from any given scheme may have the characteristics of names or locators or both, often depending on the persistence and care in the assignment of identifiers by the naming authority, rather than on any quality of the scheme. Future specifications and related documentation should use the general term "URI" rather than the more restrictive terms "URL" and "URN" [RFC3305].

Use of ABNF to define the syntax of a graph URI

The syntax described below is defined in [RFC5234] Augmented BNF. In natural language prose, references to ABNF names are enclosed in angle brackets, thus: `<graph>`.

ABNF rule names that are not defined in this specification are names defined in [RFC3986] which are incorporated by reference (used with identical meaning) here.
Examples are the ABNF names <URI>, `<unreserved> and <scheme>`.

Graph URI Scheme

The scheme name is “graph”..

```
graphURI = “graph” ":" path
```

Path Variants

The path of a graph URI is either an opaque name (in which case the URI is only usable as name) or a locator-scheme and an embedded URI:

```
path =  “name” “:” opaque-name
     /  locator-scheme ":" ( “’"escaped-locator-uri “‘“ /
                             “("escaped-locator-uri “)“ )

opaque-name = unreserved

locator-scheme = scheme

escaped-locator-uri = URI  ; see text below and footnote 2

Locator-uri = URI  ; see text below
```

A <locator-scheme> MUST conform to the rules specified in RFC3986 Uniform Resource Identifier (URI): Generic Syntax for the value of a URI scheme name, and to ensure uniqueness and consistent interpretation SHOULD be a value registered with the openCypher project, such values to be allocated on a “first-come, first-served” basis.

A <escaped-locator-uri> is produced by taking a <locator-uri> (which is an RFC3986-conformant URI) and encoding the following characters with their percent-encoding equivalents:

“%” (percent)	“%25”
“?” (question mark)	“%3F”
“#” (number sign/hash)	“%23”
“(“ (left parenthesis)	“%28”
“)” (right parenthesis)	“%29”
““ (single quote)	“%27”

Other characters SHOULD NOT be percent-encoded.

The encodings of question mark and hash are needed to avoid a general URI parser mis-parsing the graph URI[2].

The encoding of percent sign avoids existing percent-encodings being decoded too soon.

The encoding of parentheses and single quote avoids confusing the delimitation of the `<escaped-locator-uri>`.

A `<escaped-locator-uri>` is syntactically a URI—however, if escaping substitutions (percent encodings as described above) were made, the resulting URI is likely to be semantically invalid. However, an `<escaped-locator-uri>` can be converted into a `<locator-uri>` by reverse substitutions, and `<graph>` URI processors MUST carry out such substitutions in the course of dereferencing.

[2] The OASIS Committee Specification of the Extensible Resource Identifier (XRI) Syntax V2.0 uses a similar technique for ensuring that generic URI processors will not break up embedded URIs in a URI. See *2.3.2 Escaping Rules for XRI Syntax.*

Examples of graph URIs

The following five example show how these two variant syntaxes can be used to form `<graph>` URIs.

Opaque

```
graph:name:ec654ec1-7f8f-11e3-ae96-b385f4bc450c
```

RDF

```
graph:rdf+http:'http://chucknorris.com~data_~chuck~foaf_based_near'
```

Bolt anonymous graph

```
graph:bolt:'bolt+routing://production'
```

Bolt named graph

```
graph:bolt:(bolt+routing://production/west-coast/orders/snap/2016-04-12)
```

Graph stored in HDFS using Parquet storage with conventionally named files for edges and vertices

```
graph:gve+hdfs+parquet:(hdfs://production/west-coast/orders/snap/2016-04-12)
```

In these examples, the strings

```
rdf+http
bolt+routing
gve+hdfs+parquet
```

are instances of `<locator-scheme>`.

The strings

```
http//chucknorris.com/data_/chuck/foaf_based_near
bolt+routing//production
bolt+routing://production/west-coast/orders/snap/2016-04-12
```

are instances of `<locator-uri>`.

TBS example of `<escaped-locator-uri>`

Specification of Locator Schemes

A value of <locator-scheme> MUST have a specification of its meaning, to enable a consumer of a graph URI which contains a <locator-scheme> to correctly interpret the <locator-uri> of that URI. Such a specification is a **Locator Scheme Specification**.

Each value of <locator-scheme> MUST be unique and the value and a link to the scheme specification MUST be registered (on a first-come, first-served basis) with the openCypher project.

A Locator Scheme Specification MUST include a definition of the permitted syntax and meaning of <locator-uri> when used in conjunction with the <locator-scheme>.

The URI scheme or schemes that may be used in a <locator-URI> are known as **Locator URI Schemes**.

The permitted syntax of a Locator URI Scheme SHOULD be defined in a separate specification which is incorporated by reference into a Locator Scheme Specification, but MAY be defined as part of that specification.

Note that the meaning (interpretation by a consumer of a graph URI) of a <locator-uri> may or may not be fully defined by its Locator URI Specification.

A Locator Scheme Specification MAY specify whether relative graph URIs are supported in that scheme. If relative URIs are supported, the locator scheme will be the same and the relative URI MUST be resolved by reference to the <locator-uri> of the parent graph URI.

Example Locator Scheme Specifications

The following skeletal examples show the kind of information required in a Locator Scheme Specification.

Specification of graph URI `<locator-scheme>` **"bolt"**

A `<locator-uri>` used in conjunction with the `<locator-scheme>` **bolt** MUST use a URI conforming to one of the URI Schemes **bolt** or **bolt+routing.** These schemes are registered with IANA, and the Locator URI Specifications for these schemes are specified in the IANA URI scheme registration document available at [URI].

The meaning of a `<locator-uri>` for the **bolt** `<locator-scheme>` is fully defined by the relevant Locator URI specifications, and any service implementing those specifications will be able to locate a resource (graph) identified by such a URI.

Specification of graph URI <locator-scheme> "**gve+hdfs+parquet"**

A `<locator-uri>` used in conjunction with the `<locator-scheme>` **gve+hdfs+parquet** MUST be a URI conforming to one of the URI Schemes **hdfs**, **webhdfs** or **file.**

The final path element in the **<locator-URI>** MUST resolve to (be capable of interpretation by the service consuming a graph URI as) a file system directory.

That file system directory contains sub-directories and files organized and formatted in conformance with the specification *openCypher Standards for File Representations of Property Graphs,* "Vertices and Edges in Parquet Format".

References

[RFC3986] Uniform Resource Identifier (URI): Generic Syntax