Neo4j Connector Documentation

Contents

[Introduction 1](#_Toc81140973)

[Key Data Structures 1](#_Toc81140974)

[Neo4j Database 1](#_Toc81140975)

[Neo4j Element 1](#_Toc81140976)

[Neo4j Node 1](#_Toc81140977)

[Neo4j Relation 1](#_Toc81140978)

[Usage 2](#_Toc81140979)

[Event Delegates 2](#_Toc81140980)

[Output Arrays 2](#_Toc81140981)

[Filter Functions 3](#_Toc81140982)

[Appendix 4](#_Toc81140983)

[1. Neo4jDatabase functions, delegates and output arrays 4](#_Toc81140984)

# Introduction

## Key Data Structures

### Neo4j Database

The primary functional unit of the Neo4jConnector Plugin is the *Neo4jDatabase* Class. This class is meant to be an abstraction of an online Neo4j Database that allows users to easily query, retrieve and filter results within Unreal Blueprints and/or C++. You can create many instances of the *Neo4jDatabse* Class to query different databases, or many instances for the same database.

The *Neo4jDatabase* class inherits from UObject, so to create a *Neo4jDatabase* Object in a blueprint you need to use the “Construct OBJECT from Class Node”. After the *Neo4jDatabse* object is created, it must be initialized.

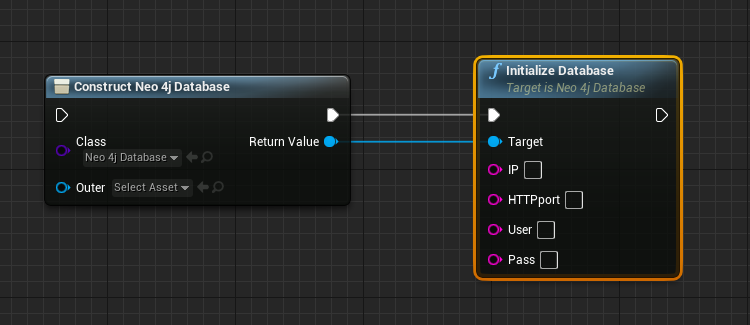


Figure Creation and initialization of a new Neo4jDatabase Object.

### Neo4j Element

*Neo4jElement* is the base data structure from which the *Neo4jNode* and *Neo4jRelation* data structures inherit from. The term “element” will be used to refer to both “nodes” and “relationships”, which is to say nodes and relations are both subsets of elements. Every element has a unique **ID** and a set of **Properties**. Element IDs are used in many of the functions in this API.

### Neo4j Node

*Neo4jNode* is the data structure which represents a node in a graph database. Along with a unique ID and set of properties which it inherits from the *Neo4jElement* structure, the *Neo4jNode* structure also has a set of **Labels**

### Neo4j Relation

*Neo4jRelation* is the data structure which represents a relationship between two nodes in the graph database.

# Usage

## Event Delegates

Since communication between Unreal Engine and Neo4j happens asynchronously, it is not possible to retrieve the result of a Neo4j query in the same function call as the query itself. To get around this, event delegates are used to let users set which functions should be called when a query returns a result. The results of a query are stored in the *Neo4jDatabase* object that queried the database.

The *Neo4jDatabase* class maps a set of event delegates to fire off when certain query types are sent. The table below shows the mappings of functions to event delegates.

Table Mapping of Neo4jDatabase functions with their respective event delegates.



i.e. The *OnGetElementCompleteDelegate* Event Delegate will fire off when the query sent from the *GetElementsByID* function receives a result.

## Output Arrays

The results of the query are then stored in a variable attached to the *Neo4jDatabase* object. Each function/query type that returns values will store their results in a corresponding array of values. Functions that create or update Neo4j Elements will return those created/updated nodes in the corresponding output array.

The table below shows the mappings of functions to output arrays.

Table Mapping Neo4jDatabse functions with their respective output arrays.



i.e. Once the *GetElementsByID* query receives a result, it will store that result in the *getElementOutput* array attached to the *Neo4jDatabase* object.

## Filter Functions

After sending a query, the results will be stored in an output array. The output array is an array of *Neo4jNodes.* To retrieve properties from nodes, you will need to use some of the filter functions. The filter functions have two inputs: 1) a node and 2) the property name.

For example, if a node has a integer property named “id” we can retrieve this property from a node using the *FilterIntProperty* function.

The general flow of a Neo4j Query from Unreal Engine is as such:

1. Bind an event to the corresponding event delegate. (i.e. Bind *OnGetElementCompleteDelegate*)
2. Query the Database (i.e. call the *GetElementsByID* function)
3. Get the results (i.e. Access the array *getElementOutput* attached to the *Neo4jDatabase* object)
4. Filter the desired info from results

# Appendix

## Neo4jDatabase functions, delegates and output arrays

