|  |
| --- |
| NCBI |
| Pubseq Gateway Server (PSG) |
| Overview and the Protocol Specification |
|  |
| **Sergey Satskiy** |
| **1/24/2018** |

|  |
| --- |
| Document version: 1.10 |

Changes history

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | What changes |
| 1.0 | Jan 24, 2018 | Sergey Satskiy | Initial revision |
| 1.1 | Mar 08, 2018 | Sergey Satskiy | Added format #2 for blob retrieval; GetBlob diagram |
| 1.2 | Mar 09, 2018 | Sergey Satskiy | Changes in the resolution request URL format |
| 1.3 | Mar 23, 2018 | Sergey Satskiy | Adding ADMIN/config request |
| 1.4 | Mar 26, 2018 | Sergey Satskiy | Adding ADMIN/info request |
| 1.5 | Mar 30, 2018 | Sergey Satskiy | Adding ADMIN/status request |
| 1.6 | Apr 16, 2018 | Sergey Satskiy | Changes in the response format; adding description of the communication protocol. |
| 1.7 | Apr 19, 2018 | Sergey Satskiy | Addig [server]/maxretries setting description |
| 1.8 | Apr 20, 2018 | Sergey Satskiy | Updates on the communication protocol |
| 1.9 | Apr 24, 2018 | Sergey Satskiy | Updates on the communication protocol |
| 1.10 | May 5, 2018 | Sergey Satskiy | Updates related to logging |

Table of Contents

[Pubseq Gateway Server (PSG) 3](#_Toc511653370)

[Requirements 3](#_Toc511653371)

[Overview 4](#_Toc511653372)

[Communication Protocol 5](#_Toc511653373)

[Files Architecture 7](#_Toc511653374)

[Requests 7](#_Toc511653375)

[Get Blob Request 7](#_Toc511653376)

[Get Request 7](#_Toc511653377)

[Configuration 8](#_Toc511653378)

[Information 9](#_Toc511653379)

[Status 11](#_Toc511653380)

[Cassandra Database 13](#_Toc511653381)

[entity 13](#_Toc511653382)

[largeentity 13](#_Toc511653383)

[Monitoring and Maintenance 14](#_Toc511653384)

[Command Line Arguments 15](#_Toc511653385)

[Configuration Parameters 15](#_Toc511653386)

[[LMDB\_CACHE] section 15](#_Toc511653387)

[[SERVER] section 15](#_Toc511653388)

[[CASSANDRA\_DB] section 16](#_Toc511653389)

[Appendix 18](#_Toc511653390)

[GetBlob Diagram 18](#_Toc511653391)

# Pubseq Gateway Server (PSG)

This document provides an overview of the Pubseq Gateway server functionality and requirements to the various aspects of the server lifecycle. Basically the server provides two types of services:

* accession resolution
* blobs retrieval with its accsessions

Accessions are string identifiers.

# Requirements

Below is a list of major requirements to the Pubseq Gateway server.

* The server must operate as a Linux operating system daemon.
* The server must read all the settings from a configuration file only.
* The server must be able to work in a cloud environment.
* The server must serve many clients simultaneously.
* The logging facilities must be provided via standard C++ toolkit facilities
* The server must provide an interface for monitoring.
* The protocol to communicate to the clients is HTTP/2.
* The server must support alerts
* The server must not deal neither with authentication nor with authorization. These features, if necessary, must be implemented outside of the server.
* The server must not keep any information about the clients it serves.

# Overview

Basically, the Pubseq Gateway server is stateless and operates in request – response mode.

The diagram below shows the main actors and entities involved into a typical Pubseq Gateway application.



The clients establish TCP/IP connections using HTTP/2 with the Pubseq Gateway server via an API (psg\_client library), and they send requests over the established connection.

The Cassandra DB stores two major types of objects: resolutions for accessors and BLOBs. So the clients receive the BLOBs through the pubseq server while the resolutions are used by the server internally.

To speed up the resolution process there is a local copy of them stored in an LMDB file. The file is populated by a synchronization utility showen on the diagram in green.

# Communication Protocol

The communication with the server is provided over HTTP/2.

The requests are standard URLs so the server extracts the parameters in a standard way.

The responces differ depending on a stage. If a problem is detected with the received URL or with the parameters then the standard HTTP/2 way is used. I.e. the status code can be 404, 400, 502, 503 etc. depending on a particular case.

In all other cases the HTTP/2 status code is 200 regardless the requested data are provided or there was an error retrieving them. The response is sent as a set of chunks and uses the following format:

\n\nPSG-Reply-Chunk: item\_type=...&chunk\_type=...[&size=...]&...\n

<DATA, "size" bytes in total>

The supported item types are:

* resolution
* blob
* reply

The supported chunk types are:

* data
* meta
* error
* message

**Note**: the “Content-Length” HTTP/2 header is not used in responses unless it is one of the ADMIN requests.

All data chunks can appear in any order, including item\_type=resolution and item\_type=reply.

For each data blob, blob\_n\_chunks must appear in an item\_type=blob&chunk\_type=meta. It must account for each and every chunk of data (PSG-Reply-Chunk – regardless of the chunk\_type, and including itself too) related to this particular data blob.

Example of the chunks:

[BOF]  
  
PSG-Reply-Chunk: item\_type=resolution&chunk\_type=data&size=1001&....

<RESOLUTION INFO HERE>

PSG-Reply-Chunk: item\_type=blob&chunk\_type=data&size=2345&blob\_id=111.22&blob\_chunk=37&....

<BINARY DATA HERE>

PSG-Reply-Chunk: item\_type=blob&chunk\_type=data&size=2222&blob\_id=222.33&blob\_chunk=8&....

<BINARY DATA HERE>

PSG-Reply-Chunk: item\_type=reply&chunk\_type=meta&reply\_n\_chunks=133

PSG-Reply-Chunk: item\_type=blob&chunk\_type=meta&blob\_id=222.33&blob\_n\_chunks=101&blob\_n\_data\_chunks=100

PSG-Reply-Chunk: item\_type=blob&chunk\_type=error&size=22&blob\_id=111.222&status=404&code=5&severity=critical

<ERROR MESSAGE HERE>

PSG-Reply-Chunk: item\_type=blob&chunk\_type=message&size=44&blob\_id=111.222&status=200&code=3&severity=info

<TEXT MESSAGE HERE>

PSG-Reply-Chunk: item\_type=blob&chunk\_type=data&size=3333&blob\_id=111.22&blob\_chunk=3&....

<BINARY DATA HERE>[EOF]

|  |  |
| --- | --- |
| Field | Description |
| blob\_id | The blob identifier which appears in the form of <sat>.<sat key> where both are integers. |
| blob\_chunk | The blob chunk number (0-based). |
| blob\_n\_chunks | The total number of chunks sent about the blob regardless of the chunk type. I.e. it includes both data and meta chunks. |
| blob\_n\_data\_chunks | The total number of the blob data chunks sent. |
| status | HTTP/2 status code |
| code | Error code – a unique code which helps to identify the exact source of the error. |
| severity | Error or message severity.  Available values for the errors are:   * error * critical * fatal   Available values for the messages are:   * info * warning * trace |
| reply\_n\_chunks | Total number of reply chunks including this very chunk. |

# Files Architecture

The diagram below shows the files used by Pubseq Gateway server.



Pubseq Gateway reads its configuration file (usually named pubseq\_gateway.ini) and configures data access and internal structures correspondingly.

Due to a predicted high load on the server there will be no extensive logging on every event. The only warnings and errors are going to be logged.

# Requests

The server accepts HTTP requests. The section describes the requests and the server responces.

## Get Blob Request

The format of the request:

http://<host:port>/ID/getblob?blob\_id=<sat>.<sat\_key>

where

|  |  |
| --- | --- |
| Parameter | Description |
| host:port | Host and port where the server accepts requests, e.g. iebdev12:2180 |
| sat | The blob sat.  Integer, e.g. 4  There is no default value |
| sat\_key | The blob sat key.  Integer, e.g. 183101333  There is no default value |

For the response format see the “Communication Protocol” chapter.

## Get Request

The format of the request:

http://<host:port>/ID/get?accession=<accession>&resolution=<res>&main\_blob=<mb>&  
prefer\_non\_split=<pns>&named\_annots=<na>&external\_annots=<ea>

where

|  |  |
| --- | --- |
| Parameter | Description |
| host:port | Host and port where the server accepts requests, e.g. iebdev12:2180 |
| accession | Accession which needs to be resolved, e.g. P43208.1  There is no default value |
| res | The type of the required resolution. Accepted values (case sensitive):   * fast * full   There is no default value  At the time of writing the only “fast” type is supported. |
| mb | The flag which tells if the main blob is required. Accepted values (case sensitive):   * yes * no   There is no default value |
| pns | The flag which tells if non split option is preferred. Accepted values (case sensitive):   * yes * no   The default value is “no”.  At the time of writing the parameter is not analyzed. |
| na | The flag which tells if the named annotations are required. Accepted values (case sensitive):   * yes * no   The default value is “no”.  At the time of writing the parameter is not analyzed. |
| ea | External annotations.  At the time of writing the parameter is not analyzed. |

For the response format see the “Communication Protocol” chapter.

## Configuration

The format of the request:

http://<host:port>/ADMIN/config

where

|  |  |
| --- | --- |
| Parameter | Description |
| host:port | Host and port where the server accepts requests, e.g. iebdev12:2180 |

Response:

The HTTP header Content-Type is set to “application/json”

The HTTP header Content-Length is set approprietely

The content is formed as a JSON dictionary with the following items:

|  |  |  |
| --- | --- | --- |
| Key | Value Type | Description |
| ConfigurationFilePath | String | Full path on the server local file system to the configuration file |
| Configuration | String | The full content of the configuration file the server started with |

## Information

The format of the request:

http://<host:port>/ADMIN/info

where

|  |  |
| --- | --- |
| Parameter | Description |
| host:port | Host and port where the server accepts requests, e.g. iebdev12:2180 |

Response:

The HTTP header Content-Type is set to “application/json”

The HTTP header Content-Length is set approprietely

The content is formed as a JSON dictionary with the following items:

|  |  |  |
| --- | --- | --- |
| Key | Value Type | Description |
| PID | Integer | Server process PID |
| ExecutablePath | String | Full local file system path to the server executable |
| CommandLineArguments | String | Command line arguments exactly as the server was started including the binary name. |
| UserTime | Double or string | If succeeded then the process user time consumed as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| SystemTime | Double or string | If succeeded then the process system time consumed as a double.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| PhysicalMemory | Integer or string | If succeeded then the number of physical memory bytes available on the host as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| MemoryUsedTotal | Integer or string | If succeeded then the number of total used memory bytes as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| MemoryUsedResident | Integer or string | If succeeded then the number of resident memory bytes as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| MemoryUsedShared | Integer or string | If succeeded then the number of used shared memory bytes as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| ProcFDSoftLimit | Integer or string | If succeeded then the process file descriptor soft limit as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| ProcFDHardLimit | Integer or string | If succeeded then the process file descriptor hard limit as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| ProcFDUsed | Integer or string | If succeeded then the number of used file descriptors as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| CPUCount | Integer | The number of CPUs on the host. |
| ProcThreadCount | Integer or string | If succeeded then the number of threads the process uses as an integer.  In case of an error getting the value from the OS then a string with a fixed value “n/a”. |
| Version | String | Package version X.Y.Z  0.0.0 if built outside of the prepare\_release framework. |
| BuildDate | String | Build timestamp. Format:  MMM DD YYYY HH:mm:SS |
| StartedAt | String | Local time when the server started. Format:  MM/DD/YYYY HH:mm:SS |

## Status

The format of the request:

http://<host:port>/ADMIN/status

where

|  |  |
| --- | --- |
| Parameter | Description |
| host:port | Host and port where the server accepts requests, e.g. iebdev12:2180 |

Response:

The HTTP header Content-Type is set to “application/json”

The HTTP header Content-Length is set approprietely

The content is formed as a JSON dictionary with the following items:

|  |  |  |
| --- | --- | --- |
| Key | Value Type | Description |
| CassandraActiveStatementsCount | Integer | The current number of active Cassandra statements. |
| NumberOfConnections | Integer | The current number of connections to the server. |
| BadUrlPathCount | Integer | The total number of bad URL path requests.  The counter is zero when the server starts and increased appropriately. |
| InsufficientArgumentsCount | Integer | The total number of requests with insufficient argumens.  The counter is zero when the server starts and increased appropriately. |
| MalformedArgumentsCount | Integer | The total number of requests with malformed arguments.  The counter is zero when the server starts and increased appropriately. |
| ResolveNotFoundCount | Integer | The total number of requests when an accession resolution is not found.  The counter is zero when the server starts and increased appropriately. |
| ResolveErrorCount | Integer | The total number of requests when an accession resolution failed.  The counter is zero when the server starts and increased appropriately. |
| GetBlobNotFoundCount | Integer | The total number of requests when a requested blob is not found.  The counter is zero when the server starts and increased appropriately. |
| GetBlobErrorCount | Integer | The total number of requests when there was an error retrieving a blob.  The counter is zero when the server starts and increased appropriately. |
| UnknownErrorCount | Integer | The total number of requests when an unknown error was encountered.  The counter is zero when the server starts and increased appropriately. |
| TotalErrorCount | Integer | The total number of requests with any kind of error encountered.  The counter is zero when the server starts and increased appropriately. |
| AdminRequestCount | Integer | The total number of successful requests for and administrative information.  The counter is zero when the server starts and increased appropriately. |
| ResolveRequestCount | Integer | The total number of successful requests to resolve an accession.  The counter is zero when the server starts and increased appropriately. |
| GetBlobByAccessionRequestCount | Integer | The total number of successful requests to get a blob by accession.  The counter is zero when the server starts and increased appropriately. |
| GetBlobBySatSatKeyRequestCount | Integer | The total number of successful requests to get a blob by sat and sat key.  The counter is zero when the server starts and increased appropriately. |
| TotalSucceededRequestCount | Integer | The total number of successful requests.  The counter is zero when the server starts and increased appropriately. |

# Cassandra Database

For each keyspace two tables are defined.

## entity

The table stores a general blob information. If a blob is small then the blob body is saved right in this table. If a blob is large, the table stores the information of how many chunks the blob is split into and the blob body goes into the largeentity table as a sequence of chunks.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| ent | int | blob identifier  it shared between the entity and largeentity tables |
| data | blob | If a blob is less than a certain size then its body is written here.  Otherwise the blob is split into chunks and the chunks are saved in the largeentity table (this field value gets null) |
| flags | bigint | A bitwise value.  1 – complete  2 – packed  4 – check failed |
| large\_parts | int | The number of chunks in a large blob.  If a blob is small and stored in this table (see the ‘data’ field) then 0 is saved here. |
| modified | timestamp | Blob modification date. |
| size | bigint | Total blob size |

## largeentity

The table stores large blob chunks.

|  |  |  |
| --- | --- | --- |
| Column | Data type | Description |
| ent | int | blob identifier |
| local\_id | int | Consequent (starting from 0) number of a chunk |
| data | blob | Blob chunk data |

# Monitoring and Maintenance

The server code uses the standard C++ Toolkit logging. So logging configuration is the same as for all other NCBI C++ written applications.

The server also supports running counters, status and configuration information which are provided via the HTTP/2 requests, see the ‘admin’ requests decribed above.

At the moment the alerts infrastructure is not supported.

# Command Line Arguments

The table below describes the server command line arguments.

|  |  |
| --- | --- |
| Argument | Description |
| -help | Prints help message and exits. |
| -nodaemon | If given then the server does not daemonize. |
| -version | Prints the server version and exits. |
| -version-full | Prints the server version, the storage version and the protocol version and then exits. |
| -logfile | The file to which the server log should be redirected. |
| -conffile | The file from which the server should read the configuration. |

# Configuration Parameters

Pubseq Gateway reads the configuration from a file. The default name of the server is pubseq\_gateway so (if the –conffile command line argument is not provided) the default configuration file name will be pubseq\_gateway.ini.

The configuration file uses the NCBI standard ini file format with sections and values within sections. The sections below describe each section of the configuration file separately.

## [LMDB\_CACHE] section

|  |  |
| --- | --- |
| Value | Description |
| dbfile | Path to the file where an LMDB with a resolutions cache is located. |

## [SERVER] section

|  |  |
| --- | --- |
| Value | Description |
| port | HTTP port (1...65534)  No default. If port is not specified or is out of range the server will not start |
| workers | Number of HTTP workers (1...100). If an out of range value is provided then a default value will be used.  Default: 32 |
| backlog | Listener backlog (5...2048). If an out of range value is provided then a default value will be used.  Default: 256 |
| maxconn | Max number of connections (5...65000). If an out of range value is provided then a default value will be used.  Default: 4096 |
| optimeout | Operation timeout in milliseconds  Default: 30000 |
| countfile | A file where various event counters are printed into  If not provided (or cannot be opened for writing) and it is a non-daemon run then the counters are printed on stdout  Default: not provided |
| maxretries | The maximum number of retries when a blob is retrieved from Cassandra DB.  Default: 1 |
| log | Tells if the server logging should be switched on or off.  If switched on then each request is accompanied with request start and request end and all incoming parameters are logged too.  Default: true |

## [CASSANDRA\_DB] section

|  |  |
| --- | --- |
| Value | Description |
| ctimeout | Connection timeout in ms  Default: 30000 |
| qtimeout | Query timeout in ms  Default: 5000 |
| namespace | Data namespace  Default: empty string |
| fallbackrdconsistency | Fallback read consistency  Default: false |
| fallbackwriteconsistency | Lower down consistency of BD write operations if local quorum can't be achieved.  0 - default cassandra driver behavior  Default: 0 |
| loadbalancing | Load balancing policy. Accepted values are: DCAware, RoundRobin  Default: DCAware |
| tokenaware | Enables TokenAware routing  Default: true |
| latencyaware | Enables LatencyAware routing  Default: true |
| numthreadsio | Number of io threads to async processing (1...32)  Default: 4 |
| numconnperhost | Number of connections per node (1...8)  Default: 2 |
| maxconnperhost | Maximum count of connections per node (1...8)  Default: 4 |
| keepalive | TCP keep-alive the initial delay in seconds  Default: 0 |
| drvlog | Location of a cassandra driver log file. If not provided then there will be no driver logs produced.  Default: empty string |
| password\_file | Cassandra password file and a section where credentials are stored. If a password\_file is not provided then password\_section value is ignored.  Default: empty, i.e. no user/password combination is used. |
| password\_section |
| service | The value is a load balancer name or a list of host[:port] items (‘,’ or ‘ ‘ separated).  If the value has neither ‘ ‘, nor ‘,’ nor ‘:’ then it is treated as a load balancer name. The load balancer resolved host ports are are sorted in accordance with their rates.  The list of host[:port] items, regardless where it came from – directly from a parameter value or from a load balancer – is analyzed further. The analizis checks that if ports are provided then they are the same. If the port is provided then it is used for the Cassandra cluster. If no ports are provided then the Cassandra driver uses its default one. |
| log | Tells if the Cassandra drivers should produce log records.  If switched on then the records go into the same destination as the rest of the server logging. The logging level matches the one configured for the application.  Default: false |

# Appendix

## GetBlob Diagram

