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

|  |
| --- |
| Document version: 2.3 |

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 |
| 1.11 | May 18, 2018 | Sergey Satskiy | Sat to keyspace mapping description |
| 2.0 | May 24, 2019 | Sergey Satskiy | Massive update of what is currently implemented. |
| 2.1 | May 30, 2019 | Sergey Satskiy | Adding ADMIN/get\_alerts and ADMIN/ack\_alert requests description |
| 2.2 | Jun 18, 2019 | Sergey Satskiy | Adding ADMIN/statistics and the [STATISTICS] configuration file section description |
| 2.3 | Aug 8, 2019 | Sergey Satskiy | Adding ID/get\_tse\_chunk request |

Table of Contents

[Pubseq Gateway Server (PSG) 4](#_Toc16167823)

[Overview 5](#_Toc16167824)

[Communication Protocol 6](#_Toc16167825)

[PSG Protocol 6](#_Toc16167826)

[Exclude Blob Cache 11](#_Toc16167827)

[Requests 12](#_Toc16167828)

[ID/getblob Request 12](#_Toc16167829)

[ID/get Request 13](#_Toc16167830)

[ID/get\_tse\_chunk Request 15](#_Toc16167831)

[ID/resolve Request 16](#_Toc16167832)

[ID/get\_na Request 18](#_Toc16167833)

[ADMIN/config Request 19](#_Toc16167834)

[ADMIN/info Request 19](#_Toc16167835)

[ADMIN/status Request 22](#_Toc16167836)

[ADMIN/shutdown Request 24](#_Toc16167837)

[ADMIN/get\_alerts Request 25](#_Toc16167838)

[ADMIN/ack\_alert Request 26](#_Toc16167839)

[ADMIN/statistics Request 26](#_Toc16167840)

[TEST/io Request 27](#_Toc16167841)

[favicon.ico Request 28](#_Toc16167842)

[Unknown URL Request 28](#_Toc16167843)

[Cassandra Database 29](#_Toc16167844)

[Monitoring and Maintenance 29](#_Toc16167845)

[Files Architecture 29](#_Toc16167846)

[Client API 30](#_Toc16167847)

[Command Line Arguments 30](#_Toc16167848)

[Signal Handling 30](#_Toc16167849)

[Configuration Parameters 31](#_Toc16167850)

[[LMDB\_CACHE] Section 31](#_Toc16167851)

[[SERVER] Section 31](#_Toc16167852)

[[AUTO\_EXCLUDE] Section 32](#_Toc16167853)

[[ADMIN] Section 32](#_Toc16167854)

[[STATISTICS] Section 32](#_Toc16167855)

[[DEBUG] Section 33](#_Toc16167856)

[[CASSANDRA\_DB] Section 33](#_Toc16167857)

[Appendix 35](#_Toc16167858)

[Protocol Diagrams 35](#_Toc16167859)

[GetBlob Diagram 37](#_Toc16167860)

# Pubseq Gateway Server (PSG)

This document provides an overview of the Pubseq Gateway server functionality. Basically the server provides the following services:

* accession resolution
* blobs retrieval based on accession or on blob identification
* named annotations retrieval
* monitoring of the server

Accessions are string identifiers.

Below is a list of major Pubseq Gateway server implementation details:

* The server operates as a Linux operating system daemon.
* The server reads all the settings from a configuration file only.
* The server serves many clients simultaneously.
* The logging facilities is provided via standard C++ toolkit facilities
* The server provides an interface for monitoring.
* The communication protocol with the clients is HTTP 1.1 or HTTP/2.
* The server does not deal neither with authentication nor with authorization. These features, if necessary, needs to be implemented outside of the server.

# 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 1.1 or HTTP/2 protocols with the Pubseq Gateway server via an API (psg\_client library), and they send requests over the established connection.

The Cassandra DB stores three major types of objects: resolutions for accessions, named annotations and BLOBs. All the data in Cassandra are split into keyspaces. The information of what data are stored in what keyspace is also located in one of the Cassandra tables in a specific keyspace. That keyspace name is configured for the PSG server so the server reads all the mapping at the startup time and uses it later on.

To speed up the data lookups there is a local copy of a certain portion of the Cassandra data stored in a few LMDB files. The file is populated by a synchronization utility shown on the diagram in green. Sometimes the LMDB cache excepts is enough to complete a request however in a worst case scenario a trip to Cassandra will also be required. Generally speaking the server is able to work with cache files (if properly configured) or without. If configured then first the lookups are done in LMDB first and then in Cassandra. Also, the user can control the use of the LMDB cache via a URL parameter for certain requests.

# Communication Protocol

The communication with the server is provided over HTTP 1.1 or HTTP/2 protocols.

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

The responces are standard HTTP 1.1 or HTTP/2 replies however in some cases the reply body introduces a higher level structure which is called PSG protocol. The PSG protocol is comprised out of reply chunks and possibly some data. Whether or not a PSG protocol appears in the reply will be described in the individual requests sections.

## PSG Protocol

Essential link: <https://confluence.ncbi.nlm.nih.gov/pages/viewpage.action?pageId=106579021> – the protocol description in confluence.

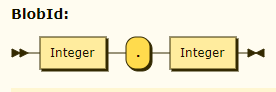
PSG protocol response is comprised of two or more PSG protocol chunks in the HTTP response body.



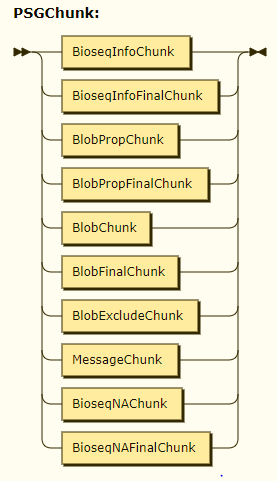
The sequence of the chunks is not guaranteed. The client understands that all the chunks are read when the final chunk is delivered. The final chunk has an information about the total number of chunks the client should expect in response to the request.

The PSG protocol HTTP status code is always 200. If an error is encountered then the error information is supplied in the reply as one of the chunks.

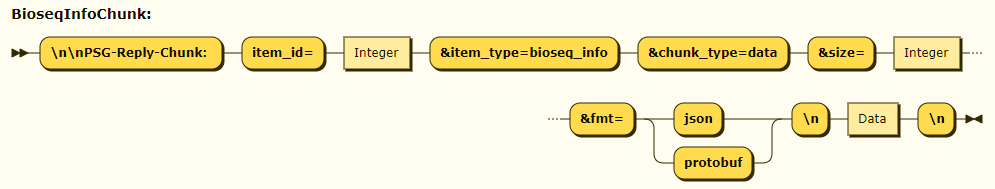
The “Content-type” header is set to “application/x-ncbi-psg”. The “Content-length” header is not set.



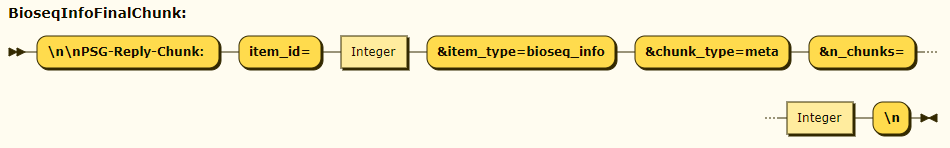
Sometimes it is needed to identify a blob. To do that two integers divided by the ‘.’ Character are used. This notation is used in some of the chunks defined below.



Each PSG chunk uses a fixed prefix and then a set of URL-like paremeters which depends on a chunk type.



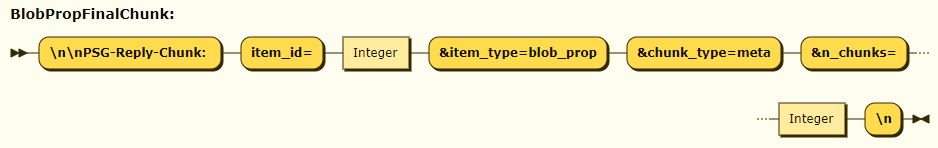
The BioseqInfoChunk is used to send bioseq info data. The item\_id parameter is a positive integer greater than zero which uniquely identifies the data item. Depending on the request the data are supplied in json or a protobuf format. So the data size is returned in the size parameter and the format is specified in the fmt parameter. The data follow the chunk and is a human readable string in case of json or a binary content in case of protobuf.



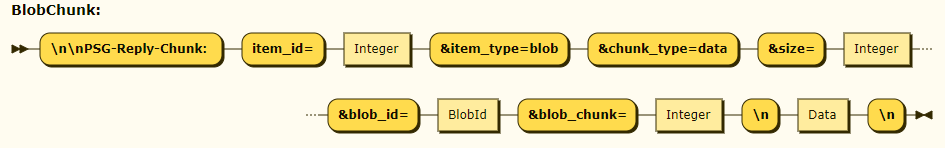
The BioseqInfoFinalChunk is used to send the information of how many chunks were sent about the blob bioseq info – see the n\_chunks parameter.



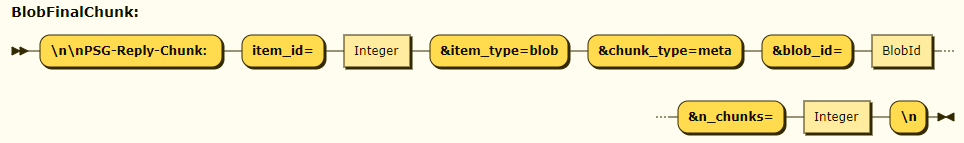
The BlobPropChunk is used to send the appropriate blob properties. The item\_id parameter is a positive integer greater than zero which uniquely identifies the data item. The provided data are always in json format and the size parameter tells the size of data.



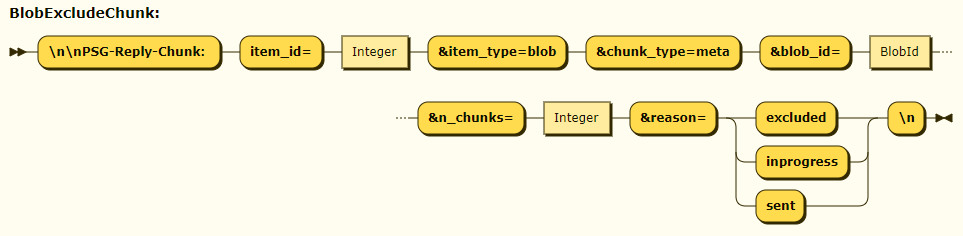
The BlobPropFinalChunk is used to send the information of how many chunks were sent about the blob properties – see the n\_chunks parameter.



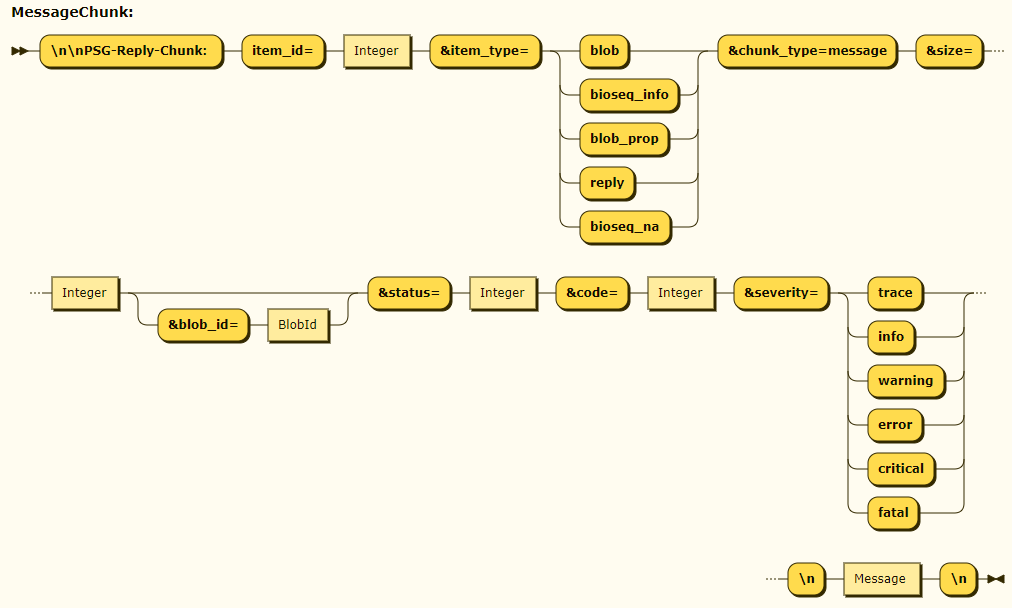
Blobs are stored in Cassandra in a form of chunks. A blob may have an arbitrary number of chunks and each of them could be of an arbitrary size. When a blob is sent to the client the chunks are transferred to the client exactly as they are stored in Cassandra. The blob chunks are numbered consequently starting from zero. So the item\_id parameter uniquely identifies the blob; it is greater than zero and stays the same for all the blob chunks. The size parameter tells the chunk size in bytes. The blob\_id parameter identifies the blob while the blob\_chunk tells the chunk sequential number.



When all blob chunks are sent to the client the server sends one more chunk with the blob finilizing information. The item\_id parameter value matches the BlobChunk chunk item\_id. The same is true about the blob\_id parameter value. The n\_chunks parameter value tells how many chunks were sent in total about the blob including this very chunk.



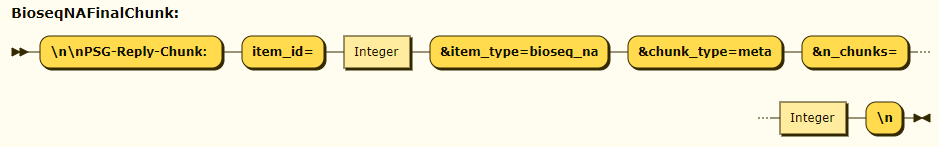
If the blob exclude cache feature is switched on, then the server may sent the BlobExcludeChunk chunk instead of the BlobChunks and the BlobFinalChunk chunks. The reason parameter in this case provides the exact reason why the blob was not sent.



In case of warnings, errors etc the server sends the MessageChunk. The message is linked to an appropriate item\_id as well as to the item type. If appropriate, the blob id is also supplied. The rest of the parameters describe a message similar to the C++ toolkit log messages.



The BioseqNAChunk is sent when the server responses with a named annotation information. The seq\_acc, seq\_ver and seq\_type provide accession, version and type respectively.



Each BioseqNAChunk is followed by a BioseqNAFinalChunk. The final chunk tells the total number of chunks sent about the item\_id.



When the server finishes response it sends the PSGFinalChunk. This chunk tells the total number of chunks in the response (including this very chunk).

The PSG protocol reserves the item\_id value zero for the cases when a chunk is related to the whole response. The examples of such chunks are error messages and a final response chunk.

# Exclude Blob Cache

The PSG server supports blob requests based on seq\_id and seq\_id\_type. In this case there is a procedure of the provided identification resolution into a pair of sat and a sat\_key. This pair of values is used internally to retrieve and transfer the blob. It may happened that the client issues a massive number of blob retrieve requests using seq\_ids. In this scenario the resolution of many different seq\_ids may lead to the very same pair of sat and sat\_key. Consequently it will lead to transferring the same blob many times to the the very same client.

To address the problem - i.e. to avoid transferring the same blob more than one time to the same client – the PSG server introduces the exclude blob cache feature. It works as follows. When a client requests a blob with seq\_id/seq\_id\_type identification it also provides the client name as well as an optional list of the blob\_ids which the client already has. When the seq\_id/seq\_id\_type resolution procedure is finished the result sat/sat\_key is looked in the list provided by the client. If found then the blob is not sent. If not found then the exclude blob cache is looked up. If the blob\_id is found for the client then the blob is not sent. Otherwise a records about the blob is created in the cache.

The BlobExcludeChunk has the reason parameter which tells the following:

* excluded: the blob was found in the list supplied by the client.
* inprogress: the blob was found in the cache; the transfer of the blob chunks is in progress. There is no guarantee that all the blob chunks will be transferred successfully.
* sent: the blob was found in the cache; the server has finished transferring the blob to the client before.

The cache supports automatic garbage collection. It is provided basing on the timeout when the client communicated with the server last time as well as on the maximum number of the most recent blob records per client.

# Requests

The server accepts HTTP 1.1 and HTTP/2 GET requests. The section describes the requests and the server responces.

The requests are split into three cathegories:

* Data requests
* Administrative requests
* Test requests

The distinguish between the request cathegory is the first path element in the request URL.

## ID/getblob Request

The format of the request:

http://<host:port>/ID/getblob

where

|  |  |
| --- | --- |
| Parameter | Description |
| blob\_id=<sat>.<sat\_key> | The blob sat and sat key. Both must be positive integers.  Mandatory parameter |
| tse=<tse\_opt> | TSE option.  The following blobs depending on the value:   |  |  |  | | --- | --- | --- | | Value | ID2 split available | ID2 split not available | | none | Nothing | Nothing | | whole | Split INFO blob only | Nothing | | orig | Split INFO blob only | All Cassandra data chunks of the blob itself | | smart | All split blobs | All Cassandra data chunks of the blob itself | | slim | All Cassandra data chunks of the blob itself | All Cassandra data chunks of the blob itself |   Optional parameter. Default value: orig |
| last\_modified=<last\_mod> | Last modified, integer.  If provided then the exact match will be requested with the Cassandra storage corresponding field value.  Optional parameter.  By default the most recent match will be provided. |
| use\_cache=<cache> | Allowed values:   * no: do not use LMDB cache (tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP) at all; go straight to Cassandra storage. * yes: do not use tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP from Cassandra storage at all. I.e., exclusively use the cache for all seq-id resolution steps. If the seq-id cannot be fully resolved through the cache alone, then code 404 must be returned.   Optional parameter.  By default (no use\_cache option specified), the behavior is to use the LMDB cache if at all possible; then, fallback to Cassandra storage. |

The response uses the PSG protocol.

The HTTP header Content-Type is set to “application/x-ncbi-psg”.

The HTTP header Content-Length is not set.

The HTTP 1.1 or HTTP/2 status code is always 200.

In case of success the following PSG protocol chunks will appear:



The sequence of chunks is not guaranteed.

In case of errors a MessageChunk will appear accompanied by the PSGFinalChunk.

## ID/get Request

The format of the request:

http://<host:port>/ID/get?

where

|  |  |
| --- | --- |
| Parameter | Description |
| seq\_id=<seq\_id> | SeqId of the blob to be retrieved (string).  Mandatory parameter. |
| seq\_id\_type=<seq\_id\_type> | SeqId type of the blob to be retrieved (integer > 0).  Optional parameter. |
| use\_cache=<cache> | Allowed values:   * no: do not use LMDB cache (tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP) at all; go straight to Cassandra storage. * yes: do not use tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP from Cassandra storage at all. I.e., exclusively use the cache for all seq-id resolution steps. If the seq-id cannot be fully resolved through the cache alone, then code 404 must be returned.   Optional parameter.  By default (no use\_cache option specified), the behavior is to use the LMDB cache if at all possible; then, fallback to Cassandra storage. |
| tse=<tse\_opt> | TSE option.  The following blobs deoending on the value:   |  |  |  | | --- | --- | --- | | Value | ID2 split available | ID2 split not available | | none | Nothing | Nothing | | whole | Split INFO blob only | Nothing | | orig | Split INFO blob only | All Cassandra data chunks of the blob itself | | Smart | All split blobs | All Cassandra data chunks of the blob itself | | slim | All Cassandra data chunks of the blob itself | All Cassandra data chunks of the blob itself |   Optional parameter. Default value: orig |
| exclude\_blobs=<exclude\_list> | A comma separated list of BlobId which client already has. If provided then if the resolution od seq\_id/seq\_id\_type matches one of the blob id then the blob will not be sent.  Optional parameter. |
| client\_id=<client\_id> | The client identifier (string).  If provided then the exclude blob feature takes place.  Optional parameter. |

The response uses the PSG protocol.

The HTTP header Content-Type is set to “application/x-ncbi-psg”.

The HTTP header Content-Length is not set.

The HTTP 1.1 or HTTP/2 status code is always 200.

In case of success the following PSG protocol chunks will appear:



The sequence of chunks is not guaranteed.

In case of errors a MessageChunk will appear accompanied by the PSGFinalChunk.

## ID/get\_tse\_chunk Request

The format of the request:

http://<host:port>/ID/get\_tse\_chunk

where

|  |  |
| --- | --- |
| Parameter | Description |
| tse\_id=<sat>.<sat\_key> | The TSE blob sat and sat key. Both must be positive integers.  Mandatory parameter |
| chunk=<chunk number> | The requied TSE blob chunk number. It must be greater than 0 integer.  Mandatory parameter |
| split\_version=<split version> | The TSE blob split version. It must be an integer.  Mandatory parameter |
| use\_cache=<cache> | Allowed values:   * no: do not use LMDB cache (BLOB\_PROP table) at all; go straight to Cassandra storage. * yes: do not use BLOB\_PROP table from Cassandra storage if the initial lookup of the TSE blob in the BLOB\_PROP cache succeeded and its split version matched the requsted one.   Optional parameter.  By default (no use\_cache option specified), the behavior is to use the LMDB cache if at all possible; then, fallback to Cassandra storage. |

The response uses the PSG protocol.

The HTTP header Content-Type is set to “application/x-ncbi-psg”.

The HTTP header Content-Length is not set.

The HTTP 1.1 or HTTP/2 status code is always 200.

In case of success the following PSG protocol chunks will appear:



## ID/resolve Request

The format of the request:

http://<host:port>/ID/resolve

where

|  |  |
| --- | --- |
| Parameter | Description |
| seq\_id=<seq\_id> | SeqId of the bioseq info to be retrieved (string).  Mandatory parameter. |
| seq\_id\_type=<seq\_id\_type> | SeqId type of the bioseq info to be retrieved (integer > 0).  Optional parameter. |
| psg\_protocol=<psg\_proto> | Indication whether to use the PSG protocol or not (Boolean).  Accepted values are: yes and no.  Optional parameter.  Default: false |
| use\_cache=<cache> | Allowed values:   * no: do not use LMDB cache (tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP) at all; go straight to Cassandra storage. * yes: do not use tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP from Cassandra storage at all. I.e., exclusively use the cache for all seq-id resolution steps. If the seq-id cannot be fully resolved through the cache alone, then code 404 must be returned.   Optional parameter.  By default (no use\_cache option specified), the behavior is to use the LMDB cache if at all possible; then, fallback to Cassandra storage. |
| fmt=<format> | The bioseq info data format (string).  Accepted values:   |  |  | | --- | --- | | protobuf | Bioseq info will be sent as a protobuf binary data | | json | Bioseq info will be sent as a serialized JSON dictionary | | native | The PSG server will decide what format to use: protobuf or json. |   Optional parameter.  Default: native |
| all\_info=<bool\_val>  canon\_id=<bool\_val>  seq\_ids=<bool\_val>  mol\_type=<bool\_val>  length=<bool\_val>  state=<bool\_val>  blob\_id=<bool\_val>  tax\_id=<bool\_val>  hash=<bool\_val>  date\_changed=<bool\_val> | It is used to specify explicitly what values to include/exclude from the provided bioseq info. The accepted values are yes and no.  It could be used e.g. as follows:  …&all\_info=yes&length=no  In this case all the fields will be supplied except of the length.  Optional parameters.  Default: do not include anything.  The parameters are taken into consideration only if the effective data format is JSON. |

The response depends on the ‘psg\_protocol’ parameter. If psg\_protocol effective value is false then the standard HTTP 1.1 or HTTP/2 will be used. Otherwise the PSG protocol is used on top.

Standard HTTP 1.1 or HTTP/2 case:

The HTTP header Content-Type is set to “application/json” if the selected format is json. Otherwise it is set to “application/octet-stream”.

The HTTP header Content-Length is set appropriately.

The response body will have the bioseq info in the appropriate format.

The protobuf format description can be found here: <https://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/lxr/source/src/objtools/pubseq_gateway/protobuf/psg_protobuf.proto>

PSG protocol case:

The following PSG protocol chunks will appear:



The sequence of chunks is not guaranteed.

In case of errors a MessageChunk will appear accompanied by the PSGFinalChunk.

## ID/get\_na Request

The format of the request:

http://<host:port>/ID/get\_na

where

|  |  |
| --- | --- |
| Parameter | Description |
| psg\_protocol=yes | Fixed parameter, the value must be yes.  Mandatory parameter. |
| seq\_id=<seq\_id> | SeqId of the bioseq info to be retrieved (string).  Mandatory parameter. |
| seq\_id\_type=<seq\_id\_type> | SeqId type of the bioseq info to be retrieved (integer > 0).  Optional parameter. |
| names=<names> | A comma separated list of named annotations to be retrieved.  Mandatory parameter. |
| use\_cache=<cache> | Allowed values:   * no: do not use LMDB cache (tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP) at all; go straight to Cassandra storage. * yes: do not use tables SI2CSI, BIOSEQ\_INFO and BLOB\_PROP from Cassandra storage at all. I.e., exclusively use the cache for all seq-id resolution steps. If the seq-id cannot be fully resolved through the cache alone, then code 404 must be returned.   Optional parameter.  By default (no use\_cache option specified), the behavior is to use the LMDB cache if at all possible; then, fallback to Cassandra storage. |
| fmt=<format> | The format of the data sent to the client.  Supported values: json and native.  Optional parameter.  Default is json.  Note: at the moment JSON format is always used. |

The response uses the PSG protocol.

The HTTP header Content-Type is set to “application/x-ncbi-psg”.

The HTTP header Content-Length is not set.

The HTTP 1.1 or HTTP/2 status code is always 200.

In case of success the following PSG protocol chunks will appear:



The sequence of chunks is not guaranteed. If

In case of errors a MessageChunk will appear accompanied by the PSGFinalChunk.

## ADMIN/config Request

The format of the request:

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

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

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 |

## ADMIN/info Request

The format of the request:

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

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

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. |
| RealTime | Double or string | If succeeded then the process real 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”. |
| UserTime | Double or string | If succeeded then the process user 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”. |
| 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”. |
| MemoryUsedTotalPeak | Integer or string | If succeeded then the peak 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”. |
| MemoryUsedResidentPeak | Integer or string | If succeeded then the peak 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”. |
| MemoryUsedData | Integer or string | If succeeded then the number of used data 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”. |
| MemoryUsedStack | Integer or string | If succeeded then the number of used stack 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”. |
| MemoryUsedText | Integer or string | If succeeded then the number of used text 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”. |
| MemoryUsedLib | Integer or string | If succeeded then the number of used library 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”. |
| MemoryUsedSwap | Integer or string | If succeeded then the number of used swap 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 |
| ExcludeBlobCacheUserCount | Integer | The number of users which have records in the exclude blob cache |

## ADMIN/status Request

The format of the request:

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

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

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

The HTTP header Content-Length is set appropriately.

All the event counters (errors, request counters etc.) are monotonically growing and are set to 0 at the server instance startup.

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. |
| ActiveRequestCount | Integer | The number of active pending requests (does not include admin or test requests). |
| ShutdownRequested | Boolean | true if a graceful shutdown was requested |
| GracefulShutdownExpiredInSec | Integer | If ShutdownRequested is true tehn the number of seconds left till the graceful shutdown timeout is over. |
| BadUrlPathCount | Integer | The total number of bad URL path requests. |
| InsufficientArgumentsCount | Integer | The total number of requests with insufficient argumens. |
| MalformedArgumentsCount | Integer | The total number of requests with malformed arguments. |
| GetBlobNotFoundCount | Integer | The total number of requests when a requested blob is not found. |
| UnknownErrorCount | Integer | The total number of requests when an unknown error was encountered. |
| ClientSatToSatNameErrorCount | Integer | The number of errors when the client supplied a blob sat however the sat could not be mapped to a Cassandra keyspace. |
| ServerSatToSatNameErrorCount | Integer | The number of errors when the server data are referring to a sat which could not be resolved to a Cassandra keyspace. |
| BioseqID2InfoErrorCount | Integer | The number of errors when ID2 info data field is invalid. |
| BlobPropsNotFoundErrorCount | Integer | The number of errors when blob properties were not found. |
| LMDBErrorCount | Integer | The number of errors when the LMDB lookup failed. |
| CassQueryTimeoutErrorCount | Integer | The number of errors when there was a Cassandra request execution timeout. |
| TotalErrorCount | Integer | The total number of requests with any kind of error encountered. |
| InputSeqIdNotResolved | Integer | The number of times the user requested SeqId was not resolved. |
| AdminRequestCount | Integer | The total number of successful requests for and administrative information. |
| ResolveRequestCount | Integer | The total number of successful requests to resolve an accession. |
| GetBlobBySeqIdRequestCount | Integer | The total number of successful requests to get a blob by SeqId. |
| GetBlobBySatSatKeyRequestCount | Integer | The total number of successful requests to get a blob by sat and sat key. |
| GetNamedAnnotationsCount | Integer | The total number of the successful get\_na requests. |
| TestIORequestCount | Integer | The total number of the io requests. |
| TotalRequestCount | Integer | The total number of requests. |
| Si2csiCacheHit | Integer | The total number of times when the si2csi cache had the required data. |
| Si2csiCacheMiss | Integer | The total number of times when the si2csi cache didn’t have the required data. |
| BioseqInfoCacheHit | Integer | The total number of times when the bioseq info cache had the required data. |
| BioseqInfoCacheMiss | Integer | The total number of times when the bioseq info cache didn’t have the required data. |
| BlobPropCacheHit | Integer | The total number of times when the blob properties cache had the required data. |
| BlobPropCacheMiss | Integer | The total number of times when the blob properties cache didn’t have the required data. |
| Si2csiNotFound | Integer | The total number of times when no data were found when the si2csi Cassandra table was looked through. |
| Si2csiFoundOne | Integer | The total number of times when exactly one record was found when the si2csi Cassandra table was looked through. |
| Si2csiFoundMany | Integer | The total number of times when more than one records ware found when the si2csi Cassandra table was looked through. |
| BioseqInfoNotFound | Integer | The total number of times when no data were found when the bioseq\_info Cassandra table was looked through. |
| BioseqInfoFoundOne | Integer | The total number of times when exactly one record was found when the bioseq\_info Cassandra table was looked through. |
| BioseqInfoFoundMany | Integer | The total number of times when more than one records ware found when the bioseq\_info Cassandra table was looked through. |
| Si2csiError | Integer | The total number of errors when the si2csi Cassandra table was looked through. |
| BioseqInfoError | Integer | The total number of errors when the bioseq\_info Cassandra table was looked through. |

## ADMIN/shutdown Request

The format of the request:

http://<host:port>/ADMIN/shutdown?

where

|  |  |
| --- | --- |
| Parameter | Description |
| username=<username> | The user name who wanted to do the shutdown (string).  At the moment the parameter is used only for logging.  Optional parameter.  Default: empty string. |
| auth\_token=<token> | Authorization token (string).  If the configuration [ADMIN]/auth\_token value is provided then the request must have the token value matching the configured to be granted.  Optional parameter.  Default: empty string. |
| timeout=<timeout> | The timeout in seconds within which the shutdown must be performed (integer).  If 0 then it leads to an immediate shutdown.  If 1 or more seconds then the server will reject all new requests and waits till the timeout is over or all the pending requests are completed and then do the shutdown.  Optional parameter.  Default: 10 (seconds) |

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

The HTTP header Content-Type is set to “text/plain”.

The HTTP header Content-Length is set appropriately.

|  |  |
| --- | --- |
| HTTP 1.1 or HTTP/2 status code | Description |
| 202 | Shutdown request has been successfully accepted |
| 409 | The previous shutdown request is shorter |
| 400 | Invalid timeout |
| 401 | Unauthorized |
| 500 | Internal error |

The content may have the corresponding message.

## ADMIN/get\_alerts Request

The format of the request:

http://<host:port>/ADMIN/get\_alerts

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

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

The HTTP header Content-Length is set appropriately.

The content has a JSON dictionary which describes the current alerts, both acknowledged and not.

## ADMIN/ack\_alert Request

The format of the request:

http://<host:port>/ADMIN/ack\_alert?

where

|  |  |
| --- | --- |
| Parameter | Description |
| alert=<alert\_id> | The alert identifier to acknowledge (string)  Mandatory parameter |
| username=<username> | The user name who acknowledges the alert (string).  The parameter is used only for logging.  Mandatory parameter |

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

The HTTP header Content-Type is set to “text/plain”.

The HTTP header Content-Length is set appropriately.

|  |  |
| --- | --- |
| HTTP 1.1 or HTTP/2 status code | Description |
| 200 | The alert has been acknowledged or had already been acknowledged before. |
| 404 | The alert has not been found. |

The content may have the corresponding message.

## ADMIN/statistics Request

The format of the request:

http://<host:port>/ADMIN/statistics?

where

|  |  |
| --- | --- |
| Parameter | Description |
| reset=<yes or no> | If provided as tes then the collected statistics is rest. Otherwise the collected statistics is sent to the client.  Default: no  Optional parameter |

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

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

The HTTP header Content-Length is set appropriately.

The content has a JSON dictionary which stores all the collected statistics since the server restart or the last reset.

## TEST/io Request

The server responses to this request only if the configuration file has the [DEBUG]/psg\_allow\_io\_test value set to true.

The format of the request:

http://<host:port>/TEST/io?

where

|  |  |
| --- | --- |
| Parameter | Description |
| return\_data\_size=<data\_size> | Size in bytes (positive integer up to 1000000000) which should be sent to the client. The data are random.  Mandatory parameter. |
| log=<log> | Boolean parameter which tells if the logging of the request is done or not.  Accepted values are yes and no.  Optional parameter.  Default: no |

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

The HTTP header Content-Type is set to “application/octet-stream”.

The HTTP header Content-Length is set to data\_size.

The content will have the data\_size of randomly generated bytes.

## favicon.ico Request

The format of the request:

http://<host:port>/favicon.ico

The request is often sent by various browsers so the PSG server implements a response to it.

Response:

The standard HTTP 1.1 or HTTP/2 protocol is used.

The status code is always 404.

## Unknown URL Request

Response:

The PSG protocol is used. The status is 200.

The HTTP header Content-Type is set to “application/x-ncbi-psg”.

The HTTP header Content-Length is not set.

The response body has two chunks:



# Cassandra Database

The database structure is described here:

<https://confluence.ncbi.nlm.nih.gov/display/CT/Design+extended+schemata+for+PubSeq+data+in+Cassandra+and+LMDB>

# 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 exposes some internal events monotonically growing counters, status and configuration information. This information is available via /ADMIN/<item> requests (see the request description above). It is also possible to shutdown the server using a URL request.

At the moment the alerts infrastructure is not supported.

# Files Architecture

The diagram below shows the files used by Pubseq Gateway server (LMDB cache files are not shown as they are not read directly).



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.

# Client API

The client API is available here:

<https://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/lxr/source/include/objtools/pubseq_gateway/client/psg_client.hpp>

# Command Line Arguments

The table below describes the server command line arguments.

|  |  |
| --- | --- |
| Argument | Description |
| -help | Prints the help message and exits. |
| -daemonize | If given then the server does the daemonization. By default 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. |

# Signal Handling

The table below describes the signal handling in the server.

|  |  |
| --- | --- |
| Signal | Description |
| INT | Immediate shutdown which will interrupt all currently executed requests. |
| TERM | Gracefull shutdown.  The server will wait till all the currently executed requests are finished. During that time all new requests are rejected. When all requests are finished, the server will shut down.  Will be called e.g. on the host reboot. |
| QUIT | immediate coredump ( most convenient for debugging, Ctrl-\ |
| HUP USR1 USR2 WINCH | The signals are intercepted and logged with no other actions. |

# 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\_si2csi | Path to the file where an LMDB si2csi cache file is located.  If not provided then no cache will be used. |
| dbfile\_bioseq\_info | Path to the file where an LMDB bioseq\_info cache file is located.  If not provided then no cache will be used. |
| dbfile\_blob\_prop | Path to the file where an LMDB blob\_prop cache file is located.  If not provided then no cache will be used. |

## [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)  Default: 32 |
| backlog | Listener backlog (5...2048)  Default: 256 |
| maxconn | Max number of connections (5...65000)  Default: 4096 |
| optimeout | Operation timeout in milliseconds  Default: 30000 |
| maxretries | Max Cassandra operation retries  Default: 1 |
| log | If set to true then request contexts will be created for each request. |
| root\_keyspace | Cassandra root keyspace which is used for discovering the sat to keyspace mapping as well as the location of the SI2CSI and BIOSEQ\_INFO tables  Default: sat\_info |
| slim\_max\_blob\_size | In most cases the blobs are not split because they are... just too small to be split. So, in the spirit of the "slim" purpose such original blobs should be sent to the client.  Default: 10KB |

## [AUTO\_EXCLUDE] Section

The section describes settings for the exclude blob cache feature.

|  |  |
| --- | --- |
| Value | Description |
| max\_cache\_size | Cache size per client.  0 - means it is disabled.  Default: 1000 |
| purge\_percentage | The percentage of the records to purge (of max\_cache\_size; 0 <= int <= 100)  Default: 20 |
| inactivity\_purge\_timeout | Client inactivity purge timeout, seconds, integer.  Used for garbage collecting  Default: 60 |

## [ADMIN] Section

|  |  |
| --- | --- |
| Value | Description |
| auth\_token | Authorization token for the shutdown request.  If provided then the URL shutdown request must have the corresponding auth\_token parameter to be authorized.  If not provided then any URL shutdown request will be authorized. |

## [STATISTICS] Section

|  |  |
| --- | --- |
| Value | Description |
| small\_blob\_size | The statistics for the blob retrieving timing is collected depending on the blob sizes in separate bins. The first bin covers the range of sizes from 0 till small\_blob\_size inclusive. Then the ranges will start from the power of 2.  Default: 16 |
| min | Min time value (microseconds)  Default: 1 |
| max | Max time value (microseconds)  Default: 16 \* 1024 \* 1024 |
| n\_bins | Number of bins  Default: 24 |
| type | Scale type  Can be "log" or "linear"  Default: log |

## [DEBUG] Section

|  |  |
| --- | --- |
| Value | Description |
| diag\_post\_level | The level of messages which will be in the log file |
| psg\_allow\_io\_test | If set to true then psg will respond to TEST/io URL sending back up to 1000000000 bytes |

## [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)  Basically it is how many blob chunks are requested simultaneously per request. For example, if a blob with 100 chunks is requested then 4 select statements will be created and maintained simultaneously.  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 |
| 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. |
| cassandralog | 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

## Protocol Diagrams

The diagrams were generated using an online tool at <https://bottlecaps.de/rr/ui>

Here is the grammar text:

PSGProtocol ::= PSGchunk+ PSGFinalChunk

ChunkPrefix ::= '\n\nPSG-Reply-Chunk: '

BlobId ::= Integer '.' Integer

PSGChunk ::= (BioseqInfoChunk | BioseqInfoFinalChunk | BlobPropChunk |

BlobPropFinalChunk | BlobChunk | BlobFinalChunk |

BlobExcludeChunk | MessageChunk | BioseqNAChunk |

BioseqNAFinalChunk)

BioseqInfoChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=bioseq\_info'

'&chunk\_type=data' '&size=' Integer '&fmt='

('json' | 'protobuf') '\n' Data '\n'

BioseqInfoFinalChunk ::= ChunkPrefix 'item\_id=' Integer

'&item\_type=bioseq\_info' '&chunk\_type=meta'

'&n\_chunks=' Integer '\n'

BlobPropChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=blob\_prop'

'&chunk\_type=data' '&size=' Integer '&blob\_id=' BlobId '\n'

Data '\n'

BlobPropFinalChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=blob\_prop'

'&chunk\_type=meta' '&n\_chunks=' Integer '\n'

BlobChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=blob'

'&chunk\_type=data' '&size=' Integer '&blob\_id=' BlobId

'&blob\_chunk=' Integer '\n' Data '\n'

BlobFinalChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=blob'

'&chunk\_type=meta' '&blob\_id=' BlobId '&n\_chunks='

Integer '\n'

BlobExcludeChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=blob'

'&chunk\_type=meta' '&blob\_id=' BlobId '&n\_chunks='

Integer '&reason='

('excluded' | 'inprogress' | 'sent') '\n'

MessageChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type='

('blob' | 'bioseq\_info' | 'blob\_prop' | 'reply' |

'bioseq\_na')

'&chunk\_type=message' '&size=' Integer ('&blob\_id=' BlobId)?

'&status=' Integer '&code=' Integer '&severity='

('trace' | 'info' | 'warning' | 'error' | 'critical' |

'fatal') '\n' Message '\n'

BioseqNAChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=bioseq\_na'

'&chunk\_type=data' '&size=' Integer '&na=' String

'&seq\_acc=' String '&seq\_ver=' Integer '&seq\_type='

Integer '\n' Data '\n'

BioseqNAFinalChunk ::= ChunkPrefix 'item\_id=' Integer '&item\_type=bioseq\_na'

'&chunk\_type=meta' '&n\_chunks=' Integer '\n'

PSGFinalChunk ::= ChunkPrefix 'item\_id=0' '&item\_type=reply'

'&chunk\_type=meta' '&n\_chunks=' Integer '\n'

GetBlobOKResponse ::= BlobPropChunk BlobPropFinalChunk

(BlobChunk+ BlobFinalChunk)+ PSGFinalChunk

GetOKResponse ::= BioseqInfoChunk BioseqInfoFinalChunk BlobPropChunk

BlobPropFinalChunk (BlobChunk+ BlobFinalChunk)+

PSGFinalChunk

ResolveOKResponse ::= BioseqInfoChunk BioseqInfoFinalChunk PSGFinalChunk

GetNAOKResponse ::= (BioseqNAChunk BioseqNAFinalChunk)\* PSGFinalChunk

GetTSEChunkOKResponse ::= BlobPropChunk BlobPropFinalChunk

BlobChunk BlobFinalChunk PSGFinalChunk

UnknownURLResponse ::= MessageChunk PSGFinalChunk

## GetBlob Diagram

