PASSWORDRBL

API GUIDE

API VERSION 3.30

Table of Contents

Summary

This guide describes the use of Password RBL's bad password blacklisting API. The service is provided via a RESTful API over secure HTTPS transport. Customers have a choice of two hashing algorithms that can be used to securely submit pre-hashed passwords to the API using this this web API. This is true when sending blacklist queries to search for a match as well as when managing a custom blacklist's entries. Hashed representations of end-user password choices are searched for existence in the Password RBL curated blacklist or a customer-specific blacklist.

The API only allows HTTPS GET requests. Parameters are passed to the API in the URL string. Refer to the API method call detail later in this guide.

What's New in this Version

This version of the Password RBL API builds upon v3.20, which added the ability for customers to also query the "Pwned Passwords" password blacklist database. Version 3.30 adds a new API call (rpt-getmetrics) to programmatically retrieve the metrics report associated with a provided TrackingID.

Recommendations

When implementing the Password RBL API on your web site or application, it is important to take into consideration all possible scenarios during your software development. Password RBL provides the following recommendations when developing software to use the Password RBL API.

- Backups – Before changing any production code base, it is important to have good, working and tested backups.
- Connectivity – The Password RBL API is a hosted solution located across the Internet and is therefore outside your completed control. It is important to consider scenarios when your software cannot make a successful connection to the API due to any number of unforeseen circumstances (Internet congestion, routing problems, etc.).
- API Responses – You should consider how your software will behave if the API returns an error code, rather than a normally formatted positive or negative result. Also, if you've exceeded your quota of blacklist queries, the API will reject your connection and instead reply with a TCP Reset packet.
- Certificates - Do not "hard code" or "memorize" any certificates or cryptographic keys in use by the API. Password RBL regularly changes certificates/keys.

API Endpoints

The API is available via two production endpoints and one development API endpoint. It is important to understand when to use each endpoint as they provide different API services and have different connectivity requirements.

Production Endpoints

- api.passwordrbl.com
  - This endpoint hosts the Query API that is called during end-user password events
  - This endpoint is firewalled to only allow access from current subscribing systems.
  - Only one method call per HTTP connection is allowed (keepalives are disabled).
- webservice.passwordrbl.com
  - This endpoint hosts API calls used for reporting and to manage custom blacklist entries.
  - This endpoint is not called during end-user password events.
  - This endpoint is firewalled to allow general access via HTTPS so connections can be made from management workstations.
  - Connections are throttled to only allow, on average, two connections per second, per source IP address. Keepalives are disabled.

Development Endpoints

- dev.passwordrbl.com
  - This is a development version of the 'api.passwordrbl.com' endpoint noted above.
  - It has the same connection restrictions as the production Query API endpoint
  - The blacklist available at this endpoint only has a few entries to use for testing code
  - This endpoint is available for free to customers and potential customers so that they can develop their Password RBL API implementation prior to beginning their subscription.
  - Contact your account representative or use the Contact form on the Password RBL website to arrange access and obtain the current DEV API documentation.

Query Process Overview

The Password RBL API only accepts pre-hashed versions of end-user password choices (the API does not accept plaintext passwords). The API provides two methods for querying the blacklist of bad passwords: Query and Prefix-Query. Both methods support two industry-standard hashing algorithms as part of the API call. This section details the hashing algorithm choices as well as a comparison of the two available query methods.

Hashing Algorithms

The Password RBL API supports two industry-standard hashing algorithms: SHA256 and PBKDF2. PBKDF2 is the recommended choice since it uses many rounds of iterative hashing to add security assurances against future attempts to reverse the hash. SHA256 uses a single round of hashing but has widespread compatibility across the industry, languages, platforms, etc. Both algorithms have specific parameter requirements (salt value, encoding format, etc.) in order to be compatible with Password RBL's API. The below sections detail the way to use each algorithm to be compatible with the Password RBL API.

Salt Value

It is important to note that both algorithms utilize a SALT value (defined below). The SALT value below is the salt value you must use. Do not choose your own SALT value or choose a random or changing SALT value.

IMPORTANT: If you do not use this specific SALT value, then every submission to the API will result in a not-listed response.

```
SALT = "fe21a0daadda8301bf69a452963a2747a6c8aab4c016d9506a9af46b5f73a9ca"
```

PBKDF2

This is the recommended algorithm. This algorithm takes a password and SALT value as input and then performs many rounds of iterative hashing using the SHA1 cryptographic hashing algorithm. All parameters of the PBKDF2 algorithm, except the password, must match the parameters below:

Hash function: SHA1
Password: <provided by your customer>
SALT value: <see above>
Rounds: 30,000
Output Size: 20 bytes represented as 40 hex characters [0-9,a-f]

Example: `hashvalue = PBKDF2(sha1, Password, SALT, 30000, 20)`

SHA256

This algorithm is provided for compatibility. The output must be 64 hexadecimal characters and is obtained by appending the clear text password to the salt value (above) and passing the resulting string through the standard SHA256 algorithm.

Example: `hashvalue = SHA256( concatenate(SALT, Password) )`

Query Methods

The Password RBL API supports two methods for querying the blacklists: Query and Prefix-Query. The Query method is the easiest and allows you to use all features of the Query API with a single method call. The Prefix-Query requires more client-side implementation, but provides additional security assurances that it is not possible (even for Password RBL) to ever determine the end-user's password choice from the API submission.

Query

This is the simplest way to use the API. You compute a hash of the password chosen by your end-user and provide that hashvalue to the API. The API returns a Yes/No answer on whether the provided hash exists in the blacklist. With a single API call, you can use all features of the API – metrics tracking (TrackingID), custom blacklists, etc. This query method relies on a high number of hash iterations (30,000) of end-user password choices prior to submission to assure subscribers that reversing the hashes is not feasible.

Prefix-Query

Using the API via prefix-query provides an additional assurance that even Password RBL cannot determine the original password cleartext since the API call only includes the first five [hex] characters of the computed hash value. This API call returns all blacklist hashes that begin with the provided five character prefix. This can be more than 50 hashes. The caller compares all returned hashes with the originally computed hash to determine if the chosen password exists in a blacklist. If you choose to use Prefix-Query and want to track metrics (how many times a blacklisted password was chosen), then you must follow-up with a separate API call (update-metric) in order to increase the count of blacklist matches or misses.

Security Assurance of Query Methods

Password RBL service architecture has callers perform cryptographic hashing of end-user chosen passwords before performing an API call. This is by design. The API does not accept plaintext passwords and Password RBL never attempts to determine the original plaintext from a submitted hash. However, as a user of the Password RBL API, you want to be assured that your submissions to the API cannot be reversed. The below table summarizes your assurances when using the API.

Algorithm / Method	SHA256	PBKDF2 @ 30,000 rounds
Query	Fair	Very Good
Prefix-query	Excellent	Excellent

Querying the Pwned Passwords Database

As of API version 3.20, the Password RBL API provides the ability for customers to query the “Pwned Passwords” blacklist database in addition to Password RBL’s curated blacklist and optional subscriber-specific custom blacklist. The Pwned Passwords blacklist database is a derivative of the “have I been pwned” service maintained by Troy Hunt. While this database is maintained by a third-party and is not as curated as the Password RBL blacklist database, it is very large (over 500 million entries). It includes an occurrence value for each hash in the database, which allows for subscribers to require a password (hash) to cross a specified threshold before considering it “bad enough.” Of course, querying the Pwned Passwords blacklist database is optional.

The Process

Querying the Pwned Passwords database is easily done by computing a standard, un-salted SHA1 hash of the end-user’s chosen password and providing this hash as an additional parameter to your existing queries to the Password RBL API. You provide the entire hashvalue when using the Password RBL Query method or the first 5 characters of the hashvalue when using the Password RBL Prefix-Query method.

Threshold

When using the Query method, you can also optionally provide a “threshold” parameter which requires matching hashes to have occurred in the blacklist at least this many times before a positive match return value is indicated. Entries in Password RBL’s curated blacklist, as well as entries in subscriber-specific custom blacklists have a very high occurrence value associated with them since these blacklists are highly-curated.

When using the Prefix-Query method, you do not specify the desired threshold in the query string. But, the occurrence counts will be returned with all matching hashvalues so you can perform the threshold comparison on the client-side of the API connection.

This Pwned Passwords database has over 500 Million unique permutation entries, but many have only been seen in the wild one time and thus have a occurrence value of 1. There are approximately 2 Million entries that have been seen over 100 times and 12 Million entries that have been seen at least 20 times. The default threshold is 1, so that all password entries in the blacklist are considered. Increasing the threshold value will limit matches to more commonly used bad passwords. Striking the right balance between security and usability is always desirable.

Example Passwords and Hashes

Below you will find a list of example passwords and the correct hashvalues for the algorithms supported by the API. This will be helpful as you develop code that implements the Password RBL API since you must produce the correct hashvalues for the service to function as expected. If your hash function is not producing the correct hash values, none of your queries will match on blacklist queries. Each

example password is followed by the correct PBKDF2, SHA256, and SHA1 (used by pwned passwords) hash values.

```
password1
PBKDF2: 12084FC0C5C6F72E55BF377F9591B81EA47ED308
SHA256: 26B5A9EB9449EE064BAF30D8F3F7DADC8AE88A102245E073186015D52621506F
SHA1:   5BAA61E4C9B93F3F0682250B6CF8331B7EE68FD8

Password
PBKDF2: FDBE01B68456C4D86514A7203FB180D8B6974659
SHA256: 1C26C47CEA12FFE94C2C45FEFBC07F32455476EB391CD59AF1363CAC63FB4CBE
SHA1:   8BE3C943B1609FFFBFC51AAD666D0A04ADF83C9D

Password123
PBKDF2: E6BAC6413C4F8300C025B807D2643E0CEB49AF8E
SHA256: 41CDE472FA5517A8E7AACA74003715CBE91864C01451DE37AA3BB858BDA09589
SHA1:   B2E98AD6F6EB8508DD6A14CFA704BAD7F05F6FB1

Pa$$w0rd
PBKDF2: D3CC91EEEF6E5553D6402C9D779C029C2991AC21
SHA256: 290DD9EF4FB0F260DE2BE0B2D38E2CDA1780D0A17144C101AF64B48C5B3F0B75
SHA1:   02726D40F378E716981C4321D60BA3A325ED6A4C

Pa$$w0rd123
PBKDF2: D7DC734F67B0399C61F667D578540FE5D21507EF
SHA256: EB8BEBC98BEE80058BAD61200752DC8EF8E969509F0BDE9A8A4601BE2F75BA31
SHA1:   20AB262F7B7286E33525711FFDC42B10244C1A98

Password123456789!
PBKDF2: 111C5F7CD576F1C239D7C1884A91084636E972B0
SHA256: E6F845AD03506188034E48D9DE7195D84F83B9C0C16EAF59DDC091913FF4F08B
SHA1:   AD793F63DA84E1E9EF3845DEF7E7ED219F4CB1A5
```

Query API Method Specifications

Method: query

Description

This is the most simple blacklist query method. All functions of the Query API can be performed with a single call. Using this method entails computing a hashvalue and providing that hashvalue in the API call. You can optionally search a custom blacklist and also tag queries with a TrackingID for later reporting. This method returns a match or no-match result and will track these metrics if a TrackingID has been provided.

GET request syntax

https://api.passwordrbl.com/query.php?[*required_param*]&[*optional_params*]

Parameter Listing

Parameter	Required	Format / Value	Default
hashvalue	Yes	40 or 64 hex characters [0-9,a-f]	n/a
trackingid	No	32 hex characters [0-9,a-f]	n/a
blacklistid	No	32 hex characters [0-9,a-f]	n/a
cblonly	No	true | false	false
apitype	No	string | xml | json	string
pphashvalue	No	40 hex characters [0-9,a-f]	non
threshold	No	Integer	1

Required Parameter :: hashvalue

This is the only required parameter and is a salted and pre-hashed representation of the password submitted by your customer to your server. There are two industry-standard hashing algorithms to choose from, PBKDF2 or SHA256. It is not necessary to identify which algorithm was chosen when submitting queries to the API as each produce a different length value. PBKDF2 is preferred due to its inherent strength against brute force password-cracking attacks, so much so that it effectively makes it infeasible for anyone to reverse (“crack”) the hashed value back to the original plaintext. SHA256 is provided for compatibility with systems that cannot perform the PBKDF2 algorithm. Refer to the prior section on hashing algorithms for specific usage information.

Optional Parameter :: trackingid

This is an optional parameter. The expected format is 32 hex characters. Queries to the Password RBL service are anonymous by default, but this prevents the service from providing customers with hit rate metrics. The customer can always perform the tracking of metrics on their own server/site. If you would like Password RBL to count queries to the API and how often each query results in a database match or not, you can supply a trackingID with each query. This allows for later reporting of these metrics using our metrics API or our online MyMetrics webpage.

Optional Parameter :: blacklistid

This is an optional parameter. The expected format is 32 hex characters. Queries that supply a blacklistID will search for a match in the identified custom blacklist. If a match is found, a positive response will be send back to the source. If a match is not found in the custom blacklist, then the API continues on to search for a match in the Password RBL curated password blacklist in the same manner that would be taken if the query did not include the blacklistID parameter.

As of version 2.1, if a query includes a blacklistID and a trackingID, then metrics will be tracked in aggregate on the trackingID and metrics will also be tracked on the blacklistID, too. You can then use the MyMetrics page to receive a report for the trackingID and the blacklistID.

IMPORTANT: If a blacklistID is specified but a trackingID is not, then metrics will not be tracked for the custom blacklist. A trackingID must be specified in order to track metrics on the custom blacklist.

Optional Parameter :: cblonly

This is an optional parameter. The expected format is either “true” or “false” and the default value is false. By default, queries that supply a blacklistID will search for a match in the specified custom blacklist and in the Password RBL curated blacklist. Set this optional parameter to “true” and the API will only search for a match in the specified custom blacklist.

If the option is set to “false” or if this option is omitted, then the default behavior will occur.

If this option is set to “true” but a custom blacklist is not specified, an error is returned.

Optional Parameter :: apitype

This parameter designates what format you prefer to receive responses in. The default is String-format but XML and JSON formats are also available. Response values are listed in the following table:

Parameter	Default	Response Format
String	Yes	A single integer [0 | 1] to indicate existence in the Password RBL database or a negative value to indicate an error in the submission to the API.

XML	No	Content-type: text/xml <?xml version="1.0" encoding="utf-8" ?> <xmlresponse> <returnint> [null | 0 | 1] </returnint> <returnbool> [null | “true” | “false”] </returnbool> <error_code> [null | 0 | negative integer value] </error_code> <error_text> [text explanation of error] </error_text> </xmlresponse> NOTE: If the submission is of valid syntax, the values of the <returnint> and <returnbool> tags will have corresponding values to indicate existence of the submitted value in the Password RBL database. The values of the <error_code> and <error_text> tags will be null. If the submission is of invalid syntax, the <returnint> and <returnbool> tags will be null and the <error_code> and <error_text> tags will be filled with values that indicate the type of error.

JSON	No	Content-type: application/json
		{
		"jsonresponse":{
		"returnint": [null | 0 | 1] ,
		"returnbool": [null | “true” | “false”] ,
		"error_code": [null | 0 | negative integer value] ,
		"error_text": [null | text explanation of error]
		}
		}
		
		NOTE: If the submission is of valid syntax, the values of the “returnint” and “returnbool” tags will have corresponding values to indicate existence of the submitted value in the Password RBL database. The values of the “error_code” and “error_text” tags will be null.
		
		If the submission is of invalid syntax, the “returnint” and “returnbool” tags will be null and the “error_code” and “error_text” tags will be filled with values that indicate the type of error.

Optional Parameter :: pphashvalue

This is an optional parameter. The expected format is 40 hexadecimal characters [0-9,a-f]. This parameter is used to provide the SHA1 hash (of the end-user’s password) to be used in querying the pwned password blacklist database.

Optional Parameter :: threshold

This is an optional parameter. The expected format is any valid 32-bit integer number. This parameter represents the number of times a password must have been seen any of the password blacklists before a positive match response is received.

Method: prefix-query

Description

Use this query method to perform a blacklist query by only sending a partial hash value – a prefix of the computed hashvalue to be searched for in the curated and/or custom blacklist. This method returns all hashes that begin with the provided prefix string.

Each hash returned will have a number of occurrences associated with it. This number represents how many times the password represented by this hash has been discovered. The higher the number, the worse of a choice the associated password is. Currently, hash entries in the Password RBL curated blacklist as well as any entries in a custom blacklist return an administratively set high number (99999). This feature is reserved for use in a future API version.

This query mechanism provides an additional assurance that Password RBL can never obtain the original password choice of an end-user. However, if you wish to use a TrackingID to track metrics, you must use a second API call (see below) in order to increase the counts of your chosen TrackingID (and/or BlacklistID) since the API cannot know if any of the returned hashvalues match the hash originally computed by the caller.

GET request syntax

https://api.passwordrbl.com/prefix-query.php?[*required_param*]&[*optional_params*]

Parameter Listing

Parameter	Required	Format / Value	Default
hashprefix	Yes	5 hex characters [0-9,a-f]	n/a
hashtype	Yes	pbkdf2 | sha256	n/a
pphashprefix	No	5 hex characters [0-9,a-f]	none
apitype	No	string | xml | json	string
blacklistid	No	32 hex characters [0-9,a-f]	n/a
cblonly	No	true | false	false
eol	No	crlf | lf | cr | br	crlf
pphashprefix	No	5 hex characters [0-9,a-f]	none

Required Parameter :: hashprefix

This is a required parameter and is the first five (5) hex characters of the salted and pre-hashed representation of the password. There are two industry-standard hashing algorithms to choose from,

PBKDF2 or SHA256. Since the prefix length is the same regardless of algorithm used, you must also specify the hashtype parameter to state which algorithm was used to compute this hashprefix. Because you are not providing the complete hash to Password RBL, you have assurances that it is impossible for anyone, including Password RBL, to determine the end-user's chosen password. Refer to the prior section on hashing algorithms for specific usage information.

Required Parameter :: hashtype

This is a required parameter and the expected value is either "sha256" or "pbkdf2". This parameter informs the API which algorithm you used to compute the hash prefix. This is necessary since the API requires the submitted hash prefix to be five (5) hex characters regardless of the algorithm used.

Optional Parameter :: eol

This parameter designates what character(s) you prefer to use as end of line characters. This parameter only affects string-based response types. If XML or JSON response types are used, those specifications dictate which end of line character are used. End of line character options are listed in the following table:

Value	Default	Response Format
crlf	Yes	Each line will end with a carriage return and line feed characters commonly denoted as "\r\n". This line ending is popular on the Windows platform.
cr	No	Each line will end with only a carriage return character, "\r", which is common on the Mac platform.
lf	No	Each line will end with a single line feed character, "\n", which is common on Unix and Linux-based systems.
br	No	Each line will end with an HTML-style "break" – which is a collection of four characters, " ".

Optional Parameter :: apitype

This parameter designates what format you prefer to receive responses in. Currently, string, XML and JSON formatting is available. Response values and examples are listed in the following table:

Parameter	Default	Response Format
String	Yes	A list of complete hash values that begin with the same 5 characters provided in the API call, followed by a colon (:) character, and an integer representing the number of times this password has been discovered. If an error is being returned, a textual description of the error is provided as well as a negative numeric error code, delimited by a colon (:) character.
XML	No	Content-type: text/xml <?xml version="1.0" encoding="utf-8" ?> <xmlresponse> <summary> <method>prefix-query</method> <response_count> # </response_count> <error_code> # </error_code> <error_text> </error_text> </summary> <response_data> <blacklist_entry> <hash_value>1111111111abcd23436575983123abcd </hash_value> <hash_count>99999</hash_count> </blacklist_entry> <blacklist_entry> <hash_value>2222222222abcd23436575983123abcd </hash_value> <hash_count>99999</hash_count> </blacklist_entry> </response_data> </xmlresponse> NOTE: If the submission is valid syntax, the value of the <response_count> tag represents the number of matching hashes returned and available in the <response_data> element. The value of the <error_code> tag will be zero and <error_text> tag will be null. If the submission is invalid syntax, or an error occurred, the <response_count> tag will be NULL and the <error_code> and <error_text> tags will be filled with values that indicate the type of error.

JSON	No	Content-type: application/json
		
		{
		"jsonresponse":{
		"summary":{
		"method": "prefix-query",
		"response_count": #,
		"error_code": 0,
		"error_text": ""
		},
		"response_data": [
		{
		"hash_value": "1111111111abcd23436575983123abcd",
		"hash_count": 99999
		},
		{
		"hash_value": "2222222222abcd23436575983123abcd",
		"hash_count": 99999
		}
		]
		}
		}
		
		IMPORTANT: The “response_data” element is a JSON array of objects.
		
		If the submission is valid syntax, the value of the “response_count” key represents the number of matching hashes returned and available in the “response_data” array. The value of the “error_code” key will be zero and “error_text” key will be empty.
		
		If the submission is invalid syntax, or an error occurred, the “response_count” key will be NULL and the “error_code” and “error_text” keys will be filled with values that indicate the type of error.

Optional Parameter :: pphashprefix

This is an optional parameter. The expected format is 5 hexadecimal characters [0-9,a-f]. This parameter is used to provide the first 5 characters of the SHA1 hash (of the end-user’s password) to be used in querying the pwned password blacklist database.

Method: update-metric

Description

Use this method to update metrics (count) for a specified TrackingID. This method commonly follows a call to the prefix-query method in order to update metrics associated with a subscriber's TrackingID.

GET request syntax

https://api.passwordrbl.com/update-metric.php?[*required_param*]&[*optional_params*]

Parameter Listing

Parameter	Required	Format / Value	Default
metric	Yes	hit | miss	n/a
trackingid	Yes	32 hex characters [0-9,a-f]	n/a
blacklistid	No	32 hex characters [0-9,a-f]	n/a
apitype	No	string | xml | json	string

Required Parameter :: metric

This is a required parameter and the expected value is either "hit" or "miss." This parameter specifies which count is incremented: the count of blacklist query matches (hit) or the count of queries that resulted in a no-match (miss). Each time this method is called, the associated metric is increased by one.

Required Parameter :: trackingid

This is a required parameter. The expected format is 32 hex characters. Calling this method with a specified TrackingID will increase the metric type count of the specified TrackingID as well as aggregate metrics. Keeping metrics accurate allows for later reporting of these metrics using the MyMetrics page of the Password RBL website.

Optional Parameter :: blacklistid

This is an optional parameter. The expected format is 32 hex characters. Calls that supply a BlacklistID will increase the metrics (counts) associated with this BlacklistID. It is important to note, that in order to update custom blacklist metrics, the call must also include a TrackingID. An error is returned if a BlacklistID is provided but a TrackingID is not.

Optional Parameter :: apitype

This parameter designates what format you prefer to receive responses in. The default is String-format but XML and JSON formats are also available. Response values are listed in the following table:

Parameter	Default	Response Format
String	Yes	A single integer [0 | 1] to indicate that the metric update process was successful or not. A negative value is returned to indicate an error in the submission to the API.
XML	No	Content-type: text/xml <?xml version="1.0" encoding="utf-8" ?> <xmlresponse> <returnint> [null | 0 | 1] </returnint> <returnbool> [null | “true” | “false”] </returnbool> <error_code> [null | 0 | negative integer value] </error_code> <error_text> [text explanation of error] </error_text> </xmlresponse> NOTE: If the submission is of valid syntax, the values of the <returnint> and <returnbool> tags will have corresponding values to indicate success or failure of the metric update process. The values of the <error_code> and <error_text> tags will be null. If the submission is of invalid syntax, or an error occurs, the <returnint> tag will have a negative value, <returnbool> tag will be “error” and the <error_code> and <error_text> tags will be filled with values that indicate the type of error.

JSON	No	Content-type: application/json
		{
		"jsonresponse":{
		"returnint": [-1 | 0 | 1] ,
		"returnbool": [“true” | “false” | “error”] ,
		"error_code": [null | 0 | negative integer value] ,
		"error_text": [empty string | text explanation of error]
		}
		}
		
		NOTE: If the submission is of valid syntax, the values of the “returnint” and “returnbool” tags will have corresponding values to indicate success or failure of the metric update process. The values of the “error_code” and “error_text” tags will be null.
		
		If the submission is of invalid syntax, or an error occurred, the “returnint” will have a negative value, “returnbool” will be set to “error”, and the “error_code” and “error_text” tags will be filled with values that indicate the type of error.

Webservice API Method Specifications

Method: cbl-management

Description

Use this method call to manage the hash value entries in your custom blacklist.

GET request syntax

https://webservice.passwordrbl.com/cbl-management.php?[*required_param*]&[*optional_params*]

Parameter Listing

Parameter	Required	Format / Value	Default
action	Yes	quota | count | add | delete | empty	n/a
blacklistid	Yes	32 hex characters [0-9,a-f]	n/a
hashvalue	Depends on action	40 or 64 hex characters [0-9,a-f]	n/a

Required Parameter :: action

The action parameter is always required and can be one of the following: quota, count, add, delete, or empty. Each action type instructs the API to perform a specified action against the custom blacklist identified in the request. See below for a detailed explanation of each action type.

Quota

Submitting a query to the custom blacklist management method with the action type set to quota will return the maximum number of blacklist entries that are allowed for the specified custom blacklist. If you wish to increase the current quota assigned to your custom blacklist, simply use the contact form on the main website. Additional subscription fees may apply.

The quota action returns a positive integer that represents the maximum number of custom blacklist entries or a negative number to indicate error.

Count

Submitting a query to the custom blacklist management method with the action type set to count will return the current number of blacklist entries in your custom blacklist.

It is important to note that Password RBL supports multiple different hashing types with custom blacklists. Currently, PBKDF2 and SHA256 are supported, but others may be added in the future. It is only necessary to populate the custom blacklist with the hash type you will use. However, it is not detrimental to populate the custom blacklist with hashes of a type that you do not query. Therefore, Password RBL always recommends keeping the population of the hash types exactly the same. If you use the provided custom blacklist management tool, it populates both hash types, by default.

The count action will always return the maximum number of entries across all hash types.

For example, if you have 50 entries of type PBKDF2 and 100 entries of type SHA256, the count action will return 100.

Add

The add action adds the provided hashvalue to the custom blacklist.

The add action returns 1 if the add was successful, 0 if the add was unnecessary (syntactically correct but the entry was already in the blacklist), and a negative number to indicate error.

Delete

The delete action removes the provided hashvalue from the custom blacklist.

The delete action returns 1 if the removal was successful, 0 if the removal was unnecessary (syntactically correct but the entry was not found in the blacklist), and a negative number to indicate error.

Empty

The empty action removes all hash values of all types from the custom blacklist identified by the accompanied blacklistID in a single request.

The empty action returns the number of entries that were removed from the custom blacklist if the removal was successful, 0 if the removal was not successful, and a negative number to indicate error.

Required Parameter :: blacklistid

This parameter is always required. The expected format is 32 hex characters. This parameter identifies which custom blacklist in the Password RBL system is to be operated upon by the current request.

Parameter :: hashvalue

This parameter represents the hashed password that you will either add or remove from your custom blacklist. The hashvalue parameter is therefore required when the requested action is either add or delete, but is unused (and ignored if provided) when the action type is quota, count, or empty. The expected format is 40 or 64 hex characters, depending on the hash function that was utilized to produce the hash. PBKDF2 should output 40 hex characters whereas SHA256 produces 64 hex characters. The method for producing the hashvalue for use by custom blacklists is exactly the same as producing the hashvalues for use by the Query API call. Refer to the prior section on hashing algorithms for specific usage information.

Method: rpt-getmetrics

Description

Use this method call to retrieve a metrics report for a provided trackingID. Metrics are recorded on a daily basis. The API call will return a row or element for any day that queries were made. If no queries were made on a day, nothing is returned for that day. If there are gaps in the results, this is because no queries were made that day. The metrics are available in multiple formats.

GET request syntax

https://webservice.passwordrbl.com/rpt-getmetrics.php?[*required_param*]&[*optional_params*]

Parameter Listing

Parameter	Required	Format / Value	Default
trackingid	Yes	32 hex characters [0-9,a-f]	n/a
apitype	No	string | xml | json | csvfile	string
eol	No	crlf | lf | cr | br	crlf

Required Parameter :: trackingid

This is a required parameter. The expected format is 32 hex characters. Use this parameter to specify the trackingID that you would like to receive a metrics for.

Optional Parameter :: apitype

This parameter designates what format you prefer to receive responses in. The default is String-format, which returns results in csv format as a direct response. XML and JSON formats are also available. Use csvfile to retrieve the results as a separate file rather than direct response. Response values are listed in the following table:

Parameter	Default	Response Format
string	Yes	CSV-formatted direct response as noted below. There is a separate row for each day that query data is available. If no rows are returned, then the specified trackingID has no data. If an error occurs, a text description of the error is returned, followed by a comma and a negative value. Content-type: text/plain date,hits,misses,total 2019-01-23,1,2,3
XML	No	Content-type: text/xml <?xml version="1.0" encoding="utf-8" ?> <xmlresponse> <summary> <method>rpt-getmetrics</method> <response_count> # </response_count> <error_code> # </error_code> <error_text> </error_text> </summary> <response_data> <metric_entry> <date> YYYY-MM-DD </date> <hits> # </hits> <misses> # </misses> <total> # </total> </blacklist_entry> </response_data> </xmlresponse> NOTE: If the submission is valid syntax, the value of the <response_count> tag represents the number of matching metric entries returned and available in the <response_data> element. The value of the <error_code> tag will be zero and <error_text> tag will be null. If the submission is invalid syntax, or an error occurred, the <response_count> tag will be NULL and the <error_code> and <error_text> tags will be filled with values that indicate the type of error.

JSON	No	Content-type: application/json
		{
		"jsonresponse": {
		"summary": {
		"method": "rpt-getmetrics",
		"response_count": #,
		"error_code": #,
		"error_text": ""
		},
		"response_data": [
		{
		"date": "YYYY-MM-DD",
		"hits": #,
		"misses": #,
		"total": #
		}
		]
		}
		}
		
		NOTE: If the submission is of valid syntax, the values of the "response_count" represents the number of matching metric entries returned and available in the response_data array. The values of the "error_code" and "error_text" tags will be null.
		
		If the submission is of invalid syntax, "response_count" will be NULL and the "error_code" and "error_text" elements will be filled with values that indicate the type of error.

csvfile	No	This is the same, CSV-formatted output that the String type returns, but this will produce the output as a file, rather than as a direct API response. The syntax is noted below. If the file contains no data rows, this indicates no data is available for this trackingID. A negative value and descriptive text indicates an error in the submission to the API. Content-type: text/plain Content-Disposition: attachment date,hits,misses,total 2019-01-23,1,2,3

Optional Parameter :: eol

This parameter designates what character(s) you prefer to use as end of line characters. This parameter only have an effect when the APITYPE parameter is "string" or "csvfile." If XML or JSON response types are used, those specifications dictate which end of line character is used. End of line character options are listed in the following table:

Value	Default	Response Format
crlf	Yes	Each line will end with a carriage return and line feed characters commonly denoted as "\r\n". This line ending is popular on the Windows platform.
cr	No	Each line will end with only a carriage return character, "\r", which is common on the Mac platform.
lf	No	Each line will end with a single line feed character, "\n", which is common on Unix and Linux-based systems.
br	No	Each line will end with an HTML-style "break" – which is a collection of four characters, " ".

Error Code Listing

Below is a listing of all the error codes that can be returned from the Password RBL API. This reference is especially helpful if you use the default string type of API response from the query method, since those return messages only include the code and not the added explanation.

Error Code	Explanation
-410	Required parameter 'hashvalue' was not provided or was empty *The HTTPS GET request did not include the required URL parameter 'hashvalue' or the parameter was specified but had a null value.*
-411	Invalid format of HTTP parameter 'hashvalue' *The HTTPS GET request contains the required parameter 'hashvalue' but it was not the correct length or included non-hex characters.*
-412	Invalid format of HTTP parameter 'apitype' *The HTTPS GET request contained the optional parameter 'apitype' but the specified value was not one of the options specified by the API.*
-413	Invalid length of HTTP parameter 'trackingid' *The HTTPS GET request contained the optional parameter 'trackingid' but the specified value was not the correct length specified by the API.*
-414	Invalid format of HTTP parameter 'trackingid' *The HTTPS GET request contained the optional parameter 'trackingid' but the specified value contained non-hex characters.*
-415	Invalid length of HTTP parameter 'blacklistid' *The HTTPS GET request contained the optional parameter 'blacklistid' but the specified value was not the correct length specified by the API.*
-416	Invalid format of HTTP parameter 'blacklistid' *The HTTPS GET request contained the optional parameter 'blacklistid' but the specified value contained non-hex characters.*
-417	Invalid length of HTTP parameter 'cblonly' *The HTTPS GET request contained the optional parameter 'cblonly' but the specified value was not the correct length specified by the API.*
-418	Invalid format of HTTP parameter 'cblonly' *The HTTPS GET request contained the optional parameter 'cblonly' but the specified value was not equal to 'true' or 'false'*
-419	The parameter 'cblonly' was specified but 'blacklistid' was not. *The HTTPS GET request contained the optional parameter 'cblonly' but this option requires that a custom blacklist ID also be included in the same HTTPS GET request*
-421	The supplied 'trackingid' is not a valid ID but the format is valid *The HTTPS GET request contained the optional parameter 'trackingid' with valid syntax, but the specified value is not a valid Tracking ID.*

-422	The supplied ‘blacklistID’ is not a valid ID but the format is valid *The HTTPS GET request contained the optional parameter ‘blacklistid’ with valid syntax, but the specified value is not a valid Blacklist ID.*
-423	Required parameter ‘hashtype’ was not provided or was empty *The HTTPS GET request did not include the required URL parameter ‘hashtype’ or the parameter was specified but had a null value.*
-424	Invalid length of HTTP parameter ‘hashtype’ *The HTTPS GET request contained the parameter ‘hashtype’ but the provided value was not the correct length.*
-425	Invalid format of HTTP parameter ‘hashtype’ *The HTTPS GET request contained the parameter ‘hashtype’ but the provided value was not one of the valid options.*
-426	Invalid length of HTTP parameter ‘eol’ *The HTTPS GET request contained the parameter ‘eol’ but the provided value was not the correct length.*
-427	Invalid format of HTTP parameter ‘eol’ *The HTTPS GET request contained the parameter ‘eol’ but the provided value was not one of the valid options.*
-428	Invalid length of HTTP parameter ‘pphashvalue’ *The HTTPS GET request contained the parameter ‘pphashvalue’ but the provided value was not the correct length.*
-429	Invalid format of HTTP parameter ‘pphashvalue *The HTTPS GET request contained the optional parameter ‘’pphashvalue’ but the specified value contained non-hex characters.*
-430	Invalid format of HTTP parameter ‘threshold’ *The HTTPS GET request contained the optional parameter ‘threshold’ but the provided value was not a valid integer.*
-432	Invalid length of HTTP parameter ‘pphashprefix’ *The HTTPS GET request contained the parameter ‘pphashprefix’ but the provided value was not the correct length.*
-433	Invalid format of HTTP parameter ‘pphashprefix’ *The HTTPS GET request contained the optional parameter ‘’pphashprefix’ but the specified value contained non-hex characters.*
-451	Required parameter ‘action’ was not provided or was empty *The HTTPS GET request did not include the required parameter ‘action’ or the parameter was specified but had a null value.*
-452	Invalid format of HTTP parameter ‘action’ *The HTTPS GET request contained the required parameter ‘action’ but the specified value was not one of the options specified by the API*

-453	Required parameter ‘blacklistid’ was not provided or was empty *The HTTPS GET request did not include the required parameter ‘blacklistid’ or the parameter was specified but had a null value.*
-454	Invalid length of HTTP parameter ‘blacklistid’ *The HTTPS GET request contained the required parameter ‘blacklistid’ but the specified value was not the correct length specified by the API.*
-455	Invalid format of HTTP parameter ‘blacklistid’ *The HTTPS GET request contained the parameter ‘blacklistid’ but the specified value contained non-hex characters.*
-456	The supplied ‘blacklistid’ is not a valid ID but the format is valid *The HTTPS GET request contained the optional parameter ‘blacklistid’ with valid syntax, but the specified value is not a valid Blacklist ID.*
-457	There was an error executing the add command *This is a generic error that arises during the beginning processing of the XXX command, or if there is unexpected data returned from the backend database. If you receive this error, please contact us using the form on the website so we can look into this.*
-458	There was an error executing the add command *This error occurs if the hashvalue could not be added to the database. This would occur if communication was interrupted midstream or a configuration error. Please contact us if you receive this error.*
-459	Blacklist entry quota exceeded *This error occurs if the custom blacklist has reached its maximum number of entries and an adding an additional entry was attempted. Contact us if you would like to increase your quota.*
-460	There was an error executing the delete command *This error occurs if a database deletion was unsuccessful. If this error persists, please contact us.*
-461	There was an error executing the empty command for pbkdf2 entries. *This error database command to delete all pbkdf2 blacklist entries fails. If this error persists, please contact us.*
-462	There was an error executing the delete command for sha256 entries. *This error database command to delete all sha256 blacklist entries fails. If this error persists, please contact us.*
-470	Required parameter ‘trackingid was not provided or was empty *The HTTPS GET request did not include the required URL parameter ‘trackingid or the parameter was specified but had a null value.*
-501	Unable to connect to database *This is an internal error and occurs if the initial database connection fails.*

-502	Unable to connect to database *This is an internal error and occurs if the database connection fails during API processing.*
-510	Encountered a problem initializing connection to pwnedpasswords API *This is an internal error and occurs if the https connection fails during initialization phase.*
-511	Encountered a problem connecting to pwnedpasswords API *This is an internal error and occurs if the https connection to the pwnedpasswords API fails during processing.*

API Version History

Version	Notable Changes
Current Version	Added new rpt-getmetrics API call for accessing metrics data Non-critical parsing bugfix
3.20	Added option for additionally querying PwnedPasswords blacklist Non customer-facing code improvements
3.10	Added XML and JSON formatted responses to prefix-query and update-metric method calls. Add whitespace formatting to XML and JSON responses to all query API methods
3.00	Add prefix-query method to allow queries with only a partial hash value provided Add update-metric method to allow updating metrics without query
2.20	Added ‘cblonly’ parameter to control which blacklists to search when also using a Custom Blacklist.
2.10	Added metrics tracking on Custom Blacklists when a trackingID is also specified
2.00	Added Custom Blacklists and new “webservice” API Endpoint
1.60	Change default hashing algorithm to PBKDF2; SHA256 still supported for backwards compatibility
1.50	Change API parameter ‘sourceID’ to ‘trackingID’ to unify naming across offerings
1.41	Enhanced error messages with clear language
1.40	Enhanced verification and error reporting associated with ‘sourceID’ / ‘trackingID’ parameter
1.31	Enhanced JSON formatting output
1.30	Added JSON formatted API responses
1.20	Added ‘sourceID’ parameter in order to track metrics if customers decide to provide an optional tracking identifier to their queries.
1.10	Maintenance release; no notable customer-facing changes.
1.00	Original version