SSL Labs API Documentation: v1.11.8

Last update: 30 October 2014

Author: Ivan Ristic <iristic@qualys.com>

Protocol Overview

This document explains the SSL Labs Assessment APIs, which can be used to test SSL servers available on the public Internet.

We are fine-tuning this API at the moment; small changes should be expected. The following changes are planned:

- **Expose all certificate chains.**
- **Don't allow clearCache for a short time (e.g., 30 seconds) after an assessment is complete.**
- **Add field `status` to Endpoint.**
- **Add a `notice` field to the Info object.**

The protocol is based on HTTP and JSON. All invocations of the API should use the GET method and specify the parameters in the query string, as documented below. The results will be returned in the response body as a JSON payload.

Do not hardcode SSL Labs IP addresses

We're constantly working to improve SSL Labs, and that often means that the underlying IP addresses change. For best results, do not hardcode SSL Labs IP addresses in your code.

The best way to use SSL Labs is to resolve our hostname once for every test, then continue to use the same IP address for that particular test. This approach will ensure consistent responses even if we use a DNS RR deployment with short TTL times. That said, the next version of SSL Labs operates as a cluster; this generally means that you should get consistent results no matter which node you hit.

But please do ensure that you have enabled session caching in your client/library. Our load balancer will always send the same SSL session to the same node.

Terms and Conditions

Our standard (pre-API) terms and conditions apply: https://www.ssllabs.com/about/terms.html As the APIs get stable, we will merge the two documents into one. When in doubt, the terms specified here override the terms currently on the SSL Labs web site.

We are making the SSL Labs APIs available to help everyone automate batch testing of their own infrastructure. Our terms and conditions are designed to help us make the best use of our limited resources without impeding our goals.

Allowed uses:

- Testing of your own infrastructure
- Integration with non-commercial open source tools
- Non-commercial security research, provided that the results are made freely available online and don't disclose information on individual web sites

We are also interested in collaborating with infrastructure providers, such as hosting companies, certification authorities, and content delivery networks. We are happy to give you limited permission to use our APIs to check the security of your customers' networks. We generally expect that only report summaries will be shown to your customers and that they will be sent to the SSL Labs web site for the full results.

We regret that other uses are not allowed. In particular, but not limited to, the following uses are not allowed:

- Use of the APIs on a web site
- Commercial use (either direct or indirect commercialization, either as part of a product or a service)

Notes:

- The names Qualys and SSL Labs must not be part of the tool name, or otherwise used to promote the tool
- The tool documentation should include a notice to inform the user about the use of the SSL Labs API. It is particularly important for the users to understand that the assessments will be performed by the SSL Labs servers and the reports stored (cached) for a period of time. SSL Labs logs submitted hostnames but only as part as normal activity logging. The hostnames are not used otherwise.
- Each tool invocation should output the notice returned by the SSL Labs `info` API call (documented later in this text).
- Individual site reports should provide links to the report on the SSL Labs web site.

Protocol Calls

This section documents the available protocol calls.

Check SSL Labs availability

This call should be used to check the availability of the SSL Labs servers, retrieve the engine and criteria version, and initialize the maximum number of concurrent assessments. Returns one *Info* object on success.

URL: `info`

Parameters:

- None.

Invoke assessment and check progress

This call is used to initiate an assessment, or to retrieve the status of an assessment in progress or in the cache. It will return a single *Host* object on success. The *Endpoint* object embedded in the *Host* object will provide partial endpoint results.

API Call: `analyze`

Parameters:

- **host** - hostname; required.
- **publish** - set to "on" if assessment results should be published on the public results boards; optional, defaults to "off".
- **clearCache** - if set to "on" then cached assessment results are ignored and a new assessment is started. However, if there's already an assessment in progress, its status is delivered instead. This parameter should be used only once to initiate a new assessment; further invocations should omit it to avoid causing an assessment loop.
- **fromCache** - always deliver *cached* assessment reports if available; optional, defaults to "off". This parameter is intended for API consumers that don't want to wait for assessment results. Can't be used at the same time as the *clearCache* parameter.
- **all** - by default this call results only summaries of individual endpoints. If this parameter is set to "on", full information will be returned. If set to "done", full information will be returned only if the assessment is complete (status is READY or ERROR).

Examples:

- `/analyze?host=www.ssllabs.com`

- `/analyze?host=www.ssllabs.com&publish=on`

Retrieve detailed endpoint information

This call is used to retrieve detailed endpoint information. It will return a single *Endpoint* object on success. The object will contain complete assessment information.

API Call: `getEndpointData`

Parameters:

- **host** - as above
- **s** - endpoint IP address
- **fromCache** - see above.

Example:

- `/getEndpointData?host=www.ssllabs.com&s=173.203.82.166`

Retrieve known status codes

This call will return one *StatusCodes* instance.

API Call: `/getStatusCodes`

Parameters:

- None.

Protocol Usage

When you want to obtain fresh test results for a particular host:

1. Invoke `analyze` with the `clearCache` parameter to `on`. Set `all` to `done`.
2. The assessment is now in progress. Call `analyze` periodically (without the `clearCache` parameter!) until the assessment is finished. You can tell by observing the `Host.status` field for either READY or ERROR values.
3. When there are multiple servers behind one hostname, they will be tested one at a time.
4. During the assessment, interim responses will contain only endpoint status, but not full information.
5. At the end of the assessment, the response will contain all available information; no further API calls will need to be made for that host.

When you're happy to receive cached information (e.g., in a browser add-on):

1. Invoke `analyze` with `fromCache` set to `on` and all set to `done`.
2. If the information you requested is available in the cache, it will be returned straight away.
3. Otherwise, a new assessment will be started.
4. You can continue to call `analyze` periodically until the assessment is complete.

Error Reporting

When an API call is incorrectly invoked, it will cause an error response to be sent back. The response will include an array of error messages. For example:

```
{"errors":[{"field":"host","message":"qp.mandatory"}]}
```

The field value references the API parameter name that has an incorrect value. The message value will tell you what the issue is. It is also possible to receive errors without the field parameter set; such messages will usually refer to the request as a whole.

Error Response Status Codes

The following status codes are used:

- 400 - invocation error (e.g., invalid parameters)
- 429 - client request rate too high
- 500 - internal error
- 503 - the service is not available (e.g., down for maintenance)
- 529 - the service is overloaded

If you get 429, 503, 529, you should sleep for several minutes (e.g., 5, 15, 30 minutes, respectively) then try again. If you're writing an API client tool and get a 529 response, randomize the back-off time. If you get 500, it's best to give up.

Access Rate and Rate Limiting

Please note the following:

- Server assessments usually take at least 60 seconds. (They are intentionally slow, to avoid harming servers.) Thus, there is no need to poll for the results very often. In fact, polling too often slows down the service for everyone. It's best to use variable polling: 5 seconds until an assessment gets under way (status changes to IN_PROGRESS), then

10 seconds until it completes.

- Keep down the number of concurrent assessments to a minimum. If you're not in a hurry, test only one hostname at a time.

We may limit your usage of the API, by enforcing a limit on concurrent assessments, and the overall number of assessments performed in a time period. If that happens, we will respond with 429 (Too Many Requests) to API calls that wish to initiate new assessments. Your ability to follow previously initiated assessments, or retrieve assessment results from the cache, will not be impacted. If you receive a 429 response, reduce the number of concurrent assessments.

If the server is overloaded (a condition that is not a result of the client's behaviour), the 529 status code will be used instead. This is not a situation we wish to be in. If you encounter it, take a break and come after at least 30 minutes of sleep.

All successful API calls contain the response header `X-ClientMaxAssessments`, which contains the maximum allowed number of active assessments for the invoking client. It is recommended that, every time you receive a response, you also update your internal limit.

Protocol Evolution

The API is versioned. New versions of the API will be introduced whenever incompatible changes need to be made to the protocol. When a new version becomes available, existing applications can continue to use the previous version for as long as it is supported.

To reduce version number inflation, new fields may added to the results without a change in protocol version number.

Response Objects

The remainder of the document explains the structure of the returned objects. The following conventions are used:

- **field** - a simple field
- **object{}** - an object
- **array[]** - an array

Host

- **host** - assessment host, which can be a hostname or an IP address
- **port** - assessment port (e.g., 443)

- **protocol** - protocol (e.g., HTTP)
- **isPublic** - true, is this assessment publicly available (listed on the SSL Labs assessment boards)
- **status** - assessment status; possible values: DNS, ERROR, IN_PROGRESS, and READY.
- **statusMessage** - status message in English. When status is ERROR, this field will contain an error message.
- **startTime** - assessment starting time, in milliseconds since 1970
- **testTime** - assessment completion time, in milliseconds since 1970
- **engineVersion** - assessment engine version (e.g., "1.0.120")
- **criteriaVersion** - grading criteria version (e.g., "2009")
- **cacheExpiryTime** - when will the assessment results expire from the cache (typically set only for assessment with errors; otherwise the results stay in the cache for as long as there's sufficient room)
- **endpoints[]** - list of Endpoint objects
- **certHostnames[]** - the list of certificate hostnames collected from the certificates seen during assessment. The hostnames may not be valid.

Endpoint

- **ipAddress** - endpoint IP address, in IPv4 or IPv6 format.
- **serverName** - server name retrieved via reverse DNS
- **statusMessage** - assessment status message
- **statusDetails** - code of the operation currently in progress
- **statusDetailsMessage** - description of the operation currently in progress
- **grade** - possible values: A+, A-F, and T (no trust) and M (certificate name mismatch)
- **hasWarnings** - if this endpoint has warnings that might affect the score (e.g., get A- instead of A).
- **isExceptional** - this flag will be raised when an exceptional configuration is encountered. The SSL Labs test will give such sites an A+.

- **progress** - assessment progress, which is a value from 0 to 100, and -1 if the assessment has not yet started
- **duration** - assessment duration, in milliseconds
- **eta** - estimated time, in seconds, until the completion of the assessment
- **delegation** - indicates domain name delegation with and without the www prefix
    - bit 0 (1) is set for non-prefixed access
    - bit 1 (2) is set for prefixed access

EndpointDetails

- **hostStartTime =** endpoint assessment starting time, in milliseconds since 1970. This field is useful when test results are retrieved in several HTTP invocations. Then, you should check that the *hostStartTime* value matches the *startTime* value of the host.
- **key{}** - key information
- **cert{}** - certificate information
- **chain{}** - chain information
- **protocols[]** - supported protocols
- **suites{}** - supported cipher suites
- **serverSignature** - Contents of the HTTP Server response header when known. This fiel could be absent for one of two reasons: 1) the HTTP request failed (check httpStatusCode) or 2) there was no Server response header returned.
- **prefixDelegation** - true if this endpoint is reachable via a hostname with the www prefix
- **nonPrefixDelegation** (moved here from the summary) - true if this endpoint is reachable via a hostname without the www prefix
- **vulnBeast** - true if the endpoint is vulnerable to the BEAST attack
- **renegSupport** - this is an integer value that describes the endpoint support for renegotiation:
    - bit 0 (1) is set if insecure client-initiated renegotiation is supported
    - bit 1 (2) is set if secure renegotiation is supported
    - bit 2 (4) is set if secure client-initiated renegotiation is supported

    - bit 3 (8) is set if the server requires secure renegotiation support

- **stsResponseHeader** - the contents of the Strict-Transport-Security (STS) response header, if seen
- **stsMaxAge** - the maxAge parameter extracted from the STS parameters; null if STS not seen, or -1 if the specified value is invalid (e.g., not a zero or a positive integer; the maximum value currently supported is 2,147,483,647)
- **stsSubdomains** - true if the includeSubDomains STS parameter is set; null if STS not seen
- **pkpResponseHeader** - the contents of the Public-Key-Pinning response header, if seen
- **sessionResumption** - this is an integer value that describes endpoint support for session resumption. The possible values are:
    - 0 - session resumption is not enabled and we're seeing empty session IDs
    - 1 - endpoint returns session IDs, but sessions are not resumed
    - 2 - session resumption is enabled
- **compressionMethods** - integer value that describes supported compression methods
    - bit 0 is set for DEFLATE
- **supportsNpn** - true if the server supports NPN
- **npnProtocols** - space separated list of supported protocols
- **sessionTickets** - indicates support for Session Tickets
    - bit 0 is set if session tickets are supported
    - bit 1 (not implemented) is set if the implementation is faulty
    - bit 2 is set if the server is intolerant to the extension
- **ocspStapling** - true if OCSP stapling is deployed on the server
- **sniRequired** - if SNI support is required to access the web site.
- **httpStatusCode** - status code of the final HTTP response seen. When submitting HTTP requests, redirections are followed, but only if they lead to the same hostname. If this field is not available, that means the HTTP request failed.
- **httpForwarding** - available on a server that responded with a redirection to some other

hostname.

- **supportsRc4** - true if the server supports at least one RC4 suite.
- **forwardSecrecy** - indicates support for Forward Secrecy
    - bit 0 - set if at least one browser from our simulations negotiated a Forward Secrecy suite.
    - bit 1 - set based on Simulator results if FS is achieved with modern clients. For example, the server supports ECDHE suites, but not DHE.
    - bit 2 - set if all simulated clients achieve FS. In other words, this requires an ECDHE + DHE combination to be supported.
- **rc4WithModern** - true if RC4 is used with modern clients.
- **sims** - instance of SimDetails.
- **heartbleed** - true if the server is vulnerable to the Heartbleed attack.
- **heartbeat** - true if the server supports the Heartbeat extension.
- **openSslCcs** - results of the CVE-2014-0224 test:
    - -1 - test failed
    - 0 - unknown
    - 1 - not vulnerable
    - 2 - possibly vulnerable, but not exploitable
    - 3 - vulnerable and exploitable

Info

- **version** - SSL Labs software version as a string (e.g., "1.11.14")
- **criteriaVersion** - rating criteria version as a string (e.g., "2009f")
- **clientMaxAssessments** - the maximum number of concurrent assessments the client is allowed to initiate.
- **notice** - TODO

Key

- **size** - key size, e.g., 1024 or 2048 for RSA and DSA, or 256 bits for EC.
- **strength** - key size expressed in RSA bits.
- **alg** - key algorithm; possible values: RSA, DSA, and EC.
- **debianFlaw** - true if we suspect that the key was generated using a weak random number generator (detected via a blacklist database)
- **q** - 0 if key is insecure, null otherwise

Cert

- **subject** - certificate subject
- **commonNames[]** - common names extracted from the subject
- **altNames[]** - alternative names
- **notBefore** - timestamp before which the certificate is not valid
- **notAfter** - timestamp after which the certificate is not valid
- **issuerSubject** - issuer subject
- **sigAlg** - certificate signature algorithm
- **issuerLabel** - issuer name
- **revocationInfo** - a number that represents revocation information present in the certificate:
    - bit 0 (1) - CRL information available
    - bit 1 (2) - OCSP information available
- **crlURIs[]** - CRL URIs extracted from the certificate
- **ocspURIs[]** - OCSP URIs extracted from the certificate
- **revocationStatus** - a number that describes the revocation status of the certificate:
    - 0 - not checked
    - 1 - certificate revoked
    - 2 - certificate not revoked
    - 3 - revocation check error

- **scg** - Server Gated Cryptography support
- **validationType** - E for Extended Validation certificates; may be null if unable to determine
- **issues** - list of certificate issues, one bit per issue:
    - bit 0 (1) - no chain of trust
    - bit 1 (2) - not before
    - bit 2 (4) - not after
    - bit 3 (8) - hostname mismatch
    - bit 4 (16) - revoked
    - bit 5 (32) - bad common name
    - bit 6 (64) - self-signed
    - bit 7 (128) - blacklisted
    - bit 8 (256) - insecure signature

Chain

- **certs[]** - a list of ChainCert objects, representing the chain certificates in the order in which they were retrieved from the server
- **issues** - a number of flags that describe the chain and the problems it has:
    - bit 0 (1) - unused
    - bit 1 (2) - incomplete chain (set only when we were able to build a chain by adding missing intermediate certificates from external sources)
    - bit 2 (4) - chain contains unrelated or duplicate certificates (i.e., certificates that are not part of the same chain)
    - bit 3 (8) - the certificates form a chain (trusted or not), but the order is incorrect
    - bit 4 (16) - contains a self-signed root certificate (not set for self-signed leafs)
    - bit 5 (32) - the certificates form a chain (if we added external certificates, but 1 will be set), but we could not validate it. If the leaf was trusted, that means that we built a different chain we trusted.

ChainCert

- **subject** - certificate subject
- **label** - certificate label (user-friendly name)
- **issuerSubject** - issuer subject
- **issuerLabel** - issuer label (user-friendly name)
- **issues** - a number of flags the describe the problems with this certificate:
    - bit 0 (1) - certificate not yet valid
    - bit 1 (2) - certificate expired
    - bit 2 (4) - weak key
    - bit 3 (8) - weak signature
    - bit 4 (16) - blacklisted
- **raw** - PEM-encoded certificate data

Protocol

- **id** - protocol version number, e.g. 0x0303 for TLS 1.2
- **name** - protocol name, i.e. SSL or TLS.
- **version** - protocol version, e.g. 1.2 (for TLS)
- **v2SuitesDisabled** - some servers have SSLv2 protocol enabled, but with all SSLv2 cipher suites disabled. In that case, this field is set to true.
- **q** - 0 if the protocol is insecure, null otherwise

SimClient

- **id** - unique client ID (integer)
- **name** - text.
- **platform** - text.
- **version** - text.
- **isReference** - true if the browser is considered representative of modern browsers, false otherwise. This flag does not correlate to client's capabilities, but is used by SSL

Labs to determine if particular configuration is effective. For example, to track Forward Secrecy support, we mark several representative browsers as "modern" and then test to see if they succeed in negotiating a FS suite. Just as an illustration, modern browsers are currently Chrome, Firefox (not ESR versions), IE/Win7, and Safari.

SimDetails

- **results[]** - instances of Simulation.

Simulation

- **client** - instance of SimClient.
- **errorCode** - zero if handshake was successful, 1 if it was not.
- **attempts** - always 1 with the current implementation.
- **protocolId** - Negotiated protocol ID.
- **suiteId** - Negotiated suite ID.

Suites

- **list[]** - list of Suite objects, see below
- **preference** - *true* if the server actively selects cipher suites; if null, we were not able to determine if the server has a preference

Suite

- **id** - suite RFC ID (e.g., 5)
- **name** - suite name (e.g., TLS_RSA_WITH_RC4_128_SHA)
- **cipherStrength** - suite strength (e.g., 128)
- **dhStrength** - strength of DH params (e.g., 1024)
- **dhP** - DH params, p component
- **dhG** - DH params, g component
- **dhYs** - DH params, Ys component
- **ecdhBits** - ECDH bits
- **ecdhStrength** - ECDH RSA-equivalent strength

- **q** - 0 if the suite is insecure, null otherwise

StatusCodes

- **statusDetails** - a map containing all status details codes and the corresponding English translations. Please note that, once in use, the codes will not change, whereas the translations may change at any time.

Changes

1.0 (June 2012)

- First release, for internal use. The changelog for the 1.x version of the APIs has been removed as it's of no interest to the 2.x users.

2.0 (30 October 2014)

- First public release (alpha).