ICANN monitoring system API

Version 2.4
2017-02-08

1. Introduction

This document describes the REST API methods provided by ICANN to registry operators in order to retrieve information collected by the ICANN monitoring system.

1.1. Date and Time

All the fields that represent dates in this document must contain timestamps indicating the date and time in Coordinated Universal Time (UTC).

1.2. Glossary

In the following section, the concepts used in the monitoring system API are explained:

- **Service**: a service that may be monitored by the ICANN monitoring system. The potential monitored services are: dns, rdds, epp and dnssec.

- **Test Cycle**: series of tests executed to verify the state of a monitored Service. For DNS, the Service is considered to be up at a particular moment, if at least two of the delegated name servers registered in the DNS have successful results from tests to each of their public-DNS registered IP addresses in the root zone for the name server. For the RDDS Services (i.e. Whois tcp/43 and web-Whois) to be considered up at a particular moment, the Services must have successful results from tests to the randomly chosen public-DNS registered IP address to which whois.nic.<TLD> resolves. If 51% or more of the testing probe nodes see a monitored Service as unavailable at a given time, the Service will be considered unavailable. For RDDS, if any of the RDDS Services (i.e. Whois tcp/43 and web-Whois) is considered unavailable, the RDDS will be considered unavailable. The minimum number of active testing probe nodes to consider the results of a test cycle as valid at any given time is 20 for DNS and 10 for RDDS; otherwise the test cycle results will be discarded and the Service will be considered up.

- **Test**: for DNS it means one non-recursive DNS query sent to a particular IP address via UDP or TCP; if DNSSEC is offered in the queried DNS zone, for a query to be considered answered, the signatures must be positively verified against a corresponding DS record published in the parent zone. For RDDS it means one query sent to a particular IP address. The answer to the query must contain the corresponding information from the Registry System, otherwise the query will be considered unanswered. A query with a RTT higher than X milliseconds will also be considered unanswered. For DNS (UDP) X=2,500 ms, DNS (TCP) X=7,500 ms for RDDS X=10,000 ms.

- **RTT (Round Trip Time)**: for DNS/UDP, the sequence of two packets, the UDP DNS query and the corresponding UDP DNS response. For DNS/TCP, the sequence of packets from the start of the TCP connection to its end. For Whois tcp/43, the sequence of packets from the start of the TCP connection to its end, including the reception of the Whois tcp/43 response. For web-Whois, the sequence of packets from the start of the TCP connection to its end, including the reception of a HTTP response; if the Registry Operator implements HTTP URL redirection (e.g. HTTP 302), only the last HTTP transaction is measured.

- **Emergency Threshold**: downtime threshold that if reached by any of the monitored Services may cause the TLD's Services emergency transition to an interim registry operator. To reach an Emergency Threshold a Service must accumulate X hours of total downtime during the last 7 days (i.e. rolling week). For DNS X=4, for RDDS X=24.

- **Incident**: an Incident is the collection of measurements from the moment an Alarm is generated until the moment that the Alarm is cleared. An Incident can have 2 distinct states:

  - Active: measurements corresponding to a current downtime.
  - Resolved: measurements corresponding to past downtime.

  The measurements of Incidents that occurred in the last 7 days (i.e. rolling week, from: the current date and time -7days, to: the current date and time) are considered for the Service's Emergency Threshold calculations.

- **Alarm**: an Alarm signals that a Service has been detected as being down because X consecutive test cycles with Y minutes between them failed. An Alarm is cleared when the Service is detected as being up because X consecutive test cycles with Y minutes between them have been successful. For DNS, X=3 and Y=1. For RDDS, X=2 and Y=5. An alarmed Service triggers the creation of an Incident; if the Alarm is cleared then the Incident will be marked as resolved.

- **False Positive**: a flag set to an Incident indicating that an Incident was found by a manual process to be a false positive. When an Incident is marked as a False Positive the measurements of the Incident are not used for the Emergency Threshold calculations.

2. Common elements used in this specification

In the following section, common elements used in this specification are explained:

- **<base_url>:** the base URL of the ICANN Monitoring System API is https://mosapi.icann.org/mosapi/<version>/<tld>, for example: https://mosapi.icann.org/mosapi/v1/example/monitoring/state

  Where:

  - <version> must be substituted by the version number of the specification supported by the server. For this specification its value must be 'v1'.
  - <tld> must be substituted by the TLD being queried. In case of an IDN TLD, the A-label must be used.

- **<service>** must be substituted by the Service being queried. The possible values of Service, as described in Section 1 of Specification 10, are: dns, dnssec, rdds, and epp.

3. Session handling

The SLA Monitoring system provides two API methods for session handling, the authentication mechanism is HTTP Basic Access Authentication as specified in RFC 2617.

Authentication credentials for the API methods are provided by ICANN per TLD. The credentials must only be used when creating a session using the <base_url>/login API method described in this section.

3.1. Creating a session

```
<base_url>/login
```

Possible results:

- HTTP/401, the <base_url>/login API method provides a HTTP/401 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Invalid credentials" when the authentication credentials are invalid.

- HTTP/403, the <base_url>/login API method provides a HTTP/403 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Your IP address is not allowed to connect for this TLD" if the credentials are valid but the connecting IP address is not whitelisted for the specified <tld>.

- HTTP/200, when a valid request is received, the <base_url>/login API method provides an HTTP/200 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Login successful". The HTTP header Set-Cookie is set with the cookie attributes "id=<sessionID>; expires=<date>; path=/sla/<version>/<tld>; secure; httpOnly".

    - The <sessionID> value is a 160-bit random value encoded in Base16.
    - The <date> value is the expiration date of the session.
    - The <version> value must be 'v1'.
    - The <tld> value is the TLD being queried.

Example using curl (https://curl.haxx.se/) for a login request:

```
curl --cookie-jar cookies.txt --user user:passwd
https://mosapi.icann.org/mosapi/v1/example/login
```

Note: the <base_url>/login API method may implement rate-limiting as one of the protection mechanisms to mitigate the risk of performance degradation.

Note: Every time the <base_url>/login API method successfully validates the credentials and origin IP address, a new session is created. Only 2 concurrent sessions are permitted per TLD. A session is only terminated after its expiration date, by using the <base_url>/logout API method, or if the session is the oldest and a new session is being created that would be above the limit of permitted concurrent sessions.

3.2. Closing a session

```
<base_url>/logout
```

In order to destroy a session, the client must set the HTTP header Cookie with the value "id=<sessionID>", where <sessionID> must be a 160-bit random value provided in the HTTP server response of a successful "login" request. If multiple cookies are provided, the first cookie is used for destroying the session.

Possible results:

- HTTP/401, the <base_url>/logout API method provides a HTTP/401 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Invalid session ID" when the specified <sessionID> is invalid.

- HTTP/403, the <base_url>/logout API method provides a HTTP/403 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Your IP address is not allowed to connect for this TLD" if the specified <sessionID> is valid but the connecting IP address is not whitelisted for the specified <tld>.

- HTTP/200, when a valid request is received, the <base_url>/logout API method provides a HTTP/200 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Logout successful". The HTTP header Set-Cookie is set with the values "id=; expires=<date>; path=/sla/<version>/<tld>; secure; httpOnly".

    - The <date> value is set to the unix epoch date and time.
    - The <version> value must be 'v1'.
    - The <tld> value is the TLD being queried.

Example using CURL for a logout request:

```
curl --cookie cookies.txt --cookie-jar cookies.txt
https://mosapi.icann.org/mosapi/v1/example/logout
```

4. API method authentication

When sending a request to the monitoring system API, the client must set the HTTP header Cookie with the value "id=<sessionID>", where <sessionID> must be the 160-bit random value provided in the last HTTP server response of a successful "login" request. If multiple cookies are provided, the first cookie is used for validating the session.

The following responses may be provided by the API by the methods shown below:

- HTTP/401, the API method provides a HTTP/401 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Invalid session ID" when the specified <sessionID> is invalid.

- HTTP/403, the API method provides a HTTP/403 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Your IP address is not allowed to connect for this TLD" if the specified <sessionID> is valid but the connecting IP address is not whitelisted for the specified <tld>.

5. Specification 10 monitoring

Registries may access the monitoring information collected by the SLA Monitoring system using the GET HTTP verb in the API methods described below. The monitoring information will be refreshed at least every 2 minutes.

5.1. Monitoring the state of a TLD

```
<base_url>/monitoring/state
```

Possible results:

- HTTP/200, when a valid request is received, the <base_url>/monitoring/state API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "tld", a JSON string that contains the monitored TLD.

  - "status", a JSON string that contains the status of the TLD as seen from the monitoring system. The "status" field may contain one of the following values:

    - Up: all of the monitored Services are up.
    - Down: one or more of the monitored Services are down.
    - Up-inconclusive: the SLA monitoring system is under maintenance or the information is not conclusive, therefore the TLD is considered to be up by default.

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "testedService", a JSON array that contains detailed information for each Service being monitored. The "testedService" object contains the following fields:

    - "service", a JSON string that contains the monitored Service (see definition of Service in section 1.2) field with the monitored Service.

    - "status", a JSON string that contains the status of the Service as seen from the monitoring system. The "status" field can contain one of the following values:

      - Up: the monitored Service is up.
      - Down: the monitored Service is down.
      - Up-inconclusive: the SLA monitoring system is under maintenance or the information is not conclusive, therefore the TLD is considered to be up by default.

- o "emergencyThreshold", a JSON number that contains the current percentage of the Emergency Threshold of the Service. Note: the value "0" specifies that the are no Incidents affecting the Emergency Threshold of the Service.

- o "incidents", a JSON array that contains "incident" objects. The "incident" object contains:

  - "incidentID", a JSON string that contains the Incident identifier (i.e. <incidentID>). The Incident identifier (i.e. <incidentID>) is a concatenation of the unix time stamp of the start date and time of the Incident, followed by a full stop (".", ASCII value 0x002E), followed by the monitoring system identifier.

  - "startTime", a JSON number that contains the unix time stamp of the start date and time of the Incident.

  - "falsePositive", a JSON boolean value that contains true or false with the False Positive status of the Incident.

  - "state", a JSON string that contains the current state (i.e. Active or Resolved) of the Incident.

  - "endTime", a JSON number that contains the unix time stamp of the end date and time of the Incident; if the Incident state is active the "endTime" field will contain a null value.

Example using CURL to request the state of a TLD:

```
curl --cookie cookies.txt  https://mosapi.icann.org/mosapi/v1/example/monitoring/state
```

Example of a JSON response for a TLD state request:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "tld": "example",
  "status": "Down",
  "testedService": [
    {
      "service": "DNS",
      "status": "Down",
      "emergencyThreshold": 25,
      "incidents": [
        {
          "incidentID": "1422492450.699",
          "startTime": 1422492450,
          "falsePositive": false,
          "state": "Active",
          "endTime": null
        },
        {
          "incidentID": "1422492850.3434",
          "startTime": 1422492850,
          "falsePositive": true,
          "state": "Resolved",
          "endTime": 1422492950
        }
      ]
    },
    {
      "service": "RDDS",
      "status": "Up",
```

```
      "emergencyThreshold": 10,
      "incidents": [
        {
          "incidentID": "1422492450.699",
          "startTime": 1422492450,
          "falsePositive": false,
          "state": "Resolved",
          "endTime": 1422492950
        }
      ]
    }
  ]
}
```

5.2. Monitoring the Alarm status of a Service

```
<base_url>/monitoring/<service>/alarmed
```

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/downtime API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/alarmed API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "alarmed", a JSON string that contains one of the following values:

    - Yes: an Alarm exists for the Service.
    - No: an Alarm does not exist for the Service.
    - No-inconclusive: the SLA monitoring system is under maintenance or the information is not conclusive, therefore the Service is considered to be up by default.
    - Disabled: the Service is not being monitored.

Example using CURL to request the Alarm status of a Service:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/alarmed
```

Example of a JSON response for a Service in Alarm status:

```
{
```

```
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "alarmed": "Yes"
}
```

5.3. Monitoring the availability of a Service

```
<base_url>/monitoring/<service>/downtime
```

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/downtime API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/downtime API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".
  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

    - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

    - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

    - "downtime", a JSON number that contains the number of minutes of downtime of the Service during a rolling week period.

Example using CURL to request the availability of a Service:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/downtime
```

Example of a JSON response for a Service availability request:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "downtime": 935
}
```

5.4. Query a list of Incidents for a Service

```
<base_url>/monitoring/<service>/incidents?startDate=<startDate>&endD
ate=<endDate>&falsePositive=<falsePositive>
```

Where:

- Optional <startDate> to be substituted by the unix time stamp of the 'after' date and time to filter by. The filter will match Incidents that started after the provided date and time.

- Optional <endDate> to be substituted by the unix time stamp of the 'before' date and time to filter by. The filter will match Incidents that started before the provided date and time.

- Optional <falsePositive> to be substituted by true or false in order to filter the Incidents marked as False Positive. If its value equals true, only Incidents marked as False Positive will be returned. If its value equals false, only Incidents not marked as False Positive will be returned. If <falsePositive> is not defined, all Incidents will be returned.

Note: The <base_url>/monitoring/<service>/incidents supports a maximum of 31 days difference between <startDate> and <endDate>. If only <startDate> is provided, the API method will return results that are within 31 days after the date and time provided. If only <endDate> is provided, the API method will return results that are within 31 days before the date and time provided. If neither <startDate> nor <endDate> are provided, the API method will return results that are within 31 days before the current date and time. If <endDate> is in the future, the value of <endDate> will be the current date and time.

Possible results:

- HTTP/400, see section 8.

- HTTP/404, the <base_url>/monitoring/<service>/incidents API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/incidents API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "incidents", JSON array, see definition in section 5.1.

Example using CURL to request a list of Incidents of a Service:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/incidents?startDate=14224924
00&endDate=1422493000
```

Example of a JSON response showing a list of Incidents:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
```

```
  "incidents": [
    {
      "incidentID": "1422492450.699",
      "startTime": 1422492450,
      "falsePositive": false,
      "state": "Active",
      "endTime": null
    },
    {
      "incidentID": "1422492850.3434",
      "startTime": 1422492850,
      "falsePositive": true,
      "state": "Resolved",
      "endTime": 1422492950
    }
  ]
}
```

5.5. Monitoring the state of a particular Incident

```
<base_url>/monitoring/<service>/incidents/<incidentID>/state
```

Where:

- <incidentID> must be substituted by the Incident id assigned by the monitoring system.

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/incidents/<incidentID>/state API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <incidentID> does not exist or if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/incidents/<incidentID>/state API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "incidents", JSON array, see definition in section 5.1.

Example using CURL to request the state of an Incident:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/incidents/1422492450.699/sta
te
```

Example of a JSON response for an Incident state request:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "incidents": [
    {
      "incidentID": "1422492450.699",
      "startTime": 1422492450,
      "falsePositive": false,
      "state": "Active",
      "endTime": null
    }
  ]
}
```

5.6. Monitoring the False Positive flag of an Incident

```
<base_url>/monitoring/<service>/incidents/<incidentID>/falsePositive
```

Where:

- <incidentID> must be substituted by the Incident id assigned by the monitoring system.

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/incidents/<incidentID>/falsePositive API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <incidentID> does not exist or if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/incidents/<incidentID>/falsePositive API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "falsePositive", a JSON boolean value that contains true or false with the False Positive status of the Incident.

  - "updateTime", a JSON number that contains the unix time stamp of the date and time the False Positive status was updated; if the False Positive status has never been updated the "updateTime" field will contain a null value.

Example using CURL to request the False Positive flag of an Incident:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/incidents/1422492930.699/falsePositive
```

Example of a JSON response for an Incident flagged as False Positive:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "falsePositive": true,
  "updateTime": 1422494780
}
```

Note: The False Positive flag is the only thing that may change after an Incident is resolved. The user MAY be notified if an Incident is marked as a false positive by an offline mechanism.

5.7. Querying the list of measurements for an Incident

```
<base_url>/monitoring/<service>/incidents/<incidentID>/
```

Where:

- <incidentID> must be substituted by the Incident id assigned by the monitoring system.

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/incidents/<incidentID>/ API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <incidentID> does not exist or if the specified <service> is not being monitored.

- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/incidents/<incidentID>/ API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".

  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.

  - "measurements", a JSON array that contains a list of <measurementID> values assigned by the monitoring system. A <measurementID> is a concatenation of the unix time stamp of the date and time when the measurement was computed, followed by a full stop (".", ASCII value 0x002E), followed by a random value, followed by a full stop (".", ASCII value 0x002E), followed by the string "json" (ASCII value, 0x006A + 0x0073 + 0x006F + 0x006E).

Example using CURL to request the list of measurements of an Incident:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/incidents/1422492930.699/
```

Example of a JSON response showing a list of measurements identifiers:

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "measurements": [
    "1422492930.699.json",
    "1422492990.699.json",
    "1422493050.699.json",
    "1422493110.699.json"
  ]
}
```

5.8. Querying the details of a particular measurement

```
<base_url>/monitoring/<service>/incidents/<incidentID>/<measurementI
D>
```

Where:

- <incidentID> must be substituted by the Incident id assigned by the monitoring system.
- <measurementID> must be substituted by the measurement id assigned by the monitoring system.

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/incidents/<incidentID>/<measurementID> API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <incidentID> does not exist, the specified <measurementID> does not exist or if the specified <service> is not being monitored.
- HTTP/200, when a valid request is received, the <base_url>/monitoring/<service>/incidents/<incidentID>/<measurementID> API method provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".
  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:
  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".
  - "lastUpdateApiDatabase", a JSON number that contains the unix time stamp of the date and time that the monitoring information provided in the monitoring system API was last updated from the monitoring system central database.
  - "tld", a JSON string that contains the monitored TLD.

- "cycleCalculationDateTime", a JSON number that contains the date and time the test cycle results were computed.

- "testedInterface", a JSON array that contains information about the interface being tested. The "testedInterface" fields contains the following fields:

    - "interface", a JSON string that contains the tested interface.

    - "status", a JSON string that contains the status of the Service after computing the test cycle results. The "status" field can contain one of the following values:

        - Up: the monitored Service is up.
        - Down: the monitored Service is down.
        - Up-inconclusive: the number of active testing probe nodes is below the minimum number needed to consider the test cycle results as conclusive.

    - "probes", a JSON array that contains detailed monitoring information per probe node. The "probes" field contains the following fields:

        - "city", a JSON string with the location the location of the probe node.

        - "status", a JSON string that contains the status of the interface as seen from the probe node. The "status" field can contain one of the following values:

            - Up: the monitored Service is up.
            - Down: the monitored Service is down.
            - Nodata: the central server did not receive enough data.

        - "testData", a JSON array that contains monitoring information. The "testData" field contains the following fields:

            + "target", a JSON string that in the case of the DNS Service contains the name server being tested, in the case of RDDS, this field contains "null".

            + "status", a JSON string that in the case of the DNS Service contains the status of the name server being tested. In the case of RDDS this field contains the status of the IP address being tested (available in the "metrics" element, see below). The "status" field contains the following fields:

                - Up: the test was successful.
                - Down: the test was not successful.
                - No result: the probe node result was not received by the central database.
                - Offline: the probe node is offline.

            + A "metrics", a JSON array with monitoring details of particular tests. The "metrics" field contains the following fields:

                - "testDateTime", a JSON number that contains the date and time the result was computed.

                - "targetIP", a JSON string with the IP Address being tested.

- "rtt", a JSON number that contains the milliseconds needed for the query to be resolved. If the "description" field contains an error code, the "rtt" field will contain a null value.

- "result", a JSON string that contains the value "ok" if the query response was valid, or an error code if it was invalid. The information regarding the error codes may be found in section 5.8.1 and 5.8.2.

Note: the JSON object for the measurement details provides the status of the test cycle computed from the results of all probe nodes.

Example using CURL to request the details of a measurement:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/monitoring/dns/incidents/1422734490.699/142
2734490.699.json
```

Example of JSON response for a DNS Service measurement details request:


```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "tld": "example",
  "cycleCalculationDateTime": 1422734490,
  "status": "Up",
  "testedInterface": [
    {
      "interface": "DNS",
      "status": "Up",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": "ns1.nic.example",
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::1",
                  "rtt": null,
                  "description": "-204, DNSSEC error"
                },
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.1",
                  "rtt": null,
                  "description": "-204, DNSSEC error"
                }
              ]
            },
            {
              "target": "ns2.nic.example",
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::2",
                  "rtt": null,
                  "description": "-204, DNSSEC error"
                },
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.2",
                  "rtt": null,
                  "description": "-204, DNSSEC error"
```

```
            }
          ]
        }
      ]
    },
    {
      "city": "Sydney",
      "status": "Up",
      "testData": [
        {
          "target": "ns1.nic.example",
          "status": "Up",
          "metrics": [
            {
              "testDateTime": 1422734508,
              "targetIP": "192.0.2.1",
              "rtt": 5,
              "description": "ok"
            }
          ]
        },
        {
          "target": "ns2.nic.example",
          "status": "Up",
          "metrics": [
            {
              "testDateTime": 1422734508,
              "targetIP": "192.0.2.2",
              "rtt": 6,
              "description": "ok"
            }
          ]
        }
      ]
    }
  ]
}
```


Example of JSON response for a RDDS Service measurement details request:


```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "tld": "example",
  "cycleCalculationDateTime": 1422734490,
  "status": "Down",
  "testedInterface": [
    {
      "interface": "RDDS43",
      "status": "Up",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": null,
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::1",
                  "rtt": null,
                  "description": "-200, No reply from name server"
                }
              ]
            }
          ]
```

```
        },
        {
          "city": "Sydney",
          "status": "Up",
          "testData": [
            {
              "target": null,
              "status": "Up",
              "metrics": [
                {
                  "testDateTime": 1422734508,
                  "targetIP": "192.0.2.1",
                  "rtt": 250,
                  "description": "ok"
                }
              ]
            }
          ]
        }
      ]
    },
    {
      "interface": "RDDS80",
      "status": "Down",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": null,
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.1",
                  "rtt": null,
                  "description": "-200, No reply from name server"
                }
              ]
            }
          ]
        },
        {
          "city": "Sydney",
          "status": "Down",
          "testData": [
            {
              "target": null,
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734508,
                  "targetIP": "192.0.2.1",
                  "rtt": null,
                  "description": "-200, No reply from name server "
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```


5.8.1. DNS/DNSSEC Monitoring error codes

The following table lists the error codes for DNS/DNSSEC monitoring:

Result Code	Message	Description
-200	No reply from name server	No reply from name server
-201	Invalid reply from name server	The response received from the server is invalid (e.g. RCODE=SERVFAIL).
-204	DNSSEC Error	The response received form the server is malformed or the digital signature does not validate using the previously validated keyset.
-206	Keyset is not valid	Error while validating the keyset of the TLD.

Note: DNSSEC errors -204 and -206 trigger a downtime for both DNS and DNSSEC Services.

Note: A future version of the API may add error codes in order to provide additional details regarding the issue being detected.

5.8.2. RDDS Monitoring error codes

The following table lists the error codes for RDDS monitoring:

Result Code	Message	Description
-200	No reply from RDDS43 server	Connection timed out while trying to get a response from the server.
-201	Syntax error on RDDS43 output	Syntax error on RDDS43 output
-204	No reply from RDDS80 server	Connection timed out while trying to get a response from the server.
-205	Cannot resolve the Whois server hostname	Error when trying to resolve the Whois server hostname (e.g. whois.nic.example).
-207	Invalid HTTP status code	No HTTP/200 status code in response (after following redirects).

Note: the DNS resolvers used in the system validate DNSSEC.

Note: A future version of the API may add error codes in order to provide additional details regarding the issue being detected.

6. Maintenance window support

The Base Registry Agreement allows the Registry Operator to inform ICANN of planned maintenance times. However, note that per the Base Registry Agreement, there is no provision for planned outages or similar periods of unavailable or slow service; any downtime, be it for maintenance or due to system failures, will be noted simply as downtime.

6.1. Common elements for maintenance window support

6.1.1. Schedule object

Registry operators use the schedule object to manage maintenance windows in the Monitoring System API. The schedule object contains the following fields:

- "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".
- "name", a JSON string that contains a descriptive name of the maintenance window.
- "enable", a JSON boolean value that contains true when the maintenance window is enabled and false when the maintenance window is disabled.
- "description", a JSON string that contains a description of the maintenance widow.
- "startTime", a JSON number that contains the date and time (specified in unix timestamp) when the maintenance window starts being active.
- "endTime", a JSON number that contains the date and time (specified in unix timestamp) when the maintenance window ends being active.

ICANN will suspend Emergency Escalation services only for the 10% Emergency Threshold alert for RDDS and EPP when an enabled ("enabled"=true) schedule object exist, and the threshold is reached on a time covered by the "startTime" and "endTime".

Example of a JSON schedule object:

```
{
    "version": 1,
    "name": "load balancer upgrade",
    "enabled": true,
    "description": "The load balancer will be upgraded to version 3.4",
    "startTime": 1485941725,
    "endTime": 1486001764
}
```

6.1.2. Schedule object identifier

A schedule object is uniquely identified by a <scheduleID> identifier. The <scheduleID> is an UUID (as defined in RFC4122) generated by the user. The user defines the <scheduleID> identifier when creating the schedule object.

6.2. Creating or updating a schedule for a maintenance window

In order to create or update a schedule for a maintenance window, the user sends a schedule object using the PUT HTTP verb in the API method provided at:

```
<base_url>/mntWin/<service>/<scheduleID>
```

Possible results:

- HTTP/400, see section 8.
- HTTP/404, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <service> does not exist.
- HTTP/200, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/200 status code if the API method was able to receive the input, no syntax issue was found in the input, and the PUT verb was successful. The API method sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "OK".

6.3. Deleting a schedule for a maintenance window

In order to delete a schedule for a maintenance window, the user make use of the DELETE HTTP verb in the API method provided at:

```
<base_url>/mntWin/<service>/<scheduleID>
```

Possible results:

- HTTP/400, see section 8.
- HTTP/404, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <scheduleID> does not exist or if the specified <service> does not exist.
- HTTP/200, the <base_url>/mntWin/<service>/<scheduleID> provides a HTTP/200 status code if the API method was able to receive the input, no syntax issue was found in the input, and the DELETE verb was successful. The API method sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "OK".

6.4. Retrieving a schedule object for a maintenance window

In order to get the information of a schedule object, the user make use of the GET HTTP verb in the following URL:

```
<base_url>/mntWin/<service>/<scheduleID>
```

Possible results:

- HTTP/400, see section 8.

- HTTP/404, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available" if the specified <scheduleID> does not exist or if the specified <service> does not exist.

- HTTP/200, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/200 status code if the API method was able to receive the input, no syntax issue was found in the input, and the GET verb was successful. The API method sets the HTTP header Content-type to "application/json; charset=utf-8". The schedule JSON object (see section 6.1.1) is provided in the HTTP Entity-body.

6.5. Getting the list of maintenance windows that have not ended yet

In order to get a list of maintenance window identifiers (i.e. "scheduleID") that have not ended yet, the user make use of the GET HTTP verb in the API method provided by ICANN at:

```
<base_url>/mntWin/<service>
```

Possible results:

- HTTP/404, the <base_url>/mntWin/<service>/<scheduleID> API method provides a HTTP/404 status code, sets the HTTP header Content-type to "text/plain; charset=utf-8", and provides a text response in the HTTP Entity-body with the string "Not available if the specified <service> does not exist.

- HTTP/200, the <base_url>/mntWin/<service> API method provides a HTTP/200 status code if the API method was able to receive the input, and the GET verb was successful. The API method sets the HTTP header Content-type to "application/json; charset=utf-8". A JSON array of schedule objects is provided in the HTTP Entity-body.

Example using CURL to request the list of maintenance windows:

```
curl --cookie cookies.txt https://mosapi.icann.org/mosapi/v1/example/mntWin/rdds
```

Example of a JSON array that contains the list of maintenance windows identifiers:

```
{
    "schedules": [{
        "scheduleID": "7b2d3012-41f7-4bce-89e9-9a9b85575fa6"
    }, {
        "scheduleID": "37e71da9-827d-450a-9909-a64ba42af1d8"
    }]
}
```

7. Probe node network

The current list of probe nodes used by the Monitoring System may be retrieved by using the GET HTTP verb in the API method provided by ICANN at:

```
<base_url>/monitoring/nodes
```

Possible results:

- HTTP/200, when a valid request is received, the API provides a HTTP/200 status code and sets the HTTP header Content-type to "application/json; charset=utf-8".

  If a valid request is received, a JSON object with the fields listed below is provided in the HTTP Entity-body:

  - "version", a JSON number that contains the version number of the JSON object intended for future upgrades of the specification; for this version the value will always be "1".
  - "updateTime", a JSON number that contains the unix time stamp of the date and time when the list was updated.
  - "probeNodes", a JSON array that provides information per probe node. The "probeNodes" contains the following JSON objects:
    - "city", a JSON string that contains the location of the probe node.
    - "ipV4", a JSON string that contains the IPv4 address of the probe node. If a probe node does not support IPv4, the "ipV4" field will contain a null value.
    - "ipV6", a JSON string that contains the IPv6 address of the probe node. If a probe node does not support IPv6, the "ipV6" field will contain a null value.

Example using CURL to request the list of probe nodes:

```
curl --cookie cookies.txt  https://mosapi.icann.org/mosapi/v1/example/monitoring/nodes
```

Example of a JSON object that contains the list of probe nodes:

```
{
  "version": 1,
  "updateTime": 1422492450,
  "probeNodes": [
    {
      "city": "Amsterdam",
      "ipV4": "192.0.2.3",
      "ipV6": "2001:DB8::3"
    },
    {
      "city": "Beijing",
      "ipV4": "192.0.2.4",
      "ipV6": null
    },
    {
      "city": "Boston",
      "ipV4": "192.0.2.5",
      "ipV6": "2001:DB8::5"
    },
    {
```

```
      "city": "Istanbul",
      "ipV4": "192.0.2.6",
      "ipV6": null
    },
    {
      "city": "WashingtonDC",
      "ipV4": "192.0.2.7",
      "ipV6": "2001:DB8::7"
    },
    {
      "city": "Sydney",
      "ipV4": "192.0.2.8",
      "ipV6": "2001:DB8::8"
    }
  ]
}
```

8. HTTP/400 extended error codes

The API methods provides a HTTP/400 if the input does not comply with the business rules or the syntax of the input is invalid. The API method sets the HTTP header Content-type to "application/json; charset=utf-8". A JSON object with the fields listed below is provided in the HTTP Entity-body:

- "resultCode", a JSON number that contains the result code.
- "message", a JSON string the contains the standard error message defined in the table below.
- "description", a JSON string the may be used to provide additional error diagnostic information.

Example of a JSON object that contains extended error codes:

```
{
   "resultCode":2001,
   "message":"The UUID syntax is incorrect",
   "description":"The UUID (ee69b727-2abb-4f1c-8208-e5e76zzd758f) syntax is incorrect"
}
```

The following table contains the extended error codes for the HTTP/400 status:

Result Code	API methods	HTTP Verb			Message
		PUT	DELETE	GET	
2001	<base_url>/mntWin/<service>/<scheduleID>	•	•	•	The UUID syntax is incorrect.
2002	<base_url>/mntWin/<service>/<scheduleID>	•			The maintenance window start date and time is not 24 hours ahead of the current date and time.
2003	<base_url>/mntWin/<service>/<scheduleID>	•			The period specified by start and end date and time is greater than the monthly SLR for the service.
2004	<base_url>/mntWin/<service>/<scheduleID>	•			The period specified in the maintenance window collides with a previously scheduled maintenance window for the service.
2005	<base_url>/mntWin/<service>/<scheduleID>	•	•	•	The maintenance window functionality is disabled for this TLD.
2006	<base_url>/mntWin/<service>/<scheduleID>		•		The maintenance window that you are trying to delete already started.
2007	<base_url>/mntWin/<service>/<scheduleID>	•			The endTime is in the past or before the startTime.
2008	<base_url>/mntWin/<service>/<scheduleID>	•			The startTime syntax is incorrect.
2009	<base_url>/mntWin/<service>/<scheduleID>	•			The endTime syntax is incorrect.
2010	<base_url>/mntWin/<service>/<scheduleID>	•			The maintenance window that you are trying to update already ended, updates are not allowed.

2011	<base_url>/monitoring/<service>/incidents			•	The difference between endDate and startDate is more than 31 days.
2012	<base_url>/monitoring/<service>/incidents			•	The endDate is before the startDate.
2012	<base_url>/monitoring/<service>/incidents			•	The startDate syntax is incorrect.
2013	<base_url>/monitoring/<service>/incidents			•	The endDate syntax is incorrect.
2100	<base_url>/mntWin/<service>/<scheduleID>	•			The JSON syntax is invalid.