ICANN Monitoring System API (MoSAPI)

Version 2.8.1
2018-11-26

Document Revision History

Version	Publishing date	Description of the change	Projected date to implement the version of the specification in production
2.5	2017/11/28	First version released to the public.	In production
2.6	2018/02/12	ADDITION - Default rate-limit and expiration date values were added in section 3.1.	In production
		ADDITION - Maximum length definitions for name and description were added in section 6.1.1.	
		ADDITION - Result code 2016 (section 8) was added to the result code table.	
		CHANGE - Result code 2007 message (section 8) was changed to cover the case of equal values in the endTime and startTime.	
2.7	2018/03/06	MoSAPI released as a production service.	In production
2.8	2018/10/02	ADDITION - UP-inconclusive-no-data and UP-inconclusive-no-probes were added to sections 5.1 and 5.8.	Never released to production, replaced with version 2.8.1
		CHANGE - Error codes changed in sections 5.8.1 and 5.8.2.	
		CHANGE - Editorial updates.	
2.8.1	2018/11/26	CHANGE - Minor adjustments to the list of error codes in sections 5.8.1 and 5.8.2.	First quarter of 2019

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 API (MoSAPI).

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. Credentials

The MoSAPI uses the same username, password, and list of IP address blocks (IPv4 and/or IPv6) as the Registry Reporting Interface (RRI). The Registry Operator manages these credentials through the Global Domain Division (GDD) portal.

The MoSAPI supports both IPv4 and IPv6 transport.

The MoSAPI requires the use of HTTPS to provide confidentiality, server authentication, and integrity in the communication channel.

1.3. Glossary

In the following section, the concepts used in the MoSAPI are explained:

- **Service**: a service that may be monitored by the MoSAPI. 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 MoSAPI 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 MoSAPI provides two 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/429, the <base_url>/login API method provides a HTTP/429 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 "You reached the limit of login requests per minute" for the specified <tld>.

  Note: Only connections originating from IP addresses whitelisted for the <tld> counts towards the limit. Connections originating from IP addresses not included in the whitelist are dropped without further validation. Currently, the rate-limit allows for two login requests every 300s per <tld>. Developers are encouraged to re-use the session to minimize the number of login requests.

- 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=**<base_url>**; secure; httpOnly".

  · The <sessionID> value is a 160-bit random value encoded in Base16.
  · The <date> value is the expiration date of the session.

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: 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 default, the value of expiration date is 15 minutes after the session is created), 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=**<base_url>**; 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 https://mosapi.icann.org/mosapi/v1/example/logout
```

4. API method authentication

When sending a request to the MoSAPI, 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 MoSAPI 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, therefore all the monitored Services of the TLD are considered to be up by default. Note: if the status is "Up-inconclusive", all Services in the "testedServices" array will have the status with a value of "disabled".

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

  - "testedServices", a JSON array that contains detailed information for each potential monitored service (i.e. DNS, RDDS, EPP, DNSSEC). Each <service> object contains the following fields:

    - "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.
      - Disabled: the Service is not being monitored.
      - UP-inconclusive-no-data: indicates that there are enough probe nodes online, but not enough raw data points were received to make a determination.
      - UP-inconclusive-no-probes: indicates that there are not enough probe nodes online to make a determination.

9

- 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:

```
{
    "tld": "example",
    "lastUpdateApiDatabase": 1496923082,
    "status": "Down",
    "testedServices": {
        "DNS": {
            "status": "Down",
            "emergencyThreshold": "10.0000",
            "incidents": [{
                "incidentID": "1495811850.1700",
                "endTime": null,
                "startTime": "1495811850",
                "falsePositive": false,
                "state": "Active"
            }]
        },
        "DNSSEC": {
            "status": "Down",
            "emergencyThreshold": "10.0000",
            "incidents": [{
                "incidentID": "1495811790.1694",
                "endTime": null,
                "startTime": "1495811790",
                "falsePositive": false,
                "state": "Active"
            }]
        },
        "EPP": {
            "status": "Disabled"
        },
        "RDDS": {
            "status": "Disabled"
        }
    },
    "version": 1
}
```

5.2.   Monitoring the Alarm status of a Service

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

Possible results:

- HTTP/404, the <base_url>/monitoring/<service>/alarmed 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 MoSAPI 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.
    · 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 MoSAPI 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 MoSAPI 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 MoSAPI 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 MoSAPI 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. The default value is false.

  - "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/fal
sePositive
```

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 MoSAPI 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"
  ]
}
```

18

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 MoSAPI was last updated from the monitoring system central database.

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

  - "service", a JSON string that contains the Service being queried. The possible values of Service, as described in Section 1 of Specification 10, are: dns, dnssec, rdds, and epp.

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

  - "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:

    o Up: the monitored Service is up.
    o Down: the monitored Service is down.
    o UP-inconclusive-no-data: indicates that there are enough probe nodes online, but not enough raw data points were received to make a determination.
    o UP-inconclusive-no-probes: indicates that there are not enough probe nodes online to make a determination.

19

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

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

    o "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.
            · Offline: the probe node is offline. Note: the probe node is not considered part of the probe node universe when calculating the rolling week thresholds.
            · No result: results from this probe node were not received by the central server when the calculations were executed. Note: the service is considered to be up for rolling week threshold calculations.

        ▪ "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 considered successful.
                · Down: the test was not considered successful.

            + 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. If the "result" field contains "no data", the "testDateTime" field will contain a null value.

                - "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 "result" field contains an error code or "no data", the "rtt" field will contain a null value.

                - "result", a JSON string that contains the value "ok" if the query response was valid, "no data" if no data was received from the probe node, or an error code if the result is not valid. 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",
  "service": "dns",
  "cycleCalculationDateTime": 1422734490,
  "status": "Up",
  "testedInterface": [
    {
      "interface": "DNS",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": "ns1.nic.example",
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::1",
                  "rtt": null,
                  "result": "-204"
                },
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.1",
                  "rtt": null,
                  "result": "-204"
                }
              ]
            },
            {
              "target": "ns2.nic.example",
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::2",
                  "rtt": null,
                  "result": "-204"
                },
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.2",
                  "rtt": null,
                  "result": "-204"
                }
              ]
            }
          ]
        },
        {
```

```
            "city": "Sydney",
            "status": "Up",
            "testData": [
              {
                "target": "ns1.nic.example",
                "status": "Up",
                "metrics": [
                  {
                    "testDateTime": 1422734508,
                    "targetIP": "192.0.2.1",
                    "rtt": 5,
                    "result": "ok"
                  }
                ]
              },
              {
                "target": "ns2.nic.example",
                "status": "Up",
                "metrics": [
                  {
                    "testDateTime": null,
                    "targetIP": "192.0.2.2",
                    "rtt": null,
                    "result": "no data"
                  }
                ]
              }
            ]
          },
          {
            "city": "Los Angeles",
            "status": "Offline",
            "testData":[]
          },
          {
            "city": "Sao Paolo",
            "status": "No result",
            "testData": []
          }
        ]
      }
    ]
  }
```

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

```
{
  "version": 1,
  "lastUpdateApiDatabase": 1422492450,
  "tld": "example",
  "service": "rdds",
  "cycleCalculationDateTime": 1422734490,
  "status": "Down",
  "testedInterface": [
    {
      "interface": "RDDS43",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": null,
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "2001:DB8::1",
                  "rtt": null,
                  "result": "-200"
                }
              ]
            }
          ]
        },
        {
          "city": "Sydney",
          "status": "Up",
          "testData": [
            {
              "target": null,
              "status": "Up",
              "metrics": [
                {
                  "testDateTime": 1422734508,
                  "targetIP": "192.0.2.1",
                  "rtt": 250,
                  "result": "ok"
                }
              ]
            }
          ]
        }
      ]
    },
    {
      "interface": "RDDS80",
      "probes": [
        {
          "city": "WashingtonDC",
          "status": "Down",
          "testData": [
            {
              "target": null,
              "status": "Down",
              "metrics": [
                {
                  "testDateTime": 1422734513,
                  "targetIP": "192.0.2.1",
                  "rtt": null,
                  "result": "-200"
                }
              ]
            }
          ]
        }
```

```
              ]
            },
            {
              "city": "Sydney",
              "status": "Down",
              "testData": [
                {
                  "target": null,
                  "status": "Down",
                  "metrics": [
                    {
                      "testDateTime": 1422734508,
                      "targetIP": "192.0.2.1",
                      "rtt": null,
                      "result": "-200"
                    }
                  ]
                }
              ]
            }
          ]
        }
      ]
    }
```

5.8.1. DNS/DNSSEC Monitoring error codes

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

Result Code	Obsolete	Internal Error	Interface	Description
-1	N	Y	DNS UDP / DNS TCP	Internal error.
-2	N	Y	DNS UDP	Expecting NOERROR RCODE but got unexpected RCODE from local resolver.
-3	N	Y	DNS TCP	Expecting NOERROR RCODE but got unexpected RCODE from local resolver.
-200	N	N	DNS UDP	No reply from the authoritative name server.
-201	Y	N	DNS UDP / DNS TCP	Invalid reply from Name Server.
-204	Y	N	DNS UDP / DNS TCP	DNSSEC error.
-206	Y	N	DNS UDP / DNS TCP	Keyset is not valid.
-207	N	N	DNS UDP	Expecting DNS class IN but got class CHAOS in the DNS response.
-208	N	N	DNS UDP	Expecting DNS class IN but got class HESIOD in the DNS response.
-209	N	N	DNS UDP	Expecting DNS class IN but got something different from class IN, CHAOS or HESIOD in the DNS response.
-210	N	N	DNS UDP	Header section incomplete in the DNS response.
-211	N	N	DNS UDP	Question section incomplete in the DNS response.
-212	N	N	DNS UDP	Answer section incomplete in the DNS response.
-213	N	N	DNS UDP	Authority section incomplete in the DNS response.
-214	N	N	DNS UDP	Additional section incomplete in the DNS response.
-215	N	N	DNS UDP	Malformed DNS response.
-250	N	N	DNS UDP	Querying for a non-existent domain - the AA flag is off (was expecting on) in the DNS response.
-251	N	N	DNS UDP	Querying for a non-existent domain - Domain name being queried not present in question section of the DNS response.

-252	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOERROR on the DNS response.
-253	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got FORMERR on the DNS response.
-254	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got SERVFAIL on the DNS response.
-255	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTIMP on the DNS response.
-256	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got REFUSED on the DNS response.
-257	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got YXDOMAIN on the DNS response.
-258	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got YXRRSET on the DNS response.
-259	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NXRRSET on the DNS response.
-260	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTAUTH on the DNS response.
-261	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTZONE on the DNS response.
-270	N	N	DNS UDP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got unexpected (i.e. 11-15) on the DNS response.
-400	N	N	DNS UDP	Timeout when waiting for a response from the TLD authoritative servers as reported by the local DNS resolver.
-401	N	N	DNS UDP	The TLD is configured as DNSSEC-enabled, but no DNSKEY was found in the apex.
-402	N	N	DNS UDP	DNSSEC error in the chain of trust from the root to the TLD apex.
-403	N	N	DNS UDP	The TLD was not found in the root.
-405	N	N	DNS UDP	Unknown cryptographic algorithm found in a DNSSEC signature.
-406	N	N	DNS UDP	Unsupported cryptographic algorithm found in a DNSSEC signature.

-407	N	N	DNS UDP	No RRSIGs were found, and the TLD is expected to be signed.
-408	N	N	DNS UDP	Querying for a non-existent domain - No NSEC/NSEC3 RRs were found in the authority section.
-410	N	N	DNS UDP	No signature covering the RRSET was found.
-414	N	N	DNS UDP	An RRSIG was found and it is not signed by a DNSKEY from the KEYSET.
-415	N	N	DNS UDP	Bogus DNSSEC signature was found.
-416	N	N	DNS UDP	An expired DNSSEC signature was found.
-417	N	N	DNS UDP	A DNSSEC signature with an inception date in the future was found.
-418	N	N	DNS UDP	A DNSSEC signature with expiration date earlier than inception date was found.
-422	N	N	DNS UDP	A resource record (RR) not covered by the given NSEC/NSEC3 RRs was found.
-425	N	N	DNS UDP	Malformed RRSIG with too few RDATA fields was found.
-427	N	N	DNS UDP	Malformed DNSSEC response.
-600	N	N	DNS TCP	Connection to the name server was successful, but the connection timed out.
-601	N	N	DNS TCP	Error when opening a connection to the name server.
-607	N	N	DNS TCP	Expecting DNS class IN but got CHAOS in the DNS response.
-608	N	N	DNS TCP	Expecting DNS class IN but got HESIOD in the DNS response.
-609	N	N	DNS TCP	Expecting DNS class IN but got something different from [IN, CHAOS or HESIOD] in the DNS response.
-610	N	N	DNS TCP	Header section incomplete in the DNS response.
-611	N	N	DNS TCP	Question section incomplete in the DNS response.
-612	N	N	DNS TCP	Answer section incomplete in the DNS response.
-613	N	N	DNS TCP	Authority section incomplete in the DNS response.
-614	N	N	DNS TCP	Additional section incomplete in the DNS response.
-615	N	N	DNS TCP	Malformed DNS response.
-650	N	N	DNS TCP	Querying for a non-existent domain - the AA flag is off (expecting on) in the DNS response.

-651	N	N	DNS TCP	Querying for a non-existent domain - Domain name being queried not present in question section of the DNS response.
-652	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOERROR on the DNS response.
-653	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got FORMERR on the DNS response.
-654	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got SERVFAIL on the DNS response.
-655	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTIMP on the DNS response.
-656	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got REFUSED on the DNS response.
-657	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got YXDOMAIN on the DNS response.
-658	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got YXRRSET on the DNS response.
-659	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NXRRSET on the DNS response.
-660	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTAUTH on the DNS response.
-661	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got NOTZONE on the DNS response.
-670	N	N	DNS TCP	Querying for a non-existent domain - Expecting NXDOMAIN RCODE but got unexpected (i.e. 11-15) on the DNS response.
-800	N	N	DNS TCP	Timeout when waiting for a response from the TLD authoritative servers as reported by the local DNS resolver.
-801	N	N	DNS TCP	The TLD is configured as DNSSEC-enabled, but no DNSKEY was found in the apex.
-802	N	N	DNS TCP	DNSSEC error in the chain of trust from the root zone to the TLD apex.
-803	N	N	DNS TCP	The TLD was not found in the root.
-805	N	N	DNS TCP	Unknown cryptographic algorithm found in a DNSSEC signature.

Result Code	Obsolete	Internal Error	Interface	Description
-806	N	N	DNS TCP	Unsupported cryptographic algorithm found in a DNSSEC signature.
-807	N	N	DNS TCP	No RRSIGs where found, and the TLD is expected to be signed.
-808	N	N	DNS TCP	Querying for a non-existent domain - No NSEC/NSEC3 RRs were found in the authority section
-810	N	N	DNS TCP	No signature covering the RRSET was found.
-814	N	N	DNS TCP	An RRSIG was found and it is not signed by a DNSKEY from the KEYSET.
-815	N	N	DNS TCP	Bogus DNSSEC signature was found.
-816	N	N	DNS TCP	An expired DNSSEC signature was found.
-817	N	N	DNS TCP	A DNSSEC signature with an inception date in the future was found.
-818	N	N	DNS TCP	A DNSSEC signature with expiration date earlier than inception date was found.
-822	N	N	DNS TCP	A RR not covered by the given NSEC/NSEC3 RRs was found.
-825	N	N	DNS TCP	Malformed RRSIG with too few RDATA fields was found.
-827	N	N	DNS TCP	Malformed DNSSEC response.

Note: error codes marked as Obsolete are listed for documentation purposes.

Note: a test with an error code marked as Internal Error will be considered to be UP.

5.8.2. RDDS Monitoring error codes

The following table lists the error codes for RDDS monitoring:

Result Code	Obsolete	Internal Error	Interface	Description
-1	N	Y	Whois-43 / Web-whois	Internal Error
-2	N	Y	Whois-43 / Web-whois	RDDS service could not be tested due to lack of IPv4/6 transport in the probe node.
-3	N	Y	Whois-43	Expecting NOERROR RCODE but got unexpected code when resolving the WHOIS-43 hostname using the local DNS resolver.
-4	N	Y	Web-whois	Expecting NOERROR RCODE but got unexpected code when resolving web-whois hostname using the local DNS resolver.

				Connection timed out while trying to get a
-200	Y	N	Whois-43	response from the server.
-201	N	N	Whois-43	Syntax error while parsing the WHOIS-43 response.
-204	Y	N	Web-whois	Connection timed out while trying to get a response from the server.
-205	Y	N	Whois-43 / Web-whois	Error when trying to resolve the Whois server hostname (e.g. whois.nic.example).
-206	N	N	Web-whois	An HTTP status code was not found in the HTTP message.
-207	Y	N	Web-whois	No HTTP/200 status code in response (after following redirects).
-222	N	N	Whois-43	Timeout when waiting for a response from the TLD authoritative servers as reported by the local DNS resolver.
-224	N	N	Whois-43	DNSSEC error when trying to resolve the hostname for the WHOIS-43 server.
-225	N	N	Whois-43	The hostname for the WHOIS-43 server was not found in the DNS.
-227	N	N	Whois-43	Connection to WHOIS-43 server was successful, but the connection timed out.
-228	N	N	Whois-43	Connection to WHOIS-43 server was unsuccessful.
-229	N	N	Whois-43	Empty response received from WHOIS-43 server.
-250	N	N	Web-whois	Timeout when waiting for a response from the TLD authoritative servers as reported by the local DNS resolver.
-252	N	N	Web-whois	DNSSEC error when trying to resolve the hostname for the web-whois server.
-253	N	N	Web-whois	The hostname for the web-whois server was not found in the DNS.
-255	N	N	Web-whois	Connection to the web-whois server was successful, but the connection timed out.
-256	N	N	Web-whois	Error when opening a connection to web-whois server.
-257	N	N	Web-whois	Malformed HTTP message.
-258	N	N	Web-whois	Malformed HTTP message or TLS general error.
-259	N	N	Web-whois	The maximum number of HTTP redirects (301, 302 and 303) were followed, and a 200 / HTTP status code was not found.
-300	N	N	Web-whois	Expecting HTTP status code 200 but got 100.

			Web-	
-301	N	N	whois	Expecting HTTP status code 200 but got 101.
-302	N	N	Web-whois	Expecting HTTP status code 200 but got 102.
-303	N	N	Web-whois	Expecting HTTP status code 200 but got 103.
-304	N	N	Web-whois	Expecting HTTP status code 200 but got 201.
-305	N	N	Web-whois	Expecting HTTP status code 200 but got 202.
-306	N	N	Web-whois	Expecting HTTP status code 200 but got 203.
-307	N	N	Web-whois	Expecting HTTP status code 200 but got 204.
-308	N	N	Web-whois	Expecting HTTP status code 200 but got 205.
-309	N	N	Web-whois	Expecting HTTP status code 200 but got 206.
-310	N	N	Web-whois	Expecting HTTP status code 200 but got 207.
-311	N	N	Web-whois	Expecting HTTP status code 200 but got 208.
-312	N	N	Web-whois	Expecting HTTP status code 200 but got 226.
-313	N	N	Web-whois	Expecting HTTP status code 200 but got 300.
-317	N	N	Web-whois	Expecting HTTP status code 200 but got 304.
-318	N	N	Web-whois	Expecting HTTP status code 200 but got 305.
-319	N	N	Web-whois	Expecting HTTP status code 200 but got 306.
-320	N	N	Web-whois	Expecting HTTP status code 200 but got 307.
-321	N	N	Web-whois	Expecting HTTP status code 200 but got 308.
-322	N	N	Web-whois	Expecting HTTP status code 200 but got 400.
-323	N	N	Web-whois	Expecting HTTP status code 200 but got 401.
-324	N	N	Web-whois	Expecting HTTP status code 200 but got 402.
-325	N	N	Web-whois	Expecting HTTP status code 200 but got 403.
-326	N	N	Web-whois	Expecting HTTP status code 200 but got 404.

			Web-	
-327	N	N	whois	Expecting HTTP status code 200 but got 405.
-328	N	N	Web-whois	Expecting HTTP status code 200 but got 406.
-329	N	N	Web-whois	Expecting HTTP status code 200 but got 407.
-330	N	N	Web-whois	Expecting HTTP status code 200 but got 408.
-331	N	N	Web-whois	Expecting HTTP status code 200 but got 409.
-332	N	N	Web-whois	Expecting HTTP status code 200 but got 410.
-333	N	N	Web-whois	Expecting HTTP status code 200 but got 411.
-334	N	N	Web-whois	Expecting HTTP status code 200 but got 412.
-335	N	N	Web-whois	Expecting HTTP status code 200 but got 413.
-336	N	N	Web-whois	Expecting HTTP status code 200 but got 414.
-337	N	N	Web-whois	Expecting HTTP status code 200 but got 415.
-338	N	N	Web-whois	Expecting HTTP status code 200 but got 416.
-339	N	N	Web-whois	Expecting HTTP status code 200 but got 417.
-340	N	N	Web-whois	Expecting HTTP status code 200 but got 421.
-341	N	N	Web-whois	Expecting HTTP status code 200 but got 422.
-342	N	N	Web-whois	Expecting HTTP status code 200 but got 423.
-343	N	N	Web-whois	Expecting HTTP status code 200 but got 424.
-344	N	N	Web-whois	Expecting HTTP status code 200 but got 426.
-345	N	N	Web-whois	Expecting HTTP status code 200 but got 428.
-346	N	N	Web-whois	Expecting HTTP status code 200 but got 429.
-347	N	N	Web-whois	Expecting HTTP status code 200 but got 431.
-348	N	N	Web-whois	Expecting HTTP status code 200 but got 451.
-349	N	N	Web-whois	Expecting HTTP status code 200 but got 500.

-350	N	N	Web-whois	Expecting HTTP status code 200 but got 501.
-351	N	N	Web-whois	Expecting HTTP status code 200 but got 502.
-352	N	N	Web-whois	Expecting HTTP status code 200 but got 503.
-353	N	N	Web-whois	Expecting HTTP status code 200 but got 504.
-354	N	N	Web-whois	Expecting HTTP status code 200 but got 505.
-355	N	N	Web-whois	Expecting HTTP status code 200 but got 506.
-356	N	N	Web-whois	Expecting HTTP status code 200 but got 507.
-357	N	N	Web-whois	Expecting HTTP status code 200 but got 508.
-358	N	N	Web-whois	Expecting HTTP status code 200 but got 510.
-359	N	N	Web-whois	Expecting HTTP status code 200 but got 511.
-360	N	N	Web-whois	Expecting HTTP status code 200 but got an unexpected status code.

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

Note: error codes marked as Obsolete are listed for documentation purposes.

Note: a test with an error code marked as an Internal Error will be considered to be UP.

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 MoSAPI. The schedule object contains the following fields (**required**):

- "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. The maximum length is 255 Unicode characters. Note: Unicode characters beyond the 255 limit will be ignored.

- "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. The maximum length is 255 Unicode characters. Note: Unicode characters beyond the 255 limit will be ignored.

- "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".

Example using CURL to create a maintenance window:

```
curl --cookie cookies.txt -H "Content-Type: application/json"
https://mosapi.icann.org/mosapi/v1/example/mntWin/RDDS/16beaa07-46a3-42eb-9e71-
c2e06cfd8a9b -X PUT -d \
'{
"enable": "true",
"name": "Maintenance window for RDDS semester II-2017",
"description": "Pre-planned maintenance window for RDDS",
"startTime": "1512003600",
"endTime": "1512006600",
"version": "1"
}'
```

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".

Example using CURL to delete a maintenance window:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/mntWin/RDDS/16beaa07-46a3-42eb-9e71-
c2e06cfd8a9b -X DELETE
```

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.

Example using CURL to request the details of a maintenance window:

```
curl --cookie cookies.txt
https://mosapi.icann.org/mosapi/v1/example/mntWin/RDDS/16beaa
07-46a3-42eb-9e71-c2e06cfd8a9b
```

Example of JSON response for a maintenance window details request:

```
{
    "enable": "true",
    "name": "Maintenance window for RDDS semester II-2017",
    "description": "Pre-planned maintenance window for RDDS",
    "startTime": "1512003600",
    "endTime": "1512006600",
    "version": "1"
}
```

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 object identifiers 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"
[INCLUDE REST OF ELEMENTS]
    }, {
        "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:

    o "city", a JSON string that contains the location of the probe node.

    o "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.

    o "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:

```json
{
  "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>/<schedul eID>	•	•	•	The UUID syntax is incorrect.
2002	<base_url>/mntWin/ <service>/<schedul eID>	•			The maintenance window start date and time is not 24 hours ahead of the current date and time.
2003	<base_url>/mntWin/ <service>/<schedul eID>	•			The period specified by start and end date and time is greater than the monthly SLR for the service.
2004	<base_url>/mntWin/ <service>/<schedul eID>	•			The period specified in the maintenance window collides with a previously scheduled maintenance window for the service.
2005	<base_url>/mntWin/ <service>/<schedul eID>	•	•	•	The maintenance window functionality is disabled for this TLD.
2006	<base_url>/mntWin/ <service>/<schedul eID>		•		The maintenance window that you are trying to delete already started.
2007	<base_url>/mntWin/ <service>/<schedul eID>	•			The endTime is in the past, before or equal to the startTime.
2008	<base_url>/mntWin/ <service>/<schedul eID>	•			The startTime syntax is incorrect.
2009	<base_url>/mntWin/ <service>/<schedul eID>	•			The endTime syntax is incorrect.
2010	<base_url>/mntWin/ <service>/<schedul eID>	•			The maintenance window that you are trying to update already ended, updates are not allowed.
2011	<base_url>/monitor ing/<service>/inci dents			•	The difference between endDate and startDate is more than 31 days.
2012	<base_url>/monitor ing/<service>/inci dents			•	The endDate is before the startDate.
2013	<base_url>/monitor ing/<service>/inci dents			•	The startDate syntax is incorrect.
2014	<base_url>/monitor ing/<service>/inci dents			•	The endDate syntax is incorrect.
2015	<base_url>/monitor ing/<service>/inci dents			•	The value of falsePositive is invalid.
2016	<base_url>/mntWin/ <service>/<schedul eID>	•			The value of name or description cannot be blank.
2100	<base_url>/mntWin/ <service>/<schedul eID>	•			The JSON syntax is invalid.
2101	<base_url>/mntWin/ <service>/<schedul eID>	•			The maintenance window that you are trying to update already started, only enabled and endTime fields can be modified.