ICANN Registry System Testing (RST) API

RST-API Specification Home Page

Last updated: 2024-03-26

The Registry System Testing (RST-API) provides a RESTful interface to ICANN's Registry System Testing platform, which is used to conduct conformance tests of critical registry functions at various points during the lifecycle of a gTLD (before initial delegation, before the transition to a new Registry Service Provider, or before the approval of new registry services). It will also used by the forthcoming Registry Service Provider (RSP) Pre-Evaluation Program.

Workflow overview

The sequence diagram below describes the process by which tests are scheduled, configured, and executed, in the context of the RSP Evaluation Program:

- High-level workflow

State diagram

Each test request object has a `status` property (see the `testStatus` schema below) indicating its position in the test lifecycle. The following state diagram describes this lifecycle:

- State diagram

Role-based access control

This API implements Role-Based Access Control, where access to certain operations is restricted based on the role that is assigned to a user. For example, external users cannot create new test request objects in the production environment, but can in OT&E.

Authentication

All access to the API is authenticated using TLS certificates that are authenticated using `TLSA` records published in the DNS. Test request objects are associated with DNS hostnames; if a user presents a certificate which matches one of the `TLSA` records published in the DNS at one of these hostnames, it will be permitted to perform operations on that object.

Internal users

Internal users must use a certificate that matches a `TLSA` record found at a hard-coded DNS hostname *(the exact name is yet to be determined)*.

HTTP status codes

In addition to the HTTP status codes described in the operations specifications below, all operations may respond to requests with one of the following HTTP status codes:

- **400:** returned when the server receives a malformed request.
- **403:** returned when the access control policy prevents access.
- **404:** returned when the resource does not exist.
- **405:** returned when the request method is invalid.
- **409:** returned when the client attempts to overwrite an resource that already exists.
- **429:** returned when the client has exceeded rate limits.
- **504:** returned when an intermediate proxy experiences an error.
- **500:** returned when there has been an internal server error.

Change Log

- 2024-03-20:
  - the `id` property of `testRequestSubmitted` objects is now a string, not and integer.
- 2024-03-13:
  - Variant labels in an IDN table may have exceptional allocation policies when allocated in the same TLD as the primary label, or in a variant TLD.
- 2024-03-06:
  - the `caseId` property of the `testCaseLog` object, and the `code` property of the `testCaseLogMessage` object are now enums, which are generated from the test specs.

- 2024-02-28:
    - As per the last release, only a single YAML file is now built from the source file, that contains both "internal" and "external" endpoints. The "internal" view is no longer published.
    - To work better with code generators, the following changes have been made:
        - The specification now conforms to v3.0.3 of the OpenAPI specification instead of v3.1.0. This means that many aspects of the API specification and object schemas (including those of input parameters) have been changed to avoid using features only available in v3.1.0 of the OpenAPI spec. This includes the `examples` property for all JSON types, which means that the examples shown in the HTML representation of the API spec are now less useful than they were previously.
        - The following changes have been made to `testCaseLog` objects:
            - the `code` and `codeRef` properties are now optional instead of nullable.
            - the `context` property has been changed so that property values are always strings.
        - The `securitySchemes` property has been removed, since code generators don't seem to offer good support for mutual TLS. The requirement for client TLS authentication has not changed, however.
        - All usages of the `patternProperties` feature of JSON Schema have been changed to avoid their use. Some may still be present in the input parameters.
    - Changes to IDN table objects:
        - the `variants` property of entries in the `validLabels` property of `idnTable` objects has been changed to an array of objects, so the corresponding language tag can be included.
        - the `lgrXML` property of `idnTable` object has been removed.
        - Renamed the `supportLevel` property to `variantSupportLevel`.
        - The operation to create a new IDN table object uses `idnTableRequest` as the request body payload, which does not allow for inclusion of server-generated object properties.
        - In OT&E, only Reference Second-Level LGRs can be used.
    - `type` properties for `enum` types have been reinstated.
- 2024-02-21:
    - Test properties (such as `applicationId` and `rsp`) that were previously ignored in OT&E now **MUST** be omitted.
    - When IDN table objects are created in OT&E, they **MUST** have a `isReferenceLGR` property that is `true`.
    - Simplified object schemas by marking properties as required (and others as therefore optional) avoiding the need to have nullable properties.
    - IDN tables are now referenced by a unique ID rather than the `{rsp, tag, version}` triple.
    - The internal and external views are now identical. The separate files will be removed in the next release.
    - Added the `supportLevel` property to the `idnTableRef` type.
    - Some IDN table management endpoints are now accessible to external clients.
    - Make access control policies clearer and more consistent.
    - Remove `DELETE /tests/{id}` endpoint.
    - Fix schema definition for the `Location` header in `POST /test` responses.
    - Changes only relevant to internal users:
        - Change the `client` parameter to `GET /tests` to `rsp`.
        - Reinstate query parameters for `GET /tables`.
        - The `PATCH /test/{id}` endpoint has been replaced with `POST /test/{id}/run` and `POST /test/{id}/result`.
        - Simplified IDN table management, so that test labels are provided when the table object is created.
- 2024-02-14:
    - Switch to a weekly release cycle.
    - Use a date-based version number instead of a commit-based version, ahead of switching to a weekly release cycle.
- 2024-01-31:
    - Add this change log.
    - Minimise the delta between the internal and external view.

More information: https://openapi-generator.tech
Contact Info: globalsupport@icann.org
Version: 1.2024086
BasePath:/v1

http://apache.org/licenses/LICENSE-2.0.html

Access

Methods

[Jump to Models]

Table of Contents

CreatingTestRequestObjects

POST /test

Up

(createTest)

This operation creates a new test request object.

This operation is not available to external users in production, but may be used in OT&E.

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

testRequest testRequest (optional)
Body Parameter —

Return type

testRequestSubmitted

Example data

Content-Type: application/json

```
{
  "testPlanVersion" : "testPlanVersion",
  "inputs" : {
    "dnssecOps.zskRolloverZone" : "dnssecOps.zskRolloverZone",
    "dnssecOps.primaryServers" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
```

```
},
"epp.registeredNames" : [ "epp.registeredNames", "epp.registeredNames" ],
"dnssecOps.kskRolloverZone" : "dnssecOps.kskRolloverZone",
"epp.secDNSInterfaces" : "dsData",
"minimumRPMS.sunriseModels" : "start-date",
"epp.clid02DataModel" : "minimum",
"rde.publicKey" : "rde.publicKey",
"srsgw.eppHostName" : "srsgw.eppHostName",
"general.registryDataModel" : "minimum",
"integration.rriACL" : {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
},
"epp.requiredContactTypes" : [ "admin", "admin" ],
"srsgw.eppClid01" : "srsgw.eppClid01",
"srsgw.eppClid02" : "srsgw.eppClid02",
"rdap.testDomains" : [ "rdap.testDomains", "rdap.testDomains" ],
"rde.signatureFile" : "rde.signatureFile",
"srsgw.eppPwd01" : "srsgw.eppPwd01",
"srsgw.eppPwd02" : "srsgw.eppPwd02",
"epp.registeredContacts" : [ "epp.registeredContacts", "epp.registeredContacts" ],
"epp.hostModel" : "objects",
"integration.rdeSFTPUsername" : "integration.rdeSFTPUsername",
"epp.greeting" : "epp.greeting",
"srsgw.rdapBaseURLs" : [ {
  "baseURL" : "http://example.com/aeiou",
  "tld" : "tld"
}, {
  "baseURL" : "http://example.com/aeiou",
  "tld" : "tld"
} ],
"dns.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.clid01DataModel" : "minimum",
"epp.serverIssuedClientCertificate02" : "epp.serverIssuedClientCertificate02",
"epp.serverIssuedClientCertificate01" : "epp.serverIssuedClientCertificate01",
"rde.depositFile" : "rde.depositFile",
"dnssecOps.tsigKey" : {
  "name" : "name",
  "secret" : "secret",
  "algorithm" : "hmac-sha256"
},
"rdap.testNameservers" : [ {
  "nameserver" : "nameserver",
  "tld" : "tld"
}, {
  "nameserver" : "nameserver",
  "tld" : "tld"
} ],
"dnssec.dsRecords" : [ {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
}, {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
```

```
      "digest" : "digest",
      "alg" : 6
    }, {
      "keyTag" : 5,
      "digestType" : 1,
      "digest" : "digest",
      "alg" : 6
    } ],
    "name" : "name"
  } ],
  "epp.hostName" : "epp.hostName",
  "srsgw.serverIssuedClientCertificate01" : "srsgw.serverIssuedClientCertificate01",
  "epp.registeredNameservers" : [ "epp.registeredNameservers", "epp.registeredNameservers" ],
  "srsgw.serverIssuedClientCertificate02" : "srsgw.serverIssuedClientCertificate02",
  "integration.rdeSFTPHostname" : "integration.rdeSFTPHostname",
  "dnssecOps.nameservers" : [ {
    "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
    "name" : "name",
    "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
  }, {
    "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
    "name" : "name",
    "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
  } ],
  "epp.restoreReportRequired" : true,
  "dnssecOps.algorithmRolloverZone" : "dnssecOps.algorithmRolloverZone",
  "epp.pwd02" : "epp.pwd02",
  "epp.pwd01" : "epp.pwd01",
  "integration.rdeSFTPDirectory" : "integration.rdeSFTPDirectory",
  "dnssecOps.csk" : true,
  "epp.clid02" : "epp.clid02",
  "epp.clid01" : "epp.clid01",
  "minimumRPMS.sunriseTLD" : "minimumRPMS.sunriseTLD",
  "rdap.baseURLs" : [ {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  }, {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  } ],
  "minimumRPMS.claimsTLD" : "minimumRPMS.claimsTLD",
  "rdap.testEntities" : [ {
    "handle" : "handle",
    "tld" : "tld"
  }, {
    "handle" : "handle",
    "tld" : "tld"
  } ]
},
"missingInputs" : [ "missingInputs", "missingInputs" ],
"dueDate" : "2000-01-23T04:56:07.000+00:00",
"dateRequested" : "2000-01-23T04:56:07.000+00:00",
"errorCodes" : [ "errorCodes", "errorCodes" ],
"clientIDs" : [ "clientIDs", "clientIDs" ],
"rsp" : "rsp",
"dateUpdated" : "2000-01-23T04:56:07.000+00:00",
"tlds" : [ [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
```

```
    } ]
  } ], [ {
    "name" : "name",
    "idnTables" : [ {
      "id" : "id",
      "variantSupportLevel" : 0
    }, {
      "id" : "id",
      "variantSupportLevel" : 0
    } ]
  }, {
    "name" : "name",
    "idnTables" : [ {
      "id" : "id",
      "variantSupportLevel" : 0
    }, {
      "id" : "id",
      "variantSupportLevel" : 0
    } ]
  } ] ],
  "dateStarted" : "2000-01-23T04:56:07.000+00:00",
  "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
  "testPlan" : "StandardPreDelegationTest",
  "files" : [ {
    "name" : "name",
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  }, {
    "name" : "name",
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  } ],
  "testID" : "testID",
  "applicationID" : "applicationID",
  "results" : [ {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
```

```
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  }, {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  } ]
}
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
A successful result. testRequestSubmitted

IDNTableManagement

POST /table

Up

(**createIDNTable**)

This operation creates a new IDN table object.

IDN table objects must be created before they can be referenced in a test request.

This operation is not available to external users in production, but may be used in OT&E.

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

idnTableRequest idnTableRequest **(optional)**
Body Parameter —

Responses
201
A successful result.

DELETE /table/{id}

Up

(**deleteIDNTable**)

This operation deletes an IDN table object.

External users can only access IDN tables that are linked to test objects associated with their credentials.

This operation is not available to external users in production, but may be used in OT&E.

Internal users can perform this operation on any object.

Path parameters

id (required)
Path Parameter — the table ID. default: null

Responses
201
A successful result.

GET /table/{id}

Up

(**getIDNTable**)

This operation returns information about an IDN table object.

External users can only access IDN tables that are linked to test objects associated with their credentials.

Internal users can perform this operation on any object.

Path parameters

id (required)
Path Parameter — the table ID. default: null

Return type
idnTable

Example data
Content-Type: application/json


```
{
  "variantPolicy" : "novar",
  "testLabels" : {
    "validLabels" : [ {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    }, {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    } ],
    "invalidLabels" : [ "invalidLabels", "invalidLabels" ]
  },
  "tableID" : "tableID",
  "active" : true,
  "tag" : "tag",
  "isReferenceLGR" : true,
  "version" : "version",
  "rsp" : "rsp"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
A successful result. idnTable

Up

`GET /tables`

(getIDNTables)

This operation returns the IDN tables matching the provided query parameters.

External users can only access IDN tables that are linked to test objects associated with their credentials.

Internal users can perform this operation on any object.

Query parameters

rsp (optional)
Query Parameter — the RSP ID (internal users only). default: null

tag (optional)
Query Parameter — the language tag. default: null

Return type
array[idnTable]

Example data
Content-Type: application/json


```
[ {
  "variantPolicy" : "novar",
  "testLabels" : {
    "validLabels" : [ {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    }, {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    } ],
    "invalidLabels" : [ "invalidLabels", "invalidLabels" ]
  },
  "tableID" : "tableID",
  "active" : true,
  "tag" : "tag",
  "isReferenceLGR" : true,
  "version" : "version",
  "rsp" : "rsp"
}, {
  "variantPolicy" : "novar",
  "testLabels" : {
    "validLabels" : [ {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
```

```
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    }, {
      "label" : "label",
      "variants" : [ {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      }, {
        "variantPolicy" : {
          "inVariantTLD" : "allblockvar",
          "inSameTLD" : "allblockvar"
        },
        "label" : "label",
        "tag" : "tag"
      } ]
    } ],
    "invalidLabels" : [ "invalidLabels", "invalidLabels" ]
  },
  "tableID" : "tableID",
  "active" : true,
  "tag" : "tag",
  "isReferenceLGR" : true,
  "version" : "version",
  "rsp" : "rsp"
} ]
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
A successful result.

`PUT /table/{id}`

Up

(**updateIDNTable**)

This operation updates an existing IDN table object.

External users can only access IDN tables that are linked to test objects associated with their credentials.

This operation is not available to external users in production, but may be used in OT&E.

Path parameters

id (required)
Path Parameter — the table ID. default: null

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

idnTableRequest idnTableRequest **(optional)**
Body Parameter —

Responses

201
A successful result.

ProvidingInputParametersFiles

`POST /test/{id}/inputs`

Up

(setTestInputParameters)

This operation submits test input parameters. Parameters in the payload will previously submitted values. Values that are present in the object but not present in the payload will not be modified.

Input parameters can only be submitted for test request objects that have the status of `inputs-needed`.

Users can only perform this operation if their certificate matches a `TLSA` record published in the DNS at one of the hostnames specified in the `clientIDs` property of the test request object.

Internal users can perform this operation on any object.

Once all required input parameters have been submitted (and any files referenced in those parameters have been uploaded), the status of the test request object will change from `input-needed` to `inputs-complete`.

Path parameters

id (required)
Path Parameter — the test ID default: null

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

inputParameters inputParameters **(optional)**
Body Parameter —

Responses
201
A successful result.

`POST /test/{id}/files`

Up

(uploadFile)

This resource may be used to upload files. Multiple files may be uploaded in a single request. If a filename matches a previously submitted file, that file will be replaced.

All files MUST be referenced in an input parameter **before** being uploaded.

Users can only perform this operation if their certificate matches a `TLSA` record published in the DNS at one of the hostnames specified in the `clientIDs` property of the test request object.

Internal users can perform this operation on any object.

Path parameters

id (required)
Path Parameter — the test ID default: null

Consumes
This API call consumes the following media types via the Content-Type request header:

- `multipart/form-data`

Form parameters

file (optional)
Form Parameter — default: null format: binary

Responses
201
A successful result.

RetrievingTestInformation

Up

`GET /test/{id}/file/{file}`

(getFile)

This retrieves an uploaded file.

Users can only perform this operation if their certificate matches a `TLSA` record published in the DNS at one of the hostnames specified in the `clientIDs` property of the test request object.

Internal users can perform this operation on any object.

Path parameters

id (required)
Path Parameter — the test ID default: null

file (required)
Path Parameter — the file name default: null

Return type
File

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/octet-stream`

Responses
200
A successful result. File

Up

`GET /test/{id}`

(getTestInfo)

This operation returns information about a specific test request object.

Users can only perform this operation if their certificate matches a `TLSA` record published in the DNS at one of the hostnames specified in the `clientIDs` property of the test request object.

Internal users can perform this operation on any object.

Path parameters

id (required)
Path Parameter — the test ID default: null

Return type
testRequestSubmitted

Example data
Content-Type: application/json


```
{
  "testPlanVersion" : "testPlanVersion",
  "inputs" : {
    "dnssecOps.zskRolloverZone" : "dnssecOps.zskRolloverZone",
    "dnssecOps.primaryServers" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
    },
    "epp.registeredNames" : [ "epp.registeredNames", "epp.registeredNames" ],
    "dnssecOps.kskRolloverZone" : "dnssecOps.kskRolloverZone",
```

```
"epp.secDNSInterfaces" : "dsData",
"minimumRPMS.sunriseModels" : "start-date",
"epp.clid02DataModel" : "minimum",
"rde.publicKey" : "rde.publicKey",
"srsgw.eppHostName" : "srsgw.eppHostName",
"general.registryDataModel" : "minimum",
"integration.rriACL" : {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
},
"epp.requiredContactTypes" : [ "admin", "admin" ],
"srsgw.eppClid01" : "srsgw.eppClid01",
"srsgw.eppClid02" : "srsgw.eppClid02",
"rdap.testDomains" : [ "rdap.testDomains", "rdap.testDomains" ],
"rde.signatureFile" : "rde.signatureFile",
"srsgw.eppPwd01" : "srsgw.eppPwd01",
"srsgw.eppPwd02" : "srsgw.eppPwd02",
"epp.registeredContacts" : [ "epp.registeredContacts", "epp.registeredContacts" ],
"epp.hostModel" : "objects",
"integration.rdeSFTPUsername" : "integration.rdeSFTPUsername",
"epp.greeting" : "epp.greeting",
"srsgw.rdapBaseURLs" : [ {
  "baseURL" : "http://example.com/aeiou",
  "tld" : "tld"
}, {
  "baseURL" : "http://example.com/aeiou",
  "tld" : "tld"
} ],
"dns.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.clid01DataModel" : "minimum",
"epp.serverIssuedClientCertificate02" : "epp.serverIssuedClientCertificate02",
"epp.serverIssuedClientCertificate01" : "epp.serverIssuedClientCertificate01",
"rde.depositFile" : "rde.depositFile",
"dnssecOps.tsigKey" : {
  "name" : "name",
  "secret" : "secret",
  "algorithm" : "hmac-sha256"
},
"rdap.testNameservers" : [ {
  "nameserver" : "nameserver",
  "tld" : "tld"
}, {
  "nameserver" : "nameserver",
  "tld" : "tld"
} ],
"dnssec.dsRecords" : [ {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
}, {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
```

```
      "keyTag" : 5,
      "digestType" : 1,
      "digest" : "digest",
      "alg" : 6
    } ],
    "name" : "name"
  } ],
  "epp.hostName" : "epp.hostName",
  "srsgw.serverIssuedClientCertificate01" : "srsgw.serverIssuedClientCertificate01",
  "epp.registeredNameservers" : [ "epp.registeredNameservers", "epp.registeredNameservers" ],
  "srsgw.serverIssuedClientCertificate02" : "srsgw.serverIssuedClientCertificate02",
  "integration.rdeSFTPHostname" : "integration.rdeSFTPHostname",
  "dnssecOps.nameservers" : [ {
    "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
    "name" : "name",
    "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
  }, {
    "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
    "name" : "name",
    "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
  } ],
  "epp.restoreReportRequired" : true,
  "dnssecOps.algorithmRolloverZone" : "dnssecOps.algorithmRolloverZone",
  "epp.pwd02" : "epp.pwd02",
  "epp.pwd01" : "epp.pwd01",
  "integration.rdeSFTPDirectory" : "integration.rdeSFTPDirectory",
  "dnssecOps.csk" : true,
  "epp.clid02" : "epp.clid02",
  "epp.clid01" : "epp.clid01",
  "minimumRPMS.sunriseTLD" : "minimumRPMS.sunriseTLD",
  "rdap.baseURLs" : [ {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  }, {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  } ],
  "minimumRPMS.claimsTLD" : "minimumRPMS.claimsTLD",
  "rdap.testEntities" : [ {
    "handle" : "handle",
    "tld" : "tld"
  }, {
    "handle" : "handle",
    "tld" : "tld"
  } ]
},
"missingInputs" : [ "missingInputs", "missingInputs" ],
"dueDate" : "2000-01-23T04:56:07.000+00:00",
"dateRequested" : "2000-01-23T04:56:07.000+00:00",
"errorCodes" : [ "errorCodes", "errorCodes" ],
"clientIDs" : [ "clientIDs", "clientIDs" ],
"rsp" : "rsp",
"dateUpdated" : "2000-01-23T04:56:07.000+00:00",
"tlds" : [ [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
} ], [ {
  "name" : "name",
```

```
    "idnTables" : [ {
      "id" : "id",
      "variantSupportLevel" : 0
    }, {
      "id" : "id",
      "variantSupportLevel" : 0
    } ]
  }, {
    "name" : "name",
    "idnTables" : [ {
      "id" : "id",
      "variantSupportLevel" : 0
    }, {
      "id" : "id",
      "variantSupportLevel" : 0
    } ]
  } ] ],
  "dateStarted" : "2000-01-23T04:56:07.000+00:00",
  "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
  "testPlan" : "StandardPreDelegationTest",
  "files" : [ {
    "name" : "name",
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  }, {
    "name" : "name",
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  } ],
  "testID" : "testID",
  "applicationID" : "applicationID",
  "results" : [ {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
```

```
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  }, {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  } ]
}
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

A successful result. testRequestSubmitted

GET /tests

Up

(getTests)

This operation performs a search on the database and returns all matching results.

External users will only see results where their certificate matches a `TLSA` record published in the DNS at one of the hostnames specified in the `clientIDs` property of the test request object.

Internal users will see results for all users.

Query parameters

rsp (optional)
Query Parameter — limit results to a specific RSP (internal users only). default: null

tld (optional)
Query Parameter — limit results to a specific TLD (internal users only). default: null

applicationID (optional)
Query Parameter — limit results to specific a application ID (internal users only). default: null

status (optional)
Query Parameter — limit results to those with the given status. default: null

result (optional)
Query Parameter — limit results to those with the given result. default: null

Return type
array[testRequestSubmitted]

Example data
Content-Type: application/json


```
[ {
  "testPlanVersion" : "testPlanVersion",
  "inputs" : {
    "dnssecOps.zskRolloverZone" : "dnssecOps.zskRolloverZone",
    "dnssecOps.primaryServers" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
    },
    "epp.registeredNames" : [ "epp.registeredNames", "epp.registeredNames" ],
    "dnssecOps.kskRolloverZone" : "dnssecOps.kskRolloverZone",
    "epp.secDNSInterfaces" : "dsData",
    "minimumRPMS.sunriseModels" : "start-date",
    "epp.clid02DataModel" : "minimum",
    "rde.publicKey" : "rde.publicKey",
    "srsgw.eppHostName" : "srsgw.eppHostName",
    "general.registryDataModel" : "minimum",
    "integration.rriACL" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
    },
    "epp.requiredContactTypes" : [ "admin", "admin" ],
    "srsgw.eppClid01" : "srsgw.eppClid01",
    "srsgw.eppClid02" : "srsgw.eppClid02",
    "rdap.testDomains" : [ "rdap.testDomains", "rdap.testDomains" ],
    "rde.signatureFile" : "rde.signatureFile",
    "srsgw.eppPwd01" : "srsgw.eppPwd01",
    "srsgw.eppPwd02" : "srsgw.eppPwd02",
    "epp.registeredContacts" : [ "epp.registeredContacts", "epp.registeredContacts" ],
    "epp.hostModel" : "objects",
    "integration.rdeSFTPUsername" : "integration.rdeSFTPUsername",
    "epp.greeting" : "epp.greeting",
    "srsgw.rdapBaseURLs" : [ {
      "baseURL" : "http://example.com/aeiou",
      "tld" : "tld"
    }, {
      "baseURL" : "http://example.com/aeiou",
      "tld" : "tld"
```

```
} ],
"dns.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.clid01DataModel" : "minimum",
"epp.serverIssuedClientCertificate02" : "epp.serverIssuedClientCertificate02",
"epp.serverIssuedClientCertificate01" : "epp.serverIssuedClientCertificate01",
"rde.depositFile" : "rde.depositFile",
"dnssecOps.tsigKey" : {
  "name" : "name",
  "secret" : "secret",
  "algorithm" : "hmac-sha256"
},
"rdap.testNameservers" : [ {
  "nameserver" : "nameserver",
  "tld" : "tld"
}, {
  "nameserver" : "nameserver",
  "tld" : "tld"
} ],
"dnssec.dsRecords" : [ {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
}, {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
} ],
"epp.hostName" : "epp.hostName",
"srsgw.serverIssuedClientCertificate01" : "srsgw.serverIssuedClientCertificate01",
"epp.registeredNameservers" : [ "epp.registeredNameservers", "epp.registeredNameservers" ],
"srsgw.serverIssuedClientCertificate02" : "srsgw.serverIssuedClientCertificate02",
"integration.rdeSFTPHostname" : "integration.rdeSFTPHostname",
"dnssecOps.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.restoreReportRequired" : true,
"dnssecOps.algorithmRolloverZone" : "dnssecOps.algorithmRolloverZone",
"epp.pwd02" : "epp.pwd02",
"epp.pwd01" : "epp.pwd01",
"integration.rdeSFTPDirectory" : "integration.rdeSFTPDirectory",
"dnssecOps.csk" : true,
```

```
  "epp.clid02" : "epp.clid02",
  "epp.clid01" : "epp.clid01",
  "minimumRPMS.sunriseTLD" : "minimumRPMS.sunriseTLD",
  "rdap.baseURLs" : [ {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  }, {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  } ],
  "minimumRPMS.claimsTLD" : "minimumRPMS.claimsTLD",
  "rdap.testEntities" : [ {
    "handle" : "handle",
    "tld" : "tld"
  }, {
    "handle" : "handle",
    "tld" : "tld"
  } ]
},
"missingInputs" : [ "missingInputs", "missingInputs" ],
"dueDate" : "2000-01-23T04:56:07.000+00:00",
"dateRequested" : "2000-01-23T04:56:07.000+00:00",
"errorCodes" : [ "errorCodes", "errorCodes" ],
"clientIDs" : [ "clientIDs", "clientIDs" ],
"rsp" : "rsp",
"dateUpdated" : "2000-01-23T04:56:07.000+00:00",
"tlds" : [ [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
} ], [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
} ] ],
"dateStarted" : "2000-01-23T04:56:07.000+00:00",
"dateCompleted" : "2000-01-23T04:56:07.000+00:00",
"testPlan" : "StandardPreDelegationTest",
"files" : [ {
  "name" : "name",
  "uploaded" : "2000-01-23T04:56:07.000+00:00",
  "href" : "http://example.com/aeiou",
  "type" : "type"
}, {
  "name" : "name",
```

```
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  } ],
  "testID" : "testID",
  "applicationID" : "applicationID",
  "results" : [ {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  }, {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
```

```
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  } ]
}, {
  "testPlanVersion" : "testPlanVersion",
  "inputs" : {
    "dnssecOps.zskRolloverZone" : "dnssecOps.zskRolloverZone",
    "dnssecOps.primaryServers" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
    },
    "epp.registeredNames" : [ "epp.registeredNames", "epp.registeredNames" ],
    "dnssecOps.kskRolloverZone" : "dnssecOps.kskRolloverZone",
    "epp.secDNSInterfaces" : "dsData",
    "minimumRPMS.sunriseModels" : "start-date",
    "epp.clid02DataModel" : "minimum",
    "rde.publicKey" : "rde.publicKey",
    "srsgw.eppHostName" : "srsgw.eppHostName",
    "general.registryDataModel" : "minimum",
    "integration.rriACL" : {
      "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
      "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
    },
    "epp.requiredContactTypes" : [ "admin", "admin" ],
    "srsgw.eppClid01" : "srsgw.eppClid01",
    "srsgw.eppClid02" : "srsgw.eppClid02",
    "rdap.testDomains" : [ "rdap.testDomains", "rdap.testDomains" ],
    "rde.signatureFile" : "rde.signatureFile",
    "srsgw.eppPwd01" : "srsgw.eppPwd01",
    "srsgw.eppPwd02" : "srsgw.eppPwd02",
    "epp.registeredContacts" : [ "epp.registeredContacts", "epp.registeredContacts" ],
    "epp.hostModel" : "objects",
    "integration.rdeSFTPUsername" : "integration.rdeSFTPUsername",
    "epp.greeting" : "epp.greeting",
    "srsgw.rdapBaseURLs" : [ {
      "baseURL" : "http://example.com/aeiou",
      "tld" : "tld"
    }, {
      "baseURL" : "http://example.com/aeiou",
      "tld" : "tld"
```

```
} ],
"dns.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.clid01DataModel" : "minimum",
"epp.serverIssuedClientCertificate02" : "epp.serverIssuedClientCertificate02",
"epp.serverIssuedClientCertificate01" : "epp.serverIssuedClientCertificate01",
"rde.depositFile" : "rde.depositFile",
"dnssecOps.tsigKey" : {
  "name" : "name",
  "secret" : "secret",
  "algorithm" : "hmac-sha256"
},
"rdap.testNameservers" : [ {
  "nameserver" : "nameserver",
  "tld" : "tld"
}, {
  "nameserver" : "nameserver",
  "tld" : "tld"
} ],
"dnssec.dsRecords" : [ {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
}, {
  "dsRecords" : [ {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  }, {
    "keyTag" : 5,
    "digestType" : 1,
    "digest" : "digest",
    "alg" : 6
  } ],
  "name" : "name"
} ],
"epp.hostName" : "epp.hostName",
"srsgw.serverIssuedClientCertificate01" : "srsgw.serverIssuedClientCertificate01",
"epp.registeredNameservers" : [ "epp.registeredNameservers", "epp.registeredNameservers" ],
"srsgw.serverIssuedClientCertificate02" : "srsgw.serverIssuedClientCertificate02",
"integration.rdeSFTPHostname" : "integration.rdeSFTPHostname",
"dnssecOps.nameservers" : [ {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
}, {
  "v6Addrs" : [ "v6Addrs", "v6Addrs" ],
  "name" : "name",
  "v4Addrs" : [ "v4Addrs", "v4Addrs" ]
} ],
"epp.restoreReportRequired" : true,
"dnssecOps.algorithmRolloverZone" : "dnssecOps.algorithmRolloverZone",
"epp.pwd02" : "epp.pwd02",
"epp.pwd01" : "epp.pwd01",
"integration.rdeSFTPDirectory" : "integration.rdeSFTPDirectory",
"dnssecOps.csk" : true,
```

```
  "epp.clid02" : "epp.clid02",
  "epp.clid01" : "epp.clid01",
  "minimumRPMS.sunriseTLD" : "minimumRPMS.sunriseTLD",
  "rdap.baseURLs" : [ {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  }, {
    "baseURL" : "http://example.com/aeiou",
    "tld" : "tld"
  } ],
  "minimumRPMS.claimsTLD" : "minimumRPMS.claimsTLD",
  "rdap.testEntities" : [ {
    "handle" : "handle",
    "tld" : "tld"
  }, {
    "handle" : "handle",
    "tld" : "tld"
  } ]
},
"missingInputs" : [ "missingInputs", "missingInputs" ],
"dueDate" : "2000-01-23T04:56:07.000+00:00",
"dateRequested" : "2000-01-23T04:56:07.000+00:00",
"errorCodes" : [ "errorCodes", "errorCodes" ],
"clientIDs" : [ "clientIDs", "clientIDs" ],
"rsp" : "rsp",
"dateUpdated" : "2000-01-23T04:56:07.000+00:00",
"tlds" : [ [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
} ], [ {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
}, {
  "name" : "name",
  "idnTables" : [ {
    "id" : "id",
    "variantSupportLevel" : 0
  }, {
    "id" : "id",
    "variantSupportLevel" : 0
  } ]
} ] ],
"dateStarted" : "2000-01-23T04:56:07.000+00:00",
"dateCompleted" : "2000-01-23T04:56:07.000+00:00",
"testPlan" : "StandardPreDelegationTest",
"files" : [ {
  "name" : "name",
  "uploaded" : "2000-01-23T04:56:07.000+00:00",
  "href" : "http://example.com/aeiou",
  "type" : "type"
}, {
  "name" : "name",
```

```
    "uploaded" : "2000-01-23T04:56:07.000+00:00",
    "href" : "http://example.com/aeiou",
    "type" : "type"
  } ],
  "testID" : "testID",
  "applicationID" : "applicationID",
  "results" : [ {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  }, {
    "dateStarted" : "dateStarted",
    "log" : [ {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
```

```
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    }, {
      "result" : "pass",
      "caseRef" : "http://example.com/aeiou",
      "dateStarted" : "2000-01-23T04:56:07.000+00:00",
      "log" : [ {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      }, {
        "severity" : "WARNING",
        "code" : "DNSSEC_DNS_QUERY_ERROR",
        "codeRef" : "http://example.com/aeiou",
        "message" : "message",
        "timestamp" : "2000-01-23T04:56:07.000+00:00"
      } ],
      "dateCompleted" : "2000-01-23T04:56:07.000+00:00",
      "caseID" : "dns-address01",
      "context" : {
        "key" : "context"
      },
      "description" : "description"
    } ],
    "dateCompleted" : "dateCompleted",
    "runID" : "runID"
  } ]
} ]
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
A successful result.

StartingTestRuns

Up

`POST /test/{id}/run`

(startTestRun)

This operation asks the test system to start a new test run. If test object's status property is `inputs-complete`, a `201` status will be returned; otherwise a `400` status will be returned.

Path parameters

id (required)
Path Parameter — the test ID default: null

Responses
201
A succesful result.

TestAdministration

POST /test/{id}/result

Up

(setTestResult)

Internal users only

This operation allows the `result` property of a test request to be overridden. The current value of this propert **MUST** be either `exception` or `fail`. If successful, the `result` property will be changed to `pass`.

Path parameters

id (required)
Path Parameter — the test ID default: null

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/x-www-form-urlencoded`

Form parameters

pass (required)
Form Parameter — default: null

Responses

201
A succesful result.

Models

[Jump to Methods]

Table of Contents

idnTable -

Up

This object describes an IDN table, or more specifically, an RSP's implementation of such a table.

tableID
String A unique ID for this table.

active
Boolean

indicates whether this table is available for use in testing. If this property is `false`, this table cannot be used in tests.

In OT&E, this **MUST** be omitted when creating a new test request.

rsp
String

The RSP's unique ID.

In OT&E, this **MUST** be omitted when creating a new test request, and will be populated using the FQDN of the first `TLSA` record which matches the certificate presented by the client.

tag
String the language tag, which must conform to the specification in RFC 5646.

version
String the version number.

isReferenceLGR
Boolean

whether the RSP's implementation of the IDN table uses a Second- Level Reference Label Generation Rules (LGRs) developed by ICANN.

In OT&E, this value **MUST** be `true`, as custom LGRs cannot be tested in OT&E.

For more information on ICANN's Second-Level LGRS, please see:

- https://www.icann.org/resources/pages/second-level-lgr-2015-06-21-en

variantPolicy
String

The variant policy supported for this table. The possible values are:

- `novar` - no variants are supported/applicable
- `allblockvar` - all variants are blocked
- `mayallocatevar` - variants may be allocated

Enum:
novar
allblockvar
mayallocatevar

testLabels (optional)
idnTestLabels

`idnTableRef` - Up

An `idnTableRef` object provides a reference to an IDN table object. IDN table objects must be created prior to being referenced in a test request.

id
String The unique ID of the table.

variantSupportLevel
Integer

The level of variant supported offered. Possible values are:

- `0` - no variants supported
- `1` - variants are blocked
- `2` - variants under the same TLD may be allocated
- `3` - variants under variant TLDs may be allocated

`idnTableRequest` - Up

This object describes an IDN table, or more specifically, an RSP's implementation of such a table.

In OT&E, the `isReferenceLGR` property **MUST** be `true`, and the `testLabels` property **MUST** be omitted.

active (optional)
Boolean

indicates whether this table is available for use in testing. If this property is `false`, this table cannot be used in tests.

In OT&E, this **MUST** be omitted when creating a new test request.

rsp (optional)
String

The RSP's unique ID.

In OT&E, this **MUST** be omitted when creating a new test request, and will be populated using the FQDN of the first `TLSA` record which matches the certificate presented by the client.

tag
String the language tag, which must conform to the specification in RFC 5646.

version
String the version number.

isReferenceLGR
Boolean

whether the RSP's implementation of the IDN table uses a Second- Level Reference Label Generation Rules (LGRs) developed by ICANN.

In OT&E, this value **MUST** be `true`, as custom LGRs cannot be tested in OT&E.

For more information on ICANN's Second-Level LGRS, please see:

- https://www.icann.org/resources/pages/second-level-lgr-2015-06-21-en

variantPolicy
String

The variant policy supported for this table. The possible values are:

- `novar` - no variants are supported/applicable
- `allblockvar` - all variants are blocked
- `mayallocatevar` - variants may be allocated

Enum:
novar
allblockvar
mayallocatevar

testLabels (optional)
idnTestLabels

`idnTestLabels` - Up

An object containing IDN test labels.

validLabels
array[idnTestLabels_validLabels_inner] A mapping of **valid** IDN labels to any variant labels thereof.

invalidLabels
array[String] A list of **invalid** IDN labels that it should not be possible to register.

`idnTestLabels_validLabels_inner` - Up

label
String The primary label.

variants
array[idnTestLabels_validLabels_inner_variants_inner] A list of variant labels.

`idnTestLabels_validLabels_inner_variants_inner` - Up

A variant label.

label
String The label.

tag
String The language tag that the label is valid in.

variantPolicy (optional)
idnTestLabels_validLabels_inner_variants_inner_variantPolicy

idnTestLabels_validLabels_inner_variants_inner_variantPolicy - Up

A variant label may have an allocation policy which overrides that of the IDN table it is associated with, and this policy may be different depending on whether the label is being considered for allocation in the same TLD as that of the primary label, or in a variant TLD. If this property is not defined for a variant label, the it inherits the allocation policy of the IDN table.

inSameTLD
String The allocation policy for the variant label in the same TLD as the primary label.
Enum:
allblockvar
mayallocatevar

inVariantTLD
String The allocation policy for the variant label in a variant TLD.
Enum:
allblockvar
mayallocatevar

inputParameters - Up

Users must provide various input parameters to be used within test cases. An `inputParameters` object is used when users submit these parameters after a test request object is created, and is also used when returning test request object information.

dnsPeriodnameservers (optional)
array[inputParameters_dns_nameservers_inner] The set of nameservers that will be authoritative for the zones used in the DNSSEC operations test suite.

dnssecPerioddsRecords (optional)
array[inputParameters_dnssec_dsRecords_inner]

The DS record(s) that may be used to validate the DNSSEC signature for the TLD(s). This input parameter is an object where the object properties are the TLD names and the values are arrays of objects representing DS records.

There **MUST** be an entry for every TLD in the TLD set and there **MUST** be at least one DS record for each TLD.

dnssecOpsPeriodalgorithmRolloverZone (optional)
String The domain name which will be monitored for the occurrence of an algorithm rollover. format: hostname

dnssecOpsPeriodcsk (optional)
Boolean A boolean indicating whether the RSP uses a Combined Signing Key (CSK, also referred to as a "Single Type Signing Scheme") instead of a split KSK/ZSK configuration.

dnssecOpsPeriodkskRolloverZone (optional)
String The domain name which will be monitored for the occurrence of a KSK rollover. format: hostname

dnssecOpsPeriodnameservers (optional)
array[inputParameters_dns_nameservers_inner] The set of nameservers that will be authoritative for the zones used in the DNSSEC operations test suite.

dnssecOpsPeriodprimaryServers (optional)
inputParameters_dnssecOps_primaryServers

dnssecOpsPeriodtsigKey (optional)
inputParameters_dnssecOps_tsigKey

dnssecOpsPeriodzskRolloverZone (optional)
String The domain name which will be monitored for the occurrence of a ZSK rollover. format: hostname

eppPeriodclid01 (optional)
String the username used to log in to the EPP server

eppPeriodclid01DataModel (optional)

String

the data model configured for this registrar. This may be omitted and will in any case be ignored unless the value of the `general.registryDataModel` input parameter is `per-registrar`.

- A value of `minimum` means that this registrar does not need to specify a registrant object when creating a domain name.
- A value of `maximum` means that this registrar **MUST** specify a registrant object when creating a domain name.

If the value of the `general.registryDataModel` input parameter is `per-registrar`, then the value of this input parameter **MUST** be different from the value of the `epp.clid02DataModel` input parameter.

Enum:
minimum
maximum

eppPeriodclid02 (optional)
String the username used for transfer tests

eppPeriodclid02DataModel (optional)
String

the data model configured for this registrar. This may be omitted and will in any case be ignored unless the value of the `general.registryDataModel` input parameter is `per-registrar`.

- A value of `minimum` means that this registrar does not need to specify a registrant object when creating a domain name.
- A value of `maximum` means that this registrar **MUST** specify a registrant object when creating a domain name.

If the value of the `general.registryDataModel` input parameter is `per-registrar`, then the value of this input parameter **MUST** be different from the value of the `epp.clid01DataModel` input parameter.

Enum:
minimum
maximum

eppPeriodgreeting (optional)
String an XML instance which contains a copy of the server's `<greeting>`.

eppPeriodhostModel (optional)
String

The host model supported by the EPP server. The possible values for this parameter are:

- `objects`
- `attributes`

Enum:
objects
attributes

eppPeriodhostName (optional)
String

The fully-qualified domain name of the EPP server.

The server name **MUST** comply with the requirements for valid hostnames described in RFC 1123, section 2.1. Additionally, all IDN labels in the server name **MUST** comply with IDNA2008.

format: hostname

eppPeriodpwd01 (optional)
String the password used to log in to the EPP server

eppPeriodpwd02 (optional)
String the password used for transfer tests

eppPeriodregisteredContacts (optional)
array[String]

An array of contact IDs that exist in the EPP server and which are therefore unavailable for registration.

If the value of `general.registryDataModel` is `maximum`, or `per-registrar`, then this array

MUST contain one member for each TLD in the TLD set. However, if it is `minimum`, the array **MUST** be empty.

eppPeriodregisteredNames (optional)
array[String] An array of domain names that exist in the EPP server and which are therefore unavailable for registration. The domains **MUST NOT** be under the sponsorship of the `epp.clid01` or `epp.clid02` registrars. The array **MUST** contain one member for each TLD in the TLD set. format: hostname

eppPeriodregisteredNameservers (optional)
array[String]

An array of host objects that exist in the EPP server and which are therefore unavailable for registration.

If the value of `epp.hostModel` is `objects`, this array **MUST** contain one member for each TLD in the TLD set. However, if it is `attributes`, the array **MUST** be empty.

format: hostname

eppPeriodrequiredContactTypes (optional)
array[String] An array containing the values of the `type` attribute of `<contact>` element(s) that are required to successfully create a domain name. If the value of `general.registryDataModel` is `minimum`, this array `MUST` be empty.
Enum:

eppPeriodrestoreReportRequired (optional)
Boolean Whether the server requires submission of a restore report when a client attempts to restore a domain.

eppPeriodsecDNSInterfaces (optional)
String Which of the interfaces defined in Section 4 of RFC 5910 the server supports (either `dsData` or `keyData`).
Enum:
dsData
keyData

eppPeriodserverIssuedClientCertificate01 (optional)
String If the EPP server uses a private CA to issue client certificates, then a certificate generated using the CSR provided in the `epp.clientCSR` resource may be provided using this parameter. This certificate will only be used in conjunction with the `epp.clid01` and `epp.pwd01` credentials. If the server will accept ICANN's own client certificate, this parameter **SHOULD** be empty.

eppPeriodserverIssuedClientCertificate02 (optional)
String If the EPP server uses a private CA to issue client certificates, then a certificate generated using the CSR provided in the `epp.clientCSR` resource may be provided using this parameter. This certificate will only be used in conjunction with the `epp.clid02` and `epp.pwd02` credentials. If the server will accept ICANN's own client certificate, this parameter **SHOULD** be empty.

generalPeriodregistryDataModel (optional)
String

This input parameter describes the data model(s) supported by the registry, determined in accordance with Section 7 of the Registration Data Policy. The possible values are:

- `minimum`: the registry does not collect registrant contact information from registrars. This policy applies to all registrars.
- `maximum`: the registry requires the transmission of registrant contact information from registrars for all registrations. This policy applies to all registrars.
- `per-registrar`: the registry may or may not require transmission of registrant contact information, depending on whether there is an appropriate legal basis, and a data processing agreement is in place between the registry operator and the registrar. Therefore, the data model is determined per-registrar rather than globally.

If the value of this parameter is `per-registrar`, then one of the registrar accounts specified by the `epp.clid01` and `epp.clid02` input parameters **MUST** be configured to use the minimum data model, and one **MUST** be configured to use the maximum data model. The `epp.clid01DataModel` and `epp.clid02DataModel` input parameters are used to identify the data model configured for each account.

Enum:
minimum
maximum
per-registrar

integrationPeriodrdeSFTPDirectory (optional)

String The directory on the SFTP server where deposit files may be found.

integrationPeriodrdeSFTPHostname (optional)
String The hostname of the operator's SFTP server. format: hostname

integrationPeriodrdeSFTPUsername (optional)
String The username that can be used to connect to the SFTP server.

integrationPeriodrriACL (optional)
inputParameters_integration_rriACL

minimumRPMSPeriodclaimsTLD (optional)
String A TLD, or other registry-class zone, which has been configured to be in perpetual trademark claims. format: hostname

minimumRPMSPeriodsunriseModels (optional)
String

The sunrise models supported by the EPP server. The possible values for this parameter are:

- `start-date`
- `end-date`
- `both`

Enum:
start-date
end-date
both

minimumRPMSPeriodsunriseTLD (optional)
String A TLD, or other registry-class zone, which has been configured to be in perpetual sunrise. format: hostname

rdapPeriodbaseURLs (optional)
array[inputParameters_rdap_baseURLs_inner]

The RDAP base URL(s) for the TLD(s).

The host name component of each URL **MUST** comply with the requirements for valid hostnames described in RFC 1123, section 2.1. Additionally, all IDN labels in the host name **MUST** comply with IDNA2008.

rdapPeriodtestDomains (optional)
array[String] The domain(s) that will be queried to validate domain responses. This input parameter is an array of domain names, which **MUST** include at least one domain name for each TLD being tested. format: hostname

rdapPeriodtestEntities (optional)
array[inputParameters_rdap_testEntities_inner] The entities(s) that will be queried to validate entity responses. This input parameter is an array of objects. At least one entity **MUST** be provided for each TLD being tested.

rdapPeriodtestNameservers (optional)
array[inputParameters_rdap_testNameservers_inner] The nameservers(s) that will be queried to validate nameserver responses. This input parameter is an array of objects. At least one nameserver **MUST** be provided for each TLD being tested.

rdePerioddepositFile (optional)
String an RDE deposit file. The TLD to which the deposit relates **MUST** match one of the TLDs that are associated with the test object.

rdePeriodpublicKey (optional)
String a PGP public key block

rdePeriodsignatureFile (optional)
String an ASCII-armoured OpenPGP signature covering the deposit file

srsgwPeriodeppClid01 (optional)
String the username used to log in to the SRS Gateway EPP server

srsgwPeriodeppClid02 (optional)
String the username used for transfer tests

srsgwPeriodeppHostName (optional)
String the fully-qualified domain name of the SRS Gateway EPP server format: hostname

srsgwPeriodeppPwd01 (optional)
String the password used to log in to the SRS Gateway EPP server

srsgwPeriodeppPwd02 (optional)

String the password used for transfer tests

srsgwPeriodrdapBaseURLs (optional)
array[inputParameters_rdap_baseURLs_inner]

The RDAP base URL(s) for the TLD(s).

The host name component of each URL **MUST** comply with the requirements for valid hostnames described in RFC 1123, section 2.1. Additionally, all IDN labels in the host name **MUST** comply with IDNA2008.

srsgwPeriodserverIssuedClientCertificate01 (optional)
String If the EPP server uses a private CA to issue client certificates, then a certificate generated using the CSR provided in the `epp.clientCSR` may be provided using this parameter. This certificate will only be used in conjunction with the `srsgw.eppClid01` and `srsgw.eppPwd01` credentials. If the server will accept ICANN's own client certificate, this parameter **SHOULD** be empty.

srsgwPeriodserverIssuedClientCertificate02 (optional)
String If the EPP server uses a private CA to issue client certificates, then a certificate generated using the CSR provided in the `epp.clientCSR` may be provided using this parameter. This certificate will only be used in conjunction with the `srsgw.eppClid02` and `srsgw.eppPwd02` credentials. If the server will accept ICANN's own client certificate, this parameter **SHOULD** be empty.

inputParameters_dns_nameservers_inner - Up

name
String The fully-qualified nameserver name. format: hostname

v4Addrs (optional)
array[String] The IPv4 address(es) for the nameserver. format: ipv4

v6Addrs (optional)
array[String] The IPv6 address(es) for the nameserver. format: ipv6

inputParameters_dnssecOps_primaryServers - Up

The primary nameserver(s) from which zones can be transferred.

v4Addrs
array[String] The IPv4 address(es) for the primary server(s). format: ipv4

v6Addrs
array[String] The IPv6 address(es) for the primary server(s). format: ipv6

inputParameters_dnssecOps_tsigKey - Up

The TSIG key which should be used to perform zone transfers.

algorithm (optional)
String The TSIG algorithm. The mnemonics are a subset of those published in the IANA registry at https://www.iana.org/assignments/tsig-algorithm-names/tsig-algorithm-names.xhtml.
Enum:
hmac-sha256
hmac-sha384
hmac-sha512

name (optional)
String The TSIG name. format: hostname

secret (optional)
String The TSIG secret.

inputParameters_dnssec_dsRecords_inner - Up

dsRecords
array[inputParameters_dnssec_dsRecords_inner_dsRecords_inner] the DS record(s)

name
String the zone name format: hostname

inputParameters_dnssec_dsRecords_inner_dsRecords_inner - Up

alg
Integer format: uint16

digest
String

digestType
Integer format: uint16

keyTag
Integer format: uint16

inputParameters_integration_rriACL - Up

An object representing the IP addresses from which requests to the RRI will be sent.

v4Addrs
array[String] format: ipv4

v6Addrs
array[String] format: ipv6

inputParameters_rdap_baseURLs_inner - Up

baseURL
String The RDAP Base URL. The URL **MUST** have trailing slash (`/`). format: url

tld
String The TLD or equivalent registry-class domain name. format: hostname

inputParameters_rdap_testEntities_inner - Up

handle
String the entity handle.

tld
String The TLD. format: hostname

inputParameters_rdap_testNameservers_inner - Up

nameserver
String The nameserver name. format: hostname

tld
String The TLD. format: hostname

testCaseLog - Up

A detailed log of an individual test case.

caseID
String the Test Case ID.
Enum:
dns-address01
dns-address02
dns-address03
dns-connectivity01
dns-connectivity02
dns-connectivity03
dns-consistency02
dns-consistency03
dns-consistency04
dns-consistency05
dns-consistency06
dns-delegation01
dns-delegation02
dns-delegation03
dns-delegation04
dns-delegation05
dns-delegation07

dns-nameserver01
dns-nameserver02
dns-nameserver04
dns-nameserver05
dns-nameserver06
dns-nameserver08
dns-nameserver09
dns-nameserver10
dns-nameserver11
dns-nameserver12
dns-nameserver13
dns-syntax05
dns-syntax06
dns-syntax07
dns-zone07
dns-zone10
dns-zz-idna2008-compliance
dnssec-01
dnssec-02
dnssec-03
dnssec-04
dnssec-05
dnssec-06
dnssec-08
dnssec-09
dnssec-10
dnssec-13
dnssec-14
dnssec-91
dnssec-92
dnssec-93
dnssecOps01-ZSKRollover
dnssecOps02-KSKRollover
dnssecOps03-AlgorithmRollover
epp-01
epp-02
epp-03
epp-04
epp-05
epp-06
epp-07
epp-08
epp-09
epp-10
epp-11
epp-12
epp-13
epp-14
epp-15
epp-16
epp-17
epp-18
epp-19
epp-20
epp-21
epp-22
epp-23
epp-24
idn-01
idn-02
idn-03
idn-04
integration-01
integration-02
integration-03
minimumRPMs-01
minimumRPMs-02
minimumRPMs-03
rdap-01-ipv4Validation
rdap-02-ipv6Validation
rdap-03-domainNameValidation
rdap-04-webUriValidation
rdap-05-domainCaseFoldingValidation

rdap-06-stdRdapConformanceValidation
rdap-07-stdRdapLinksValidation
rdap-08-stdRdapNoticesRemarksValidation
rdap-09-stdRdapLanguageIdentifierValidation
rdap-10-stdRdapEventsValidation
rdap-11-stdRdapStatusValidation
rdap-12-stdRdapPort43WhoisServerValidation
rdap-13-stdRdapPublicIdsValidation
rdap-14-stdRdapAsEventActorValidation
rdap-15-stdRdapIpAddressesValidation
rdap-16-stdRdapVariantsValidation
rdap-17-stdRdapUnicodeNameValidation
rdap-18-stdRdapLdhNameValidation
rdap-19-stdRdapRolesValidation
rdap-20-stdRdapEntitiesValidation
rdap-21-stdRdapSecureDnsValidation
rdap-22-stdRdapErrorResponseBodyValidation
rdap-23-stdRdapDomainLookupValidation
rdap-24-stdRdapEntityLookupValidation
rdap-25-stdRdapNameserverLookupValidation
rdap-26-stdRdapHelpValidation
rdap-27-stdRdapNameserversSearchValidation
rdap-91
rdap-92
rde-01
rde-02
rde-03
rde-04
rde-05
rde-06
rde-07
rde-08
rde-09
rde-10
rde-11
rde-12
rde-13
rde-14
srsgw-01
srsgw-02
srsgw-03
srsgw-04
srsgw-05
srsgw-06
srsgw-07
srsgw-08
srsgw-09
srsgw-10
srsgw-11
srsgw-12
srsgw-13
srsgw-14
srsgw-15

caseRef
String a link to the test case specification format: url

result
String

The result of the test. The possible values are:

- `pass` - the test passed.
- `fail` - the test was not passed.
- `exception` - an error occurred which meant a result could not be determined. This indicates an issue on the RST test system side, not the test subject's.
- `skipped` - the test case was not applicable and was not carried out.
- `aborted` - the test case was aborted before it could complete.

Enum:
pass
fail
exception
skipped

aborted

description
String A short description of the outcome of the test.

dateStarted
Date date/time when the test case started. format: date-time

dateCompleted
Date date/time when the test case finished. format: date-time

log
array[testCaseLogMessage] detailed test logs

context
map[String, String] an object containing context parameters (eg input parameters, software versions, environment variables, etc). These are intended to assist in debugging any issues that may have caused the test case to fail or error.

`testCaseLogMessage` - Up

A log message.

code (optional)
String the error code (if any).
Enum:
DNSSEC_DNS_QUERY_ERROR
DNSSEC_INVALID_DIGEST_ALGORITHM
DNSSEC_INVALID_SIGNING_ALGORITHM
DNSSEC_NSEC3_ITERATIONS_IS_NOT_ZERO
DNSSEC_NSEC3_SALT_IS_NOT_EMPTY
DNSSEC_OPS_ALGORITHM_ROLLOVER_CHAIN_OF_TRUST_BROKEN
DNSSEC_OPS_ALGORITHM_ROLLOVER_NOT_COMPLETED
DNSSEC_OPS_DNS_QUERY_FAILED_TOO_MANY_TIMES
DNSSEC_OPS_INVALID_ALGORITHM
DNSSEC_OPS_KSK_ROLLOVER_CHAIN_OF_TRUST_BROKEN
DNSSEC_OPS_KSK_ROLLOVER_NOT_COMPLETED
DNSSEC_OPS_XFR_FAILED_TOO_MANY_TIMES
DNSSEC_OPS_ZONE_IS_INVALID
DNSSEC_OPS_ZSK_ROLLOVER_CHAIN_OF_TRUST_BROKEN
DNSSEC_OPS_ZSK_ROLLOVER_NOT_COMPLETED
DNS_IDNA2008_INVALID_MNAME
DNS_IDNA2008_INVALID_NS_NSDNAME
DNS_IDNA2008_INVALID_RNAME
DNS_IDNA2008_QUERY_FAILED
DNS_INCONSISTENT_RESPONSES
EPP_CONTACT_CHECK_INVALID_CONTACT_ID_INCORRECT_AVAIL
EPP_CONTACT_CHECK_REGISTERED_CONTACT_ID_INCORRECT_AVAIL
EPP_CONTACT_CHECK_VALID_CONTACT_ID_INCORRECT_AVAIL
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_CC
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_CITY
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_EMAIL
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_ID
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_NAME
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_ORG
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_PC
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_POSTALINFO_TYPE
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_SP
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_STREET
EPP_CONTACT_CREATE_INFO_RESPONSE_MISSING_OR_INCORRECT_VOICE
EPP_CONTACT_CREATE_INFO_RESPONSE_NOT_1000
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_CC
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_CITY
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_EMAIL
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_NAME
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_STREET
EPP_CONTACT_CREATE_SERVER_ACCEPTS_EMPTY_VOICE
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_CC
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_CITY
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_EMAIL
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_ID
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_NAME
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_ORG
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_POSTALINFO_TYPE
EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_STREET

EPP_CONTACT_CREATE_SERVER_ACCEPTS_INVALID_VOICE
EPP_CONTACT_DELETE_OBJECT_STILL_EXISTS
EPP_CONTACT_DELETE_RESPONSE_NOT_1000_OR_1001
EPP_CONTACT_INFO_RESPONSE_NOT_2201
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_CC
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_CITY
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_EMAIL
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_ID
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_NAME
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_ORG
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_PC
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_POSTALINFO_TYPE
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_SP
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_STATUS
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_STREET
EPP_CONTACT_UPDATE_INFO_RESPONSE_MISSING_OR_INCORRECT_VOICE
EPP_CONTACT_UPDATE_INFO_RESPONSE_NOT_1000
EPP_CONTACT_UPDATE_RESPONSE_NOT_2201
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_CC
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_CITY
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_EMAIL
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_ORG
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_PC
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_POSTALINFO_TYPE
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_SP
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_STATUS
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_STREET
EPP_CONTACT_UPDATE_SERVER_ACCEPTS_INVALID_VOICE
EPP_DNS_RESOLUTION_ERROR
EPP_DOMAIN_CHECK_INVALID_DOMAIN_INCORRECT_AVAIL
EPP_DOMAIN_CHECK_REGISTERED_DOMAIN_INCORRECT_AVAIL
EPP_DOMAIN_CHECK_VALID_DOMAIN_INCORRECT_AVAIL
EPP_DOMAIN_CREATE_INFO_RESPONSE_INVALID_ROID
EPP_DOMAIN_CREATE_INFO_RESPONSE_MISSING_OBJECT_PROPERTIES
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_AUTHINFO
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_HOST_ATTRIBUTES_WITHOUT_GLUE
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_INVALID_DNSSEC_DATA
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_INVALID_DOMAIN_NAME
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_INVALID_HOST_OBJECT
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_INVALID_PERIOD
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_NON_EXISTENT_CONTACT_OBJECT
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_NON_EXISTENT_HOST_OBJECT
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_NO_REGISTRANT_FOR_THICK_REGISTRY
EPP_DOMAIN_CREATE_SERVER_ACCEPTS_REGISTRANT_FOR_THIN_REGISTRY
EPP_DOMAIN_CREATE_SERVER_INCORRECTLY_ACCEPTS_HOST_ATTRIBUTES
EPP_DOMAIN_CREATE_SERVER_INCORRECTLY_ACCEPTS_HOST_OBJECTS
EPP_DOMAIN_DELETE_INFO_RESPONSE_OBJECT_NOT_PENDING_DELETE
EPP_DOMAIN_DELETE_INFO_RESPONSE_OBJECT_STILL_EXISTS
EPP_DOMAIN_DELETE_INFO_RESPONSE_RGP_STATUS_NOT_PENDING_DELETE
EPP_DOMAIN_UPDATE_INFO_RESPONSE_MISSING_DNSSEC_DATA
EPP_DOMAIN_UPDATE_INFO_RESPONSE_MISSING_HOST_ATTRIBUTE
EPP_DOMAIN_UPDATE_INFO_RESPONSE_MISSING_HOST_OBJECT
EPP_DOMAIN_UPDATE_INFO_RESPONSE_MISSING_REGISTRANT
EPP_DOMAIN_UPDATE_INFO_RESPONSE_MISSING_STATUS_CODE
EPP_DOMAIN_UPDATE_INFO_RESPONSE_UNEXPECTED_DNSSEC_DATA
EPP_DOMAIN_UPDATE_INFO_RESPONSE_UNEXPECTED_HOST_ATTRIBUTE
EPP_DOMAIN_UPDATE_INFO_RESPONSE_UNEXPECTED_HOST_OBJECT
EPP_DOMAIN_UPDATE_INFO_RESPONSE_UNEXPECTED_REGISTRANT
EPP_DOMAIN_UPDATE_INFO_RESPONSE_UNEXPECTED_STATUS_CODE
EPP_DOMAIN_UPDATE_SERVER_ACCEPTS_HOST_ATTRIBUTES_WITHOUT_GLUE
EPP_DOMAIN_UPDATE_SERVER_ACCEPTS_INVALID_DNSSEC_DATA
EPP_DOMAIN_UPDATE_SERVER_ACCEPTS_INVALID_HOST_ATTRIBUTES
EPP_DOMAIN_UPDATE_SERVER_ACCEPTS_NON_EXISTENT_CONTACT_OBJECT
EPP_DOMAIN_UPDATE_SERVER_ACCEPTS_NON_EXISTENT_HOST_OBJECT
EPP_DOMAIN_UPDATE_SERVER_INCORRECTLY_ACCEPTS_HOST_ATTRIBUTES
EPP_DOMAIN_UPDATE_SERVER_INCORRECTLY_ACCEPTS_HOST_OBJECTS
EPP_GENERIC_COMMAND_ERROR
EPP_GREETING_DOES_NOT_MATCH
EPP_GREETING_INVALID_LANG
EPP_GREETING_MISSING_EN_LANG
EPP_GREETING_MISSING_EXTURI
EPP_GREETING_MISSING_OBJURI
EPP_GREETING_RECOMMENDED_EXTENSION_MISSING

EPP_GREETING_SVDATE_INVALID
EPP_GREETING_SVID_INVALID
EPP_GREETING_UNEXPECTED_EXTURI
EPP_GREETING_UNEXPECTED_OBJURI
EPP_GREETING_VERSION_INVALID
EPP_HOST_CHECK_INVALID_HOST_INCORRECT_AVAIL
EPP_HOST_CHECK_REGISTERED_HOST_INCORRECT_AVAIL
EPP_HOST_CHECK_VALID_HOST_INCORRECT_AVAIL
EPP_HOST_CREATE_INFO_RESPONSE_MISSING_OBJECT_PROPERTIES
EPP_HOST_CREATE_INFO_RESPONSE_OBJECT_DOES_NOT_EXIST
EPP_HOST_CREATE_SERVER_ACCEPTS_INVALID_HOSTNAME
EPP_HOST_CREATE_SERVER_ACCEPTS_INVALID_IPV4_ADDRESS
EPP_HOST_CREATE_SERVER_ACCEPTS_INVALID_IPv6_ADDRESS
EPP_HOST_DELETE_INFO_RESPONSE_OBJECT_STILL_EXISTS
EPP_HOST_RENAME_SERVER_ACCEPTS_INVALID_HOSTNAME
EPP_HOST_RENAME_SERVER_ACCEPTS_RENAME_TO_ANOTHER_REGISTRARS_DOMAIN
EPP_HOST_RENAME_SERVER_ACCEPTS_RENAME_TO_NONEXISTENT_DOMAIN
EPP_HOST_RENAME_SERVER_REJECTS_OUT_OF_BAILIWICK_NAME
EPP_HOST_RENAME_SERVER_UNEXPECTEDLY_REJECTS_RENAME
EPP_HOST_UPDATE_AUTHZ_ERROR
EPP_HOST_UPDATE_INFO_RESPONSE_MISSING_OBJECT_PROPERTIES
EPP_HOST_UPDATE_INFO_RESPONSE_OBJECT_DOES_NOT_EXIST
EPP_HOST_UPDATE_SERVER_ACCEPTS_INVALID_IPV4_ADDRESS
EPP_HOST_UPDATE_SERVER_ACCEPTS_INVALID_IPv6_ADDRESS
EPP_HOST_UPDATE_SERVER_ACCEPTS_INVALID_STATUS_CODE
EPP_INTEGRITY_SERVER_ACCEPTS_DELETE_FOR_LINKED_CONTACT_OBJECT
EPP_INTEGRITY_SERVER_ACCEPTS_DELETE_FOR_LINKED_HOST_OBJECT
EPP_LOGIN_ERROR
EPP_LOGIN_UNEXPECTEDLY_FAILED
EPP_LOGIN_UNEXPECTEDLY_SUCCEEDED
EPP_MISSING_AAAA_RECORDS
EPP_MISSING_A_RECORDS
EPP_NO_GREETING_RECEIVED
EPP_NO_SERVICE_PORTS_REACHABLE
EPP_RENEW_INFO_RESPONSE_MISSING_OR_INVALID_RGP_STATUS
EPP_RENEW_INFO_RESPONSE_UNEXPECTED_EXPIRY_DATE
EPP_RENEW_SERVER_ACCEPTS_INVALID_CURRENT_EXPIRY_DATE
EPP_RENEW_SERVER_ACCEPTS_INVALID_PERIOD
EPP_RESTORE_DOMAIN_STILL_PENDINGDELETE
EPP_SCHEMA_VALIDATION_ERROR
EPP_SERVICE_PORT_NOT_CONSISTENT
EPP_SERVICE_PORT_UNREACHABLE
EPP_TLS_BAD_CIPHER
EPP_TLS_CERTIFICATE_CHAIN_MISSING
EPP_TLS_CERTIFICATE_HOSTNAME_MISMATCH
EPP_TLS_CONNECTION_ERROR
EPP_TLS_EXPIRED_CERTIFICATE
EPP_TLS_FORBIDDEN_PROTOCOL_SUPPORTED
EPP_TLS_REQUIRED_PROTOCOL_NOT_SUPPORTED
EPP_TLS_UNTRUSTED_CERTIFICATE
EPP_TRANSFER_GAINING_REGISTRAR_NO_MESSAGE_RECEIVED
EPP_TRANSFER_INFO_RESPONSE_AUTHINFO_NOT_RESET
EPP_TRANSFER_INFO_RESPONSE_MISSING_OR_INVALID_RGP_STATUS
EPP_TRANSFER_INFO_RESPONSE_MISSING_OR_INVALID_STATUS_CODE
EPP_TRANSFER_INFO_RESPONSE_UNEXPECTED_EXPIRY_DATE
EPP_TRANSFER_LOSING_REGISTRAR_NO_MESSAGE_RECEIVED
EPP_TRANSFER_SERVER_ACCEPTS_INCORRECT_AUTHINFO
EPP_TRANSFER_SERVER_ACCEPTS_INSECURE_AUTHINFO
EPP_TRANSFER_SERVER_ACCEPTS_INVALID_PERIOD
EPP_TRANSFER_SERVER_PROCESSED_REJECTED_TRANSFER
EPP_UNEXPECTED_COMMAND_FAILURE
EPP_XML_PARSE_ERROR
IDN_SERVER_ACCEPTS_INVALID_LABEL
IDN_SERVER_REJECTS_VALID_LABEL
IDN_VARIANT_LABEL_NOT_BLOCKED
IDN_VARIANT_SERVER_ACCEPTS_VARIANT_CREATE_FROM_INCORRECT_REGISTRAR
IDN_VARIANT_SERVER_ACCEPTS_VARIANT_CREATE_WITH_INCORRECT_REGISTRANT
IDN_VARIANT_SERVER_REJECTS_VARIANT_CREATE_FROM_SAME_REGISTRAR
IDN_VARIANT_SERVER_REJECTS_VARIANT_CREATE_WITH_SAME_REGISTRANT
INTEGRATION_DNS_QUERY_FAILED
INTEGRATION_DOMAIN_NOT_PRESENT_IN_DNS
INTEGRATION_DOMAIN_NOT_PRESENT_IN_RDAP
INTEGRATION_DOMAIN_NOT_PRESENT_IN_RDE

INTEGRATION_RDAP_REQUEST_FAILED
INTEGRATION_RDE_SFTP_SERVER_AUTHENTICATION_ERROR
INTEGRATION_RDE_SFTP_SERVER_UNREACHABLE
RDAP_01_IPV4VALIDATION_FAILED
RDAP_02_IPV6VALIDATION_FAILED
RDAP_03_DOMAINNAMEVALIDATION_FAILED
RDAP_04_WEBURIVALIDATION_FAILED
RDAP_05_DOMAINCASEFOLDINGVALIDATION_FAILED
RDAP_06_STDRDAPCONFORMANCEVALIDATION_FAILED
RDAP_07_STDRDAPLINKSVALIDATION_FAILED
RDAP_08_STDRDAPNOTICESREMARKSVALIDATION_FAILED
RDAP_09_STDRDAPLANGUAGEIDENTIFIERVALIDATION_FAILED
RDAP_10_STDRDAPEVENTSVALIDATION_FAILED
RDAP_11_STDRDAPSTATUSVALIDATION_FAILED
RDAP_12_STDRDAPPORT43WHOISSERVERVALIDATION_FAILED
RDAP_13_STDRDAPPUBLICIDSVALIDATION_FAILED
RDAP_14_STDRDAPASEVENTACTORVALIDATION_FAILED
RDAP_15_STDRDAPIPADDRESSESVALIDATION_FAILED
RDAP_16_STDRDAPVARIANTSVALIDATION_FAILED
RDAP_17_STDRDAPUNICODENAMEVALIDATION_FAILED
RDAP_18_STDRDAPLDHNAMEVALIDATION_FAILED
RDAP_19_STDRDAPROLESVALIDATION_FAILED
RDAP_20_STDRDAPENTITIESVALIDATION_FAILED
RDAP_21_STDRDAPSECUREDNSVALIDATION_FAILED
RDAP_22_STDRDAPERRORRESPONSEBODYVALIDATION_FAILED
RDAP_23_STDRDAPDOMAINLOOKUPVALIDATION_FAILED
RDAP_24_STDRDAPENTITYLOOKUPVALIDATION_FAILED
RDAP_25_STDRDAPNAMESERVERLOOKUPVALIDATION_FAILED
RDAP_26_STDRDAPHELPVALIDATION_FAILED
RDAP_27_STDRDAPNAMESERVERSSEARCHVALIDATION_FAILED
RDAP_SERVICE_PORT_NOT_CONSISTENT
RDAP_TLS_BAD_CIPHER
RDAP_TLS_CERTIFICATE_CHAIN_MISSING
RDAP_TLS_CERTIFICATE_HOSTNAME_MISMATCH
RDAP_TLS_DNS_RESOLUTION_ERROR
RDAP_TLS_EXPIRED_CERTIFICATE
RDAP_TLS_FORBIDDEN_PROTOCOL_SUPPORTED
RDAP_TLS_NO_SERVICE_PORTS_REACHABLE
RDAP_TLS_REQUIRED_PROTOCOL_NOT_SUPPORTED
RDAP_TLS_SERVICE_PORT_UNREACHABLE
RDAP_TLS_UNTRUSTED_CERTIFICATE
RDE_CONTACT_HAS_INVALID_CC
RDE_CONTACT_HAS_INVALID_EMAIL
RDE_CONTACT_HAS_INVALID_ROID
RDE_CONTACT_HAS_MULTIPLE_POSTALINFO_TYPES
RDE_CONTACT_HAS_NON_UNIQUE_ID
RDE_CONTACT_HAS_UNKNOWN_ACRR
RDE_CONTACT_HAS_UNKNOWN_CLID
RDE_CONTACT_HAS_UNKNOWN_CRRR
RDE_CONTACT_HAS_UNKNOWN_RERR
RDE_CONTACT_HAS_UNKNOWN_UPRR
RDE_DECRYPTION_FAILED
RDE_DOMAIN_HAS_INVALID_CLID
RDE_DOMAIN_HAS_INVALID_CRDATE
RDE_DOMAIN_HAS_INVALID_EXDATE
RDE_DOMAIN_HAS_INVALID_NAME
RDE_DOMAIN_HAS_INVALID_REGISTRANT
RDE_DOMAIN_HAS_INVALID_ROID
RDE_DOMAIN_HAS_INVALID_STATUS
RDE_DOMAIN_HAS_MISSING_CLID
RDE_DOMAIN_HAS_MISSING_CRDATE
RDE_DOMAIN_HAS_MISSING_EXDATE
RDE_DOMAIN_HAS_MISSING_REGISTRANT
RDE_DOMAIN_HAS_MISSING_ROID
RDE_DOMAIN_HAS_MISSING_STATUS
RDE_GREETING_DOES_NOT_MATCH
RDE_HOST_HAS_INVALID_CLID
RDE_HOST_HAS_INVALID_IP_ADDRESS
RDE_HOST_HAS_INVALID_NAME
RDE_HOST_HAS_INVALID_ROID
RDE_HOST_HAS_INVALID_STATUS
RDE_HOST_HAS_MISSING_CLID
RDE_HOST_HAS_MISSING_IP_ADDRESS

RDE_HOST_HAS_MISSING_ROID
RDE_HOST_HAS_MISSING_STATUS
RDE_IDN_OBJECT_INVALID
RDE_IDN_OBJECT_MISSING
RDE_IDN_OBJECT_UNEXPECTED
RDE_INVALID_CSV
RDE_INVALID_FILENAME
RDE_INVALID_SIGNATURE
RDE_MISSING_OBJECT_URI
RDE_MIXED_TYPES
RDE_NNDN_CONFLICTS_WITH_DOMAIN
RDE_OBJECT_COUNT_MISMATCH
RDE_POLICY_OBJECT_INVALID
RDE_POLICY_OBJECT_MISSING
RDE_POLICY_OBJECT_MISSING_OBJECTS
RDE_POLICY_OBJECT_UNEXPECTED_OBJECTS
RDE_REGISTRAR_HAS_INVALID_GURID
RDE_REGISTRAR_HAS_INVALID_ID
RDE_REGISTRAR_HAS_INVALID_NAME
RDE_REGISTRAR_HAS_MISSING_GURID
RDE_REGISTRAR_HAS_MISSING_ID
RDE_REGISTRAR_HAS_MISSING_NAME
RDE_SCHEMA_VALIDATION_ERROR
RDE_UNEXPECTED_OBJECT_URI
RDE_XML_PARSE_ERROR
RPMS_MISSING_CLAIMS_KEY
RPMS_SUNRISE_CREATE_INFO_OBJECT_DOES_NOT_EXIST
RPMS_SUNRISE_CREATE_INFO_OBJECT_IS_HAS_MISSING_OR_INVALID_PROPERTIES
RPMS_SUNRISE_CREATE_UNEXPECTED_FAILURE_USING_VALID_SMD
RPMS_SUNRISE_CREATE_UNEXPECTED_SUCCESS_USING_EXPIRED_SMD
RPMS_SUNRISE_CREATE_UNEXPECTED_SUCCESS_USING_INCORRECT_SMD
RPMS_SUNRISE_CREATE_UNEXPECTED_SUCCESS_USING_REVOKED_SMD
RPMS_SUNRISE_CREATE_UNEXPECTED_SUCCESS_USING_SMD_WITH_REVOKED_SIGNATURE
RPMS_TRADEMARK_CREATE_INFO_OBJECT_DOES_NOT_EXIST
RPMS_TRADEMARK_CREATE_INFO_OBJECT_IS_HAS_MISSING_OR_INVALID_PROPERTIES
RPMS_TRADEMARK_CREATE_UNEXPECTED_FAILURE_USING_VALID_CLAIM_KEY
RPMS_TRADEMARK_CREATE_UNEXPECTED_SUCCESS_USING_INVALID_ACCEPTANCE_DATE
RPMS_TRADEMARK_CREATE_UNEXPECTED_SUCCESS_USING_INVALID_CLAIM_KEY
RPMS_UNEXPECTED_CLAIMS_KEY
SRSGW_CONTACT_CREATE_FAILED
SRSGW_CONTACT_CREATE_OBJECT_HAS_MISSING_OR_INVALID_PROPERTIES
SRSGW_CONTACT_CREATE_OBJECT_NOT_FOUND_WITHIN_DEADLINE
SRSGW_DOMAIN_CREATE_FAILED
SRSGW_DOMAIN_CREATE_OBJECT_HAS_MISSING_OR_INVALID_PROPERTIES
SRSGW_DOMAIN_CREATE_OBJECT_NOT_FOUND_WITHIN_DEADLINE
SRSGW_DOMAIN_DELETE_DOMAIN_NOT_PENDINGDELETE
SRSGW_DOMAIN_DELETE_FAILED
SRSGW_DOMAIN_DELETE_RGP_STATUS_NOT_PENDINGDELETERESTORABLE
SRSGW_DOMAIN_RENEW_FAILED
SRSGW_DOMAIN_RENEW_OBJECT_NOT_UPDATED_WITHIN_DEADLINE
SRSGW_DOMAIN_TRANSFER_APPROVAL_FAILED
SRSGW_DOMAIN_TRANSFER_APPROVAL_OBJECT_HAS_INCORRECT_CLID
SRSGW_DOMAIN_TRANSFER_APPROVAL_OBJECT_NOT_UPDATED_WITHIN_DEADLINE
SRSGW_DOMAIN_TRANSFER_APPROVAL_OBJECT_STILL_PENDING_TRANSFER
SRSGW_DOMAIN_TRANSFER_REQUEST_FAILED
SRSGW_DOMAIN_TRANSFER_REQUEST_OBJECT_NOT_PENDING_TRANSFER
SRSGW_DOMAIN_TRANSFER_REQUEST_OBJECT_NOT_UPDATED_WITHIN_DEADLINE
SRSGW_EPP_CONTACT_DELETE_FAILED
SRSGW_EPP_CONTACT_DELETE_OBJECT_STILL_EXISTS
SRSGW_EPP_CONTACT_UPDATE_FAILED
SRSGW_EPP_CONTACT_UPDATE_INFO_RESPONSES_DIFFER
SRSGW_EPP_HOST_DELETE_FAILED
SRSGW_EPP_HOST_DELETE_OBJECT_STILL_EXISTS
SRSGW_EPP_HOST_UPDATE_FAILED
SRSGW_EPP_HOST_UPDATE_INFO_RESPONSES_DIFFER
SRSGW_HOST_CREATE_FAILED
SRSGW_HOST_CREATE_OBJECT_HAS_MISSING_OR_INVALID_PROPERTIES
SRSGW_HOST_CREATE_OBJECT_NOT_FOUND_WITHIN_DEADLINE
SRSGW_RDAP_DNS_RESOLUTION_ERROR
SRSGW_RDAP_QUERY_FAILED
SRSGW_RDAP_RESPONSES_DIFFER
SRSGW_RDAP_SERVICE_PORT_UNREACHABLE
SRSGW_RDAP_TLS_CONNECTION_ERROR

ZM_AAAA_BAD_RDATA
ZM_AAAA_QUERY_DROPPED
ZM_AAAA_UNEXPECTED_RCODE
ZM_ALGORITHM_DEPRECATED
ZM_ALGORITHM_NOT_RECOMMENDED
ZM_ALGORITHM_NOT_ZONE_SIGN
ZM_ALGORITHM_PRIVATE
ZM_ALGORITHM_RESERVED
ZM_ALGORITHM_UNASSIGNED
ZM_A_UNEXPECTED_RCODE
ZM_BREAKS_ON_EDNS
ZM_CAN_NOT_BE_RESOLVED
ZM_CASE_QUERIES_RESULTS_DIFFER
ZM_CASE_QUERY_DIFFERENT_ANSWER
ZM_CASE_QUERY_DIFFERENT_RC
ZM_CASE_QUERY_NO_ANSWER
ZM_CHILD_NS_FAILED
ZM_CHILD_NS_SAME_IP
ZM_CHILD_ZONE_LAME
ZM_CN01_MISSING_NS_RECORD_UDP
ZM_CN01_MISSING_SOA_RECORD_UDP
ZM_CN01_NO_RESPONSE_NS_QUERY_UDP
ZM_CN01_NO_RESPONSE_SOA_QUERY_UDP
ZM_CN01_NO_RESPONSE_UDP
ZM_CN01_NS_RECORD_NOT_AA_UDP
ZM_CN01_SOA_RECORD_NOT_AA_UDP
ZM_CN01_UNEXPECTED_RCODE_NS_QUERY_UDP
ZM_CN01_UNEXPECTED_RCODE_SOA_QUERY_UDP
ZM_CN01_WRONG_NS_RECORD_UDP
ZM_CN01_WRONG_SOA_RECORD_UDP
ZM_CN02_MISSING_NS_RECORD_TCP
ZM_CN02_MISSING_SOA_RECORD_TCP
ZM_CN02_NO_RESPONSE_NS_QUERY_TCP
ZM_CN02_NO_RESPONSE_SOA_QUERY_TCP
ZM_CN02_NO_RESPONSE_TCP
ZM_CN02_NS_RECORD_NOT_AA_TCP
ZM_CN02_SOA_RECORD_NOT_AA_TCP
ZM_CN02_UNEXPECTED_RCODE_NS_QUERY_TCP
ZM_CN02_UNEXPECTED_RCODE_SOA_QUERY_TCP
ZM_CN02_WRONG_NS_RECORD_TCP
ZM_CN02_WRONG_SOA_RECORD_TCP
ZM_DEL_NS_SAME_IP
ZM_DIFFERENT_SOURCE_IP
ZM_DNSKEY_SMALLER_THAN_REC
ZM_DNSKEY_TOO_LARGE_FOR_ALGO
ZM_DNSKEY_TOO_SMALL_FOR_ALGO
ZM_DS01_DS_ALGO_2_MISSING
ZM_DS01_DS_ALGO_DEPRECATED
ZM_DS01_DS_ALGO_NOT_DS
ZM_DS01_DS_ALGO_RESERVED
ZM_DS02_DNSKEY_NOT_FOR_ZONE_SIGNING
ZM_DS02_DNSKEY_NOT_SEP
ZM_DS02_DNSKEY_NOT_SIGNED_BY_ANY_DS
ZM_DS02_NO_DNSKEY_FOR_DS
ZM_DS02_NO_MATCHING_DNSKEY_RRSIG
ZM_DS02_NO_MATCH_DS_DNSKEY
ZM_DS02_NO_VALID_DNSKEY_FOR_ANY_DS
ZM_DS02_RRSIG_NOT_VALID_BY_DNSKEY
ZM_DS08_DNSKEY_RRSIG_EXPIRED
ZM_DS08_DNSKEY_RRSIG_NOT_YET_VALID
ZM_DS08_MISSING_RRSIG_IN_RESPONSE
ZM_DS08_NO_MATCHING_DNSKEY
ZM_DS08_RRSIG_NOT_VALID_BY_DNSKEY
ZM_DS09_MISSING_RRSIG_IN_RESPONSE
ZM_DS09_NO_MATCHING_DNSKEY
ZM_DS09_RRSIG_NOT_VALID_BY_DNSKEY
ZM_DS09_SOA_RRSIG_EXPIRED
ZM_DS09_SOA_RRSIG_NOT_YET_VALID
ZM_DS10_INCONSISTENT_NSEC_NSEC3
ZM_DS10_MISSING_NSEC_NSEC3
ZM_DS10_MIXED_NSEC_NSEC3
ZM_DS10_NAME_NOT_COVERED_BY_NSEC
ZM_DS10_NAME_NOT_COVERED_BY_NSEC3

ZM_DS10_NON_EXISTENT_RESPONSE_ERROR
ZM_DS10_NSEC3_MISSING_SIGNATURE
ZM_DS10_NSEC3_RRSIG_VERIFY_ERROR
ZM_DS10_NSEC_MISSING_SIGNATURE
ZM_DS10_NSEC_RRSIG_VERIFY_ERROR
ZM_DS10_UNSIGNED_ANSWER
ZM_DS13_ALGO_NOT_SIGNED_DNSKEY
ZM_DS13_ALGO_NOT_SIGNED_NS
ZM_DS13_ALGO_NOT_SIGNED_SOA
ZM_DURATION_LONG
ZM_EDNS_RESPONSE_WITHOUT_EDNS
ZM_EDNS_VERSION_ERROR
ZM_EMPTY_ASN_SET
ZM_ERROR_ASN_DATABASE
ZM_EXTRA_ADDRESS_CHILD
ZM_EXTRA_NAME_PARENT
ZM_EXTRA_PROCESSING_BROKEN
ZM_IN_BAILIWICK_ADDR_MISMATCH
ZM_IPV4_ONE_ASN
ZM_IPV6_ONE_ASN
ZM_IS_A_RECURSOR
ZM_IS_NOT_AUTHORITATIVE
ZM_MISSING_OPT_IN_TRUNCATED
ZM_MNAME_DISCOURAGED_DOUBLE_DASH
ZM_MNAME_HAS_NO_ADDRESS
ZM_MNAME_NON_ALLOWED_CHARS
ZM_MNAME_NUMERIC_TLD
ZM_MULTIPLE_NS_SET
ZM_MULTIPLE_SOA
ZM_MULTIPLE_SOA_MNAMES
ZM_MULTIPLE_SOA_RNAMES
ZM_MULTIPLE_SOA_TIME_PARAMETER_SET
ZM_N10_EDNS_RESPONSE_ERROR
ZM_N10_NO_RESPONSE_EDNS1_QUERY
ZM_N10_UNEXPECTED_RCODE
ZM_N11_NO_EDNS
ZM_N11_NO_RESPONSE
ZM_N11_RETURNS_UNKNOWN_OPTION_CODE
ZM_N11_UNEXPECTED_ANSWER_SECTION
ZM_N11_UNEXPECTED_RCODE
ZM_N11_UNSET_AA
ZM_NAMESERVER_IP_PRIVATE_NETWORK
ZM_NAMESERVER_IP_WITHOUT_REVERSE
ZM_NOT_ENOUGH_IPV4_NS_CHILD
ZM_NOT_ENOUGH_IPV4_NS_DEL
ZM_NOT_ENOUGH_IPV6_NS_CHILD
ZM_NOT_ENOUGH_IPV6_NS_DEL
ZM_NOT_ENOUGH_NS_CHILD
ZM_NOT_ENOUGH_NS_DEL
ZM_NO_DNSKEY
ZM_NO_EDNS_SUPPORT
ZM_NO_IPV4_NS_CHILD
ZM_NO_IPV4_NS_DEL
ZM_NO_IPV6_NS_CHILD
ZM_NO_IPV6_NS_DEL
ZM_NO_RESOLUTION
ZM_NO_RESPONSE
ZM_NO_RESPONSE_DNSKEY
ZM_NO_RESPONSE_NS_QUERY
ZM_NO_RESPONSE_PTR_QUERY
ZM_NO_RESPONSE_SOA_QUERY
ZM_NO_SOA_IN_RESPONSE
ZM_NS_ERROR
ZM_NS_IS_CNAME
ZM_OUT_OF_BAILIWICK_ADDR_MISMATCH
ZM_QNAME_CASE_INSENSITIVE
ZM_REFERRAL_SIZE_TOO_LARGE
ZM_REMAINING_LONG
ZM_REMAINING_SHORT
ZM_RNAME_MAIL_DOMAIN_INVALID
ZM_RNAME_MAIL_DOMAIN_LOCALHOST
ZM_RNAME_MAIL_ILLEGAL_CNAME
ZM_RNAME_MISUSED_AT_SIGN

ZM_RNAME_RFC822_INVALID
ZM_RRSIG_EXPIRED
ZM_SAME_IP_ADDRESS
ZM_TOO_MANY_ITERATIONS
ZM_TOTAL_NAME_MISMATCH
ZM_UNEXPECTED_RCODE
ZM_WRONG_SOA
ZM_Z_FLAGS_NOTCLEAR

codeRef (optional)
String a link to a description for the error code (if any). format: url

severity
String The log level of the message, a subset of the values defined in RFC5424.
Enum:
WARNING
ERROR
CRITICAL

timestamp
Date the timestamp when the message was generated. format: date-time

message
String the message.

`testRequest` - Up

This object type is used to define the properties of a new test request object. The `testRequestSubmitted` type inherits from it.

applicationID (optional)
String For RSP evaluation, the application ID, otherwise it should be omitted. In OT&E, this **MUST** be omitted when creating a new test request.

rsp (optional)
String

The RSP's unique ID.

In OT&E, this **MUST** be omitted when creating a new test request, and will be populated using the FQDN of the first `TLSA` record which matches the certificate presented by the client.

tlds
array[array[tldInfo]]

This structure describes the TLDs to which the test relates. It is an array which contains an array of logically grouped TLDs (such as those with a variant relationship).

Simple example of an ASCII TLD with no IDN tables:

```
{
  "tlds": [
    [
      {
        "string": "example",
        "idnTables": []
      }
    ]
  ]
}
```

In RSP testing, this property will only contain a single TLD, but more complex sets of TLDs with variants are supported for pre-delegation testing.

Example of an ASCII TLD with one or more IDN tables:

```
{
  "tlds": [
    [
      {
        "string": "example",
        "idnTables": [
          {
```

```
            "id": "06e6ab5b-0e7a-4ff2-8e67-d6320e5ef4b7",
            "variantSupportLevel": 0,
          },
          {
            "id": "25eb306b-1fb0-4def-bf01-fa18815f614b",
            "variantSupportLevel": 0,
          }
        ]
      }
    ]
  ]
}
```

Example of a set of variant TLDs:

```
{
  "tlds": [
    [
      {
        "string": "xn--8pvxs",
        "idnTables": [
          {
            "id": "06e6ab5b-0e7a-4ff2-8e67-d6320e5ef4b7",
            "variantSupportLevel": 3,
          }
        ]
      },
      {
        "string": "xn--8pvz8d",
        "idnTables": [
          {
            "id": "3f60939b-b7bf-46db-9004-126bf08923af",
            "variantSupportLevel": 3,
          }
        ]
      }
    ]
  ]
}
```

All the IDN tables referenced in this property **MUST** already exist when the test object is created. In OT&E, users can create IDN table obejcts, but these **MUST** correspond to Second-Level Reference LGRs: custom IDN tables cannot be used.

If a TLD offers registrations at the third or higher levels, then at least one second-level "registry-class" domain name(s) should be separately listed, rather than the TLD itself.

clientIDs (optional)
array[String]

An array of FQDNs at which one or more `TLSA` records may be found which can be used for authentication.

In OT&E, this **MUST** be omitted when creating a new test request, but the resulting test will be populated with the FQDN of the first `TLSA` record which matches the certificate presented by the client.

format: hostname

testPlan
String The test plan to be followed. A list of test plans and the permitted values for this property may be found at https://icann.github.io/rst-test-specs/rst-test-specs.html#test-plans.

Enum:
StandardPreDelegationTest
StandardRSPChangeTest
DNSRSPChangeTest
StandardIDNTest
RSPEvaluationIDNTest
SRSGatewayTest
MainRSPEvaluationTest
DNSRSPEvaluationTest
DNSSECRSPEvaluationTest

SRSGatewayRSPTest
StandardDNSOnly
StandardDNSSECOnly
StandardRDAPOnly
StandardEPPOnly
StandardRDEOnly
StandardSRSGatewayOnly
MinimumRPMsOnly

dueDate (optional)
Date

The date/time before which the test must be passed. If not provided, the test remains open indefinitely (unless the `completed` status is reached).

In OT&E, this **MUST** be omitted when creating a new test request.

format: date-time

`testRequestSubmitted` - Up

This type describes a test request object that has been successfully submitted. It inherits all the properties defined in the `testRequest` type.

applicationID (optional)
String For RSP evaluation, the application ID, otherwise it should be omitted. In OT&E, this **MUST** be omitted when creating a new test request.

rsp (optional)
String

The RSP's unique ID.

In OT&E, this **MUST** be omitted when creating a new test request, and will be populated using the FQDN of the first `TLSA` record which matches the certificate presented by the client.

tlds
array[array[tldInfo]]

This structure describes the TLDs to which the test relates. It is an array which contains an array of logically grouped TLDs (such as those with a variant relationship).

Simple example of an ASCII TLD with no IDN tables:

```
{
  "tlds": [
    [
      {
        "string": "example",
        "idnTables": []
      }
    ]
  ]
}
```

In RSP testing, this property will only contain a single TLD, but more complex sets of TLDs with variants are supported for pre-delegation testing.

Example of an ASCII TLD with one or more IDN tables:

```
{
  "tlds": [
    [
      {
        "string": "example",
        "idnTables": [
          {
            "id": "06e6ab5b-0e7a-4ff2-8e67-d6320e5ef4b7",
            "variantSupportLevel": 0,
          },
          {
            "id": "25eb306b-1fb0-4def-bf01-fa18815f614b",
```

```
            "variantSupportLevel": 0,
          }
        ]
      }
    ]
  ]
}
```

Example of a set of variant TLDs:

```
{
  "tlds": [
    [
      {
        "string": "xn--8pvxs",
        "idnTables": [
          {
            "id": "06e6ab5b-0e7a-4ff2-8e67-d6320e5ef4b7",
            "variantSupportLevel": 3,
          }
        ]
      },
      {
        "string": "xn--8pvz8d",
        "idnTables": [
          {
            "id": "3f60939b-b7bf-46db-9004-126bf08923af",
            "variantSupportLevel": 3,
          }
        ]
      }
    ]
  ]
}
```

All the IDN tables referenced in this property **MUST** already exist when the test object is created. In OT&E, users can create IDN table obejcts, but these **MUST** correspond to Second-Level Reference LGRs: custom IDN tables cannot be used.

If a TLD offers registrations at the third or higher levels, then at least one second-level "registry-class" domain name(s) should be separately listed, rather than the TLD itself.

clientIDs (optional)
array[String]

An array of FQDNs at which one or more `TLSA` records may be found which can be used for authentication.

In OT&E, this **MUST** be omitted when creating a new test request, but the resulting test will be populated with the FQDN of the first `TLSA` record which matches the certificate presented by the client.

format: hostname

testPlan
String The test plan to be followed. A list of test plans and the permitted values for this property may be found at https://icann.github.io/rst-test-specs/rst-test-specs.html#test-plans.

Enum:

StandardPreDelegationTest
StandardRSPChangeTest
DNSRSPChangeTest
StandardIDNTest
RSPEvaluationIDNTest
SRSGatewayTest
MainRSPEvaluationTest
DNSRSPEvaluationTest
DNSSECRSPEvaluationTest
SRSGatewayRSPTest
StandardDNSOnly
StandardDNSSECOnly
StandardRDAPOnly
StandardEPPOnly

StandardRDEOnly
StandardSRSGatewayOnly
MinimumRPMsOnly

dueDate (optional)
Date

The date/time before which the test must be passed. If not provided, the test remains open indefinitely (unless the `completed` status is reached).

In OT&E, this **MUST** be omitted when creating a new test request.

format: date-time

testID (optional)
String The unique ID for this test request object.

testPlanVersion
String The version of the Test Plan that will be used for the test. This will be determined using the `Version` property of the RST Test Specifications and follows the Semantic Versioning convention.

dateRequested
Date date/time when this request was submitted. format: date-time

dateUpdated (optional)
Date date/time when this request was last updated. format: date-time

dateStarted (optional)
Date date/time when the test run (if any) started. format: date-time

dateCompleted (optional)
Date date/time when the test run (if any) completed. format: date-time

status
testStatus

result
testResult

errorCodes (optional)
array[String] if the result of the test is a fail or an error, then this property will contain any `ERROR` or `CRITICAL` error codes generated by the test run. Otherwise it will be omitted.

inputs
inputParameters

missingInputs (optional)
array[String] An array listing any required input parameters that have not yet been provided.

files (optional)
array[testRequestSubmitted_files_inner] A list of any files uploaded.

results (optional)
array[testRunLog]

A test may result in multiple "runs". Each run is represented by a `testRunLog` which contains a number of `testCaseLog` entries.

The `results` property is an array of `testRunLog` objects. This property will initially be an empty array, until a test run is started. When the test run starts, an empty `testRunLog` object will be added to the array. As the test progresses, `testCaseLog` entries will be appended to the `testRunLog` object.

testRequestSubmitted_files_inner - Up

name (optional)
String the file name.

type (optional)
String the media type of the file.

uploaded (optional)
Date when the file was uploaded. format: date-time

href (optional)
String the URL of the file. format: url

`testResult` - [Up](#)

A string indicating the result of a test request object.

1. The `pass` value indicates that the test run completed with no errors.
2. The `fail` value indicates that at least one test case failed.
3. The `exception` value indicates that an internal issue prevented the test run from completing.

`testRunLog` - [Up](#)

an object representing a discrete test run.

runID
String unique ID for this test run.

dateStarted
String the date and time the run started. format: datetime

dateCompleted
String the date and time the run finished. format: datetime

result
testResult

log
array[testCaseLog] an array of test case log objects.

`testStatus` - [Up](#)

A string indicating the status of a test request object.

1. The `inputs-needed` value indicates that the request has been received, but input parameters are needed before the test can begin.
2. The `inputs-complete` value indicates that all the required input parameters and files have been provided. The test is therefore ready to start.
3. The `in-progress` value indicates that a test run is in progress.
4. The `completed` value indicates that the test run has completed. If a test object has this status, then its `result` property will indicate the outcome of the test.
5. The `expired` value indicates that the test did not have a status of `completed` when the `dueDate` was reached.

- State diagram

`tldInfo` - [Up](#)

A top-level domain (or equivalent registry-class domain name) and its associated IDN tables.

name
String The TLD (or equivalent registry-class domain name). format: hostname

idnTables
array[idnTableRef] the IDN tables(s) for the TLD.