strongTNC REST API Draft 1

Part of the strongTNC Bachelor's Thesis

Danilo Bargen, Christian Fässler, Jonas Furrer

Spring 2014

Contents

1 Introduction

The REST resource design is based on the recommendations of the *REST API Design Rulebook*[1] from O'Reilly.

URI Definition

URIs[1] are defined as follows, according to RFC 3986[2]:

```
URI = scheme "://" authority "/" path [ "?" query ] [ "#" fragment ]
```

Resource-Archetypes

We used the resource archetype terminology from Masse 2011[1]. The explanation texts are taken directly from said book.

Document A document resource is a singular concept that is akin to an object instance or database record. A document's state representation typically includes both fields with values and links to other related resources.

Collection A collection resource is a server-managed directory of resources. Clients may propose new resources to be added to a collection. However, it is up to the collection to choose to create a new resource, or not.

Store A store is a client-managed resource repository. A store resource lets an API client put resources in, get them back out, and decide when to delete them. On their own, stores do not create new resources; therefore a store never generates new URIs. Instead, each stored resource has a URI that was chosen by a client when it was initially put into the store.

Controller A controller resource models a procedural concept. Controller resources are like executable functions, with parameters and return values; inputs and outputs. Like a traditional web application's use of HTML forms, a REST API relies on controller resources to perform application-specific actions that cannot be logically mapped to one of the standard methods (create, retrieve, update, and delete, also known as CRUD).

[1] Uniform Resource Identifier

2 Archetype Representation

The JSON representation in this API documentation depends on the resource archetype.

Document A document returns a JSON object, which contains all relevant fields.

If a document contains references to other resources, then this reference is returned as an URI. Additionally, a `depth` query parameter can be specified to embed nested resources directly into the response.

Example:

```
{
  "field1": "<str,value-1>",
  "field2": <int,value-2>,
  "field3": <bool,value-3>,
  "subObject": "<uri,sub-object>"
  "subCollection": [
    {
      "uri": "<uri,sub-object>"
    },
    {
      "uri": "<uri,sub-object>"
    }
  ]
}
```

Example with `depth=1`:

```
{
  "field1": "<str,value-1>",
  "field2": <int,value-2>,
  "field3": <bool,value-3>,
  "subObject": {
    "field1": <int,value-1>,
    ...
    "uri": "<uri,sub-object>"
  }
  "subCollection": [
    {
      "field": <int,field>,
      ...
      "uri": "<uri,sub-object>"
    },
    {
      "field": <int,field>,
      ...
      "uri": "<uri,sub-object>"
    }
  ]
}
```

Collection A collection returns a (possibly filtered) list of all contained documents as JSON-objects. The documents are annotated with an additional field called `uri`, which contains the URI to the document resource.
Collections also allow the usage of a `depth` parameter to embed nested objects.
Example:

```
{
  {
    "field1": "<str,value-1>",
    "subObject": "<uri,sub-object>",
    "uri": "<uri,objek>"
  },
  {
    "field1": "'<str,value-1>",
    "subObject": "<uri,sub-object>",
    "uri": "<uri,resource>"
  },
}
```

Example with `depth=1`:

```
{
  {
    "field1": "<str,value-1>",
    "subObject": {
      "field": "<str,value>",
    },
    "uri": "<uri,resource>"
  },
  {
    "field1": "'<str,value-1>",
    "subObject": {
      "field": "<str,value>",
    },
    "uri": "<uri,resource>"
  },
}
```

Controller The output of a controller varies, depending on the implementation.

3 HTTP Status Codes

The result of a request is sent by means of HTTP status codes. In some cases, additional information is returned in the response body.

The following status codes can be expected:

200 OK The request has succeeded.
201 Created The request has been fulfilled and resulted in a new resource being created.
204 No Content The server has fulfilled the request but does not need to return an entity-body.
400 Bad Request Generic client side error.
404 Not Found The server has not found anything matching the Request-URI.
405 Method Not Allowed The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
409 Conflict The entity to be created already exists.
412 Precondition Failed Additional steps are necessary to successfully process the sent request.
500 Internal Server Error Generic server side error.

If a resource uses additional error codes, it's documentation can be found in the corresponding resource documentation

4 Further Remarks

4.1 Notation

For the request- and response-format of a resource, pseudo-JSON notation is used. The values are denoted as a tuple in angle brackets. The first part of the tuple denotes the type, while the second is a description of the value.

The following types are used:

int Integer
num Decimal
str String
bool Boolean
xml An XML document
hex A HEX string
uri The fully qualified URI of a resource
doc The resource document

4.2 Default Behavior

The following items describe the default behavior of resources:

Depth Query-Parameter Each resource that returns an URI to another resource, supports a `depth` query parameter. This parameter embeds the nested resources into the returned document up to the depth specified by the query parameter. If no parameter is specified, the depth `0` is used.

Filter Query-Parameter Filter query parameters are – if available – always optional. They can be used to filter the returned data.

All Collections support – unless specified otherwise – generic filters on all fields except foreign keys. The parameter looks like this: `fieldName=query`. Filters can also be used on document resources – if the document field values don't match, then a HTTP 404 status code is returned.

Fields Query-Parameter The `fields` query parameter can be used to limit the returned fields of a resource. This is similar to a `SELECT field1, field2, fieldn` SQL statement.

5 Data Definitions

5.1 Policy Arguments

The following workitem- and policy-types are currently available:

`00:` `RESVD` Deny
`01:` `PCKGS` Installed Packages
`02:` `UNSRC` Unknown Source
`03:` `FWDEN` Forwarding Enabled
`04:` `PWDEN` Default Password Enabled
`05:` `FREFM` File Reference Measurement
`06:` `FMEAS` File Measurement
`07:` `FMETA` File Metadata
`08:` `DREFM` Directory Reference Measurement
`09:` `DMEAS` Directory Measurement
`10:` `DMETA` Directory Metadata
`11:` `TCPOP` Open TCP Listening Ports
`12:` `TCPBL` Blocked TCP Listening Ports
`13:` `UDPOP` Open UDP Listening Ports
`14:` `UDPBL` Blocked UDP Listening Ports
`15:` `SWIDT` SWID Tag Inventory
`16:` `TPMRA` TPM Remote Attestation

Some of the types have different parameters. They can be grouped as follows:

No arguments 00, 01, 02, 03, 04
File path 05, 06, 07
Directory path 08, 09, 10
Port list 11, 12, 13, 14
SWID request flags 15
TPM attestation flags 16

Depending on the type, the following arguments need to be submitted:

No arguments

```
{}
```

File path

```
{
  "file": "<str,file-path>"
}
```

Directory path

```
{
  "directory": "<str,directory-path>"
}
```

Port list

```
{
  "portList": [
    "<str,port or port-range>"
  ]
}
```

SWID request flags

```
{
  "swidFlags": [
    "<str,swid-flag>"
  ]
}
```

TPM attestation flags

```
{
  "tpmFlags": [
    "<str,tpm-flag>"
  ]
}
```

5.2 Recommendation Types

The following recommendation types and -actions are available:

`0:` `ALLOW` It is recommended to allow access to the client
`1:` `BLOCK` It is recommended to deny access to the client
`2:` `ISOLATE` It is recommended to isolate the client
`3:` `NONE`

The values should be submitted to the API as integer values.

5.3 Hash-Set

A hash-set is used in the file documents. They are structured as follows:

```
[
  {
    "algorithm": "<str,algorithm-name>",
    "hash": "<hex,hash> ",
    "product": "<uri,product>"
  }
]
```

If an algorithm doesn't exist yet, it will be created automatically. Current default algorithms are `SHA384, SHA256, SHA1, SHA1-IMA`.

6 REST Resources

6.1 Session control

6.1.1 Controller

URI Path `/sessionss/start/`
Archetype Controller
Methods POST
Request Parameters

`connectionId` strongSwan Connection ID
`clientIdentity` strongSwan Client-Identity
`hardwareId` The ID that identifies the device, for example the AIK, Android-ID, DBUS Machine-ID, etc.
`productName` The productName is the name of the operating system as present in the `product` table of the database.

JSON Formatted Response

```
{
  "sessionId": <int,id>,
  "workitems": [
     <doc,workitem>,
     ...
  ],
  "uri": "<uri,session>"
}
```

Description This controller creates and starts a session. The device to be assigned to the session is determined by the `hardwareId` and the `productName`. If one of the related objects does not exist yet, it will be created by this controller. The ID which is returned in the response document is used to identify the initiated session. Furthermore, a list of workitems is returned in the response. These workitems should be processed in the corresponding session.

URI Path `/sessions/{id}/end/`
Archetype Controller
Methods POST
Request Parameters

`recommendation` Final result / recommendation for this session.

Description This controller finalizes the session and starts all related processes. Among other things, the workitems related to this session will be removed, their results will be stored and can be accessed via `/sessions/{id}/results`.

6.1.2 Documents and Collections

URI Path `/sessions/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "time": <int,time>,
  "identity": "<uri,identity>",
  "connectionId": <int,connection-id>,
  "device": "<uri,device>",
  "recommendation": <int,rec>
}
```

Description Information related to a specific session. Sessions should never be changed directly, changes can only be made via the corresponding controllers. The reason for this is that a change in the session requires more processing tasks in the background.

URI Path `/sessions/`
Archetype Readonly Collection
Filter Query
timeFrom `<int,timestamp>`
timeTo `<int,timestamp>`
Methods GET
JSON Formatted Response

```
[<doc,session>, ...]
```

Description List of all sessions. New sessions can only be created via the corresponding controllers.

Workitems

URI Path `/sessions/{id}/workitems/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "session": "<uri,session>",
  "type": <int,type>,
  "argument": <policy-argument>
}
```

Description A workitem related to the specified session. Workitems cannot be created directly, they will be created automatically during the session initialization depending on the configured enforcements. Workitems only exist as long as a session is active. After the session has ended, a `HTTP 404` status code will be returned.

URI Path `/sessions/{id}/workitems/`
Archetype Readonly Collection
Filter Query
type `<int,policy-type>`
Methods GET
JSON Formatted Response

```
[<doc,workitem>, ...]
```

Description A list of all workitems that belong to the specified session. Workitems cannot be created directly, they will be created automatically during the session initialization depending on the configured enforcements. Workitems only exist as long as a session is active. After the session has ended, an empty list will be returned.

URI Path `/sessions/{id}/workitems/{id}/result/`

Archetype Document

Request Parameters

`recommendation` Result / recommendation for this workitem.

`comment` Comment concerning the result.

Methods GET, POST

Response Statuscodes

201 Created Result has been saved successfully.

409 Conflict Result already exists.

JSON Formatted Response

```
{
  "recommendation": <int,type>,
  "comment": <str,comment>
}
```

Description To this resource the result of the corresponding workitem should be submitted. The results are transferred from the workitem to the session result when the session is closed.

Results

URI Path `/sessions/{id}/results/{id}/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "enforcement": "<uri,enforcement>",
  "recommendation": <int,type>,
  "comment": "<str,comment>"
}
```

Description A single result. In contrast to the result resource (relative to the workitem resource), this document also contains a link to the corresponding enforcement.

URI Path `/sessions/{id}/results/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,result>, ...]
```

Description After closing a session, this collection allows the querying of results. The collection is readonly, in order that session results cannot be changed afterwards.

6.2 SWID Extension to strongTNC

6.2.1 SWID Tags: Measurement

URI Path `/sessions/{id}/swid-measurement/`
Archetype Controller
Methods POST
Content-Type `application/json; charset=utf-8`
Request Parameters
`softwareId` Software-IDs as JSON-list.
Response Statuscodes
200 OK SWID Tags for the submitted software IDs alreay exist and have been linked with the current session.
404 Not Found Session with the specified ID could not be found.
412 Precondition Failed For some of the submited software IDs no SWID tags could be found. The response body contains a list of those software IDs.
JSON Formatted Response

```
["<str,software-id>", ...]
```

Description A list of software IDs can be submitted to this endpoint. If SWID tags already exist for all submitted software IDs, they are linked with the current session and a `HTTP 200` status code is returned.
Otherwise, a list of missing software IDs is returned. The requester first has to add these SWID tags via the SWID tag endpoint. Afterwards he can repeat the original request.

6.2.2 SWID Tags: Creation

URI Path `/swid/add-tags/`
Archetype Controller
Methods POST
Content-Type `application/json; charset=utf-8`
Request Parameters
`xmlData` SWID Tag as JSON-list.
Response Statuscodes
200 OK SWID tags have been processed successfully.
400 Bad Request Details regarding the error are contained in the response body.
Description The submitted tags are stored in the database. If a tag already exists, it is ignored.

6.3 CRUD Resources

6.3.1 Products

URI Path `/products/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "name": "<str,productname>"
}
```

URI Path `/products/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "name": "<str,productname>"
  }
]
```

URI Path `/products/{id}/default-groups/`
Archetype Collection
Methods GET, POST
Request Parameters

`groupId` group-id

JSON Formatted Response

```
[<doc,group>, ...]
```

6.3.2 Packages

URI Path `/packages/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "name": "<str,packagename>"
}
```

URI Path `/packages/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "name": "<str,packagename>",
  }
]
```

URI Path `/packages/{id}/versions/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[<doc,version>, ...]
```

6.3.3 Versions

URI Path `/versions/{id}/`
Archetype Document
Methods GET, PUT, PATCH, DELETE
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "package": "<uri,package>",
  "product": "<uri,product>",
  "release": "<str,release>",
  "securtiy": <int,security>,
  "blacklist": <bool,blacklist>,
  "time": <int,time>
}
```

URI Path `/versions/`
Archetype Collection
Filter Query
productName `<str,product-name>`
packageName `<str,package-name>`
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "package": "<uri,package>",
    "product": "<uri,product>",
    "release": "<str,release>",
    "securtiy": <int,security>,
    "blacklist": <bool,blacklist>,
    "time": <int,time>
  }
]
```

6.3.4 Identities

URI Path `/identities/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "type": <int,type>,
  "value": "<str,value>"
}
```

URI Path `/identities/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "type": <int,type>,
    "value": "<str,value>"
  }
]
```

6.3.5 Devices

URI Path `/devices/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "description": "<str,description>",
  "value": "<str,value>",
  "product": "<uri,product>",
  "created": <int,created>
}
```

URI Path `/devices/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,\\
    "uri": "<uri,resource>",
    "description": "<str,description>",
    "value": "<str,value>",
    "product": "<uri,product>",
    "created": <int,created>,
  }
]
```

URI Path `/device/{id}/sessions/`
Archetype Readonly Collection
Filter Query
timeFrom `<int,timestamp>`
timeTo `<int,timestamp>`
Methods GET
JSON Formatted Response

```
[<doc,session>, ...]
```

URI Path `/device/{id}/sessions/{id}/results/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,result>, ...]
```

URI Path `/device/{id}/groups/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[<doc,group>, ...]
```

URI Path `/device/{id}/swid-tags/`
Archetype Readonly Collection
Methods POST
JSON Formatted Response

```
[<doc,swid-tag>, ...]
```

6.3.6 Enforcements

URI Path `/enforcements/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "policy": "<uri,policy>",
  "group": "<uri,group>",
  "failRecommendation": <int,rec_fail>,
  "noresultRecommendation": <int,rec_noresult>",
  "maxAge": <int,max_age>
}
```

URI Path `/enforcements/`
Archetype Readonly Collection
Methods GET
Filter Query

groupName `<str,group-name>`
policyName `<str,policy-name>`

JSON Format

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "policy": "<uri,policy>",
    "group": "<uri,group>",
    "failRecommendation": <int,rec_fail>,
    "noresultRecommendation": <int,rec_noresult>",
    "maxAge": <int,max_age>
  }
]
```

URI Path `/enforcements/{id}/groups/`
Archetype Readonly Collection
Methods GET
JSON Format

```
[<doc,group>, ...]
```

6.3.7 Policies

URI Path `/policies/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "type": <int,type>,
  "name": "<str,name>",
  "argument": <policy-argument>,
  "failRecommendation": <int,rec_fail>,
  "noresultRecommendation": <int,rec_noresult>",
}
```

URI Path `/policies/`
Archetype Readonly Collection
Methods GET
JSON Format

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "type": <int,type>,
    "name": "<str,name>",
    "argument": <policy-argument>,
    "failRecommendation": <int,rec_fail>,
    "noresultRecommendation": <int,rec_noresult>"
  }
]
```

URI Path `/policies/{id}/enforcements/`
Archetype Readonly Collection
Methods GET
JSON Format

```
[<doc,enforcement>, ...]
```

6.3.8 Groups

URI Path `/groups/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "name": "<str,name>",
  "parent": "<uri,group>"
}
```

URI Path `/groups/`
Archetype Readonly Collection
Methods GET
JSON Format

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "name": "<str,name>",
    "parent": "<uri,group>"
  }
]
```

URI Path `/groups/{id}/devices/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[<doc,device>, ...]
```

URI Path `/groups/{id}/enforcements/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,enfocement>, ...]
```

6.3.9 Files

URI Path `/files/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "name": "<str,name>",
  "hashes: <doc,hash-set>,
  "dir": "<uri,directory>"
}
```

URI Path `/files/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "name": "<str,name>"
    "hashes: <doc,hash-set>,
    "dir": "<uri,directory>"
  }
]
```

6.3.10 Directories

URI Path `/directories/{id}/`
Archetype Document
Methods GET, PUT, PATCH
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "path": "<str,path>"
}
```

URI Path `/directories/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "path": "<str,path>"
  }
]
```

URI Path `/directories/{id}/files/`
Archetype Collection
Methods GET, POST
JSON Formatted Response

```
[<doc,file>, ...]
```

6.3.11 SWID Tags

URI Path `/swid-tags/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,swid-tag>, ...]
```

URI Path `/swid-tags/{id}/`
Archetype Readonly Document
Methods GET
Filter Query

packageName `<str,package-name>`
version `<str,version>`
uniqueId `<str,unique-id>`

JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "packageName": "<str,package-name>",
  "version": "<str,version>",
  "uniqueId": "<str,unique-id>",
  "entities": [
    {
    "entity": "<uri,entity>",
    "role":  <int,role>
    }
  ],
  "tagXml": "<xml,swid-tag>"
}
```

URI Path `/swid-tags/{id}/files/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,file>, ...]
```

6.3.12 Entities

URI Path `/swid-entities/{id}/`
Archetype Readonly Document
Methods GET
JSON Formatted Response

```
{
  "id": <int,id>,
  "uri": "<uri,resource>",
  "name": "<str,name>",
  "regid": "<str,regid>"
}
```

URI Path `/swid-entities/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[
  {
    "id": <int,id>,
    "uri": "<uri,resource>",
    "name": "<str,name>",
    "regid": "<str,regid>",
  }
]
```

URI Path `/swid-entities/{id}/swid-tags/`
Archetype Readonly Collection
Methods GET
JSON Formatted Response

```
[<doc,swid-tag>, ...]
```

References

[1] Masse, Mark. *REST API design rulebook.* O'Reilly Media, Inc., 2011.

[2] Berners-Lee, Tim, Roy Fielding, and Larry Masinter. *RFC 3986: Uniform resource identifier (uri): Generic syntax.* The Internet Society (2005).