

# **strongTNC REST API**

Im Rahmen der strongTNC BA

Danilo Bargaen, Christian Fässler, Jonas Furrer

Frühlingssemester 2014

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Repräsentation der Archetypen</b>	<b>4</b>
<b>3</b>	<b>HTTP Statuscodes</b>	<b>6</b>
<b>4</b>	<b>Allgemeine Hinweise</b>	<b>7</b>
4.1	Schreibweise . . . . .	7
4.2	Standard Verhalten . . . . .	7
<b>5</b>	<b>Daten Definition</b>	<b>8</b>
5.1	Policy Arguments . . . . .	8
5.2	Recommendation Types . . . . .	9
5.3	Hash-Set . . . . .	9
<b>6</b>	<b>REST Ressourcen</b>	<b>10</b>
6.1	Session Steuerung und Ablauf . . . . .	10
6.1.1	Controller . . . . .	10
6.1.2	Documents und Collections . . . . .	11
6.2	SWID Erweiterung . . . . .	15
6.2.1	SWID Tags: Messung . . . . .	15
6.2.2	SWID Tags: Erstellung . . . . .	15
6.3	CRUD Ressourcen . . . . .	16
6.3.1	Products . . . . .	16
6.3.2	Packages . . . . .	17
6.3.3	Versions . . . . .	18
6.3.4	Identities . . . . .	19
6.3.5	Devices . . . . .	20
6.3.6	Enforcements . . . . .	22
6.3.7	Policies . . . . .	23
6.3.8	Groups . . . . .	24
6.3.9	Files . . . . .	25
6.3.10	Directories . . . . .	26
6.3.11	SWID Tags . . . . .	27
6.3.12	Entities . . . . .	28

# 1 Einleitung

Die Definition der Ressourcen orientiert sich an den Regeln des Buches *REST API Design Rulebook*[1] aus dem O'Reilly Verlag.

## URI Definition

Bei der Bezeichnung der URIs<sup>1</sup> wurde folgende Terminologie gemäss RFC 3986[2] verwendet:

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

## Ressource-Archetypen

Nachfolgend die Ressource-Archetypen gemäss Masse 2011[1]. Die Erklärungstexte wurden direkt dem besagten Buch entnommen.

**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).

---

<sup>1</sup>Uniform Resource Identifier

## 2 Repräsentation der Archetypen

Die JSON-Repräsentation ist abhängig vom Ressource-Archetypen.

**Document** Ein Document gibt ein JSON Objekt zurück, welches alle relevanten Felder enthält. Informationen, welche durch die Ressource-URI bereits gegeben sind (z.B. Type), müssen nicht, können jedoch, erneut in der Liste erscheinen.

Falls im Document Unterobjekte auftauchen, wird für das Feld die URI des betreffenden Unterobjekts eingesetzt, damit kann dieses Objekt direkt abgefragt werden. Zusätzlich wird ein Query-Parameter bereit gestellt, welcher es erlaubt, Unterobjekte bis zu einer gewissen Tiefe aufzulösen.

Beispiel:

```
{
  "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>"
    }
  ]
}
```

Beispiel mit Query 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** Eine Collection gibt eine Liste mit allen enthaltenen (ggf. gefilterten) JSON-Objekten zurück. Die Objekte werden zusätzlich um ein Feld `uri` ergänzt, welches die Ressource-

URI des jeweiligen Objektes enthält. Wie bei einem Document gibt es bei Collections einen optionalen Query Parameter welcher es erlaubt Unterobjekte aufzulösen,

Beispiel:

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

Beispiel mit Query depth=1:

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

**Store** Ein Store verhält sich wie eine Collection, wenn sie direkt angesprochen wird (`/store`) und wie ein Document, wenn ein spezifisches Element des Store angesprochen wird (`/store/{element-id}`).

**Controller** Der Output des Controllers ist abhängig vom Verwendungszweck.

### 3 HTTP Statuscodes

Das Ergebnis einer Anfrage wird grundsätzlich über einen HTTP Statuscode mitgeteilt, gewissen Fällen kann auch noch Payload im Body geliefert werden. Folgendes ist eine Auswahl von Statuscodes die zu erwarten sind:

**200 OK** Request erfolgreich

**201 Created** Entity wurde erfolgreich erstellt

**204 No Content** Entity wurde erfolgreich geändert

**400 Bad Request** Generischer Client Fehler

**404 Not Found** Entity nicht gefunden

**405 Method Not Allowed** Die verwendete HTTP Methode ist auf dieser Ressource nicht erlaubt

**409 Conflict** Zu erstellende Entity existiert bereits

**412 Precondition Failed** Es sind zusätzliche Schritte nötig um die Anfrage auszuführen.

**500 Internal Server Error** Generischer Server Fehler

Falls zusätzliche Codes verwendet werden oder die Codes eine aussergewöhnliche oder erwähnenswerte Bedeutung haben, ist dies bei der betroffenen Resource vermerkt.

## 4 Allgemeine Hinweise

### 4.1 Schreibweise

Für die Definition des Response- und Request-Formats wird eine pseudo JSON schreibweise verwendet. Die Werte werden als Tupel in spitzen Klammern dargestellt. Der erste Teil des Tupels zeigt den Typ, der zweite Teil ist eine Beschreibung für den Wert. Folgende Typen werden verwendet:

**int** Integer Zahlenwert

**num** Dezimalzahl

**str** String

**bool** Wahrheitswert

**xml** Ein XML Dokument

**hex** Ein HEX String

**uri** Die vollqualifizierte URI einer Resource

**doc** Das Dokument einer Resource, wie dieses aussieht kann im jeweiligen Abschnitt zur genannten Resource nachgeschlagen werden

### 4.2 Standard Verhalten

Folgende Punkte beschreiben das Standardverhalten der Ressourcen:

**Depth Query-Parameter** Jede Resource die als Antwort eine URI auf eine andere Resource zurück gibt unterstützt den **depth** Query-Parameter. Dieser löst die URIs bis zur gewünschten tiefe auf. Möglicherweise wird die maximale Tiefe eingeschränkt. Wenn keine tiefe angegeben wird, gilt die Auflösungstiefe 0.

**Filter Query-Parameter** Filter Query-Parameter sind, falls vorhanden, immer optional, sie dienen der Präzisierung des Rückgabesets. Wenn sie weggelassen werden, wird das vollständige Set zurückgegeben.

**Generische Filter** Collections unterstützen wenn nicht anders erwähnt generische Filter auf alle Feldern mit Ausnahme der Foreign Keys. Das heisst, es kann grundsätzlich nach jedem Attribut des Rückgabedokuments gefiltert werden. Der Parameter ist wie folgt aufgebaut: **attributeName=query**. Filter können auch auf Document-Resources genutzt werden, es wird bei nicht passenden Filterwerten ein HTTP 404 Status Code zurückgegeben.

**Batch Create** Collections unterstützen, sofern sie nicht readonly sind oder es anders erwähnt ist, das sogenannte 'Batch Create'. D.h zum Erstellen von neuen Dokumenten kann einer Collection auch eine Liste gesendet werden. Wird 'Batch Create' verwendet, besteht die Antwort des Servers nur aus einem Statuscode, es wird nicht eine Liste der Erstellten Dokumente zurück geschickt. Fall die Erstellung fehlschlägt wird kein Dokument angelegt. Es wird versucht mitzuteilen warum die Erstellung fehlschlug.

## 5 Daten Definition

### 5.1 Policy Arguments

Folgende Workitem, bzw. Policy Typen sind momentan vorhanden:

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: TCPPOP Open TCP Listening Ports  
12: TCPBL Blocked TCP Listening Ports  
13: UDPPOP Open UDP Listening Ports  
14: UDPBL Blocked UDP Listening Ports  
15: SWIDT SWID Tag Inventory  
16: TPMRA TPM Remote Attestation

Einige Typen benötigen unterschiedliche Argumente, Gemeinsamkeiten lassen sich wie folgt gruppieren:

**Keine Argumente** 00, 01, 02, 03, 04

**Datei Pfad** 05, 06, 07

**Verzeichnis Pfad** 08, 09, 10

**Port Liste** 11, 12, 13, 14

**SWID Request Flags** 15

**TPM Attestation Flags** 16

Dem entsprechend sollen die Argumente der Workitems übermittelt werden. Folgende Objekte werden je nach Type übermittelt, als Verweis wird `<doc,policy-argument>` verwendet.

**Keine Argumente**

```
{}
```

**Datei Pfad**

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

**Verzeichnis Pfad**

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



**Port Liste**

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

**SWID Request Flags**

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

**TPM Attestation Flags**

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

**5.2 Recommendation Types**

Folgende Recommendation Types bzw. Actions sind vorhanden:

- 0: ALLOW Die Empfehlung/Aktion ist, den Client zuzulassen
- 1: BLOCK Die Empfehlung/Aktion ist, den Client zu blockieren
- 2: ISOLATE Die Empfehlung/Aktion ist, den Client in einem isoliertem Segment zu platzieren
- 3: NONE

Diese Werte werden von der API als Integer Id verwendet. Es wird kein Objekt für die Re-präsentation erstellt.

**5.3 Hash-Set**

Ein Hash-Set wird innerhalb eines File-Documents ausgeliefert, der Verweis lautet `<doc,hash-set>` und ist wie folgt aufgebaut:

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

Falls ein Algorithmus noch nicht existiert wird er erfasst, momentan sind folgende Algorithmen erfasst: SHA384, SHA256, SHA1, SHA1-IMA

## 6 REST Ressourcen

### 6.1 Session Steuerung und Ablauf

#### 6.1.1 Controller

**URI Path** /sessions/start/

**Archetype** Controller

**Methods** POST

**Request Parameter**

**connectionId** strongSwan Connection Id

**clientIdentity** strongSwan Client-Identity

**hardwareId** Die Id, welche das Gerät identifiziert, so zum Beispiel, AIK, Android-Id, DBUS machine-id, o.ä. Dies entspricht dem **value** Feld in der **device** Tabelle in der Datenbank

**productName** Der Productname ist der Name des OS wie er in der **product** Tabelle der Datenbank steht

**JSON Format Response**

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

**Beschreibung** Dieser Controller erstellt und startet eine Session, das Device welches der Session zugeordnet werden soll wird anhand der **hardwareId** und dem **productName** bestimmt. Falls eines der Objekte noch nicht existiert wird dieses durch den Controller erstellt. Die Id, die im Response Dokument zurück geliefert wird, dient zur zukünftigen Identifikation der soeben gestarteten Session. Ausserdem erhält man eine Liste von Workitems die für diese Session abgearbeitet werden müssen.

**URI Path** /sessions/{id}/end/

**Archetype** Controller

**Methods** POST

**Request Parameter**

**recommendation** Endgültiges Resultat/Empfehlung für diese Session.

**Beschreibung** Dieser Controller schliesst die Session ab und startet alle zusätzlich nötigen Vorgänge, so werden zum Beispiel die Workitems dieser Session abgeräumt und die jeweiligen Resultate gespeichert und können via /sessions/{id}/results abgerufen werden.

### 6.1.2 Documents und Collections

**URI Path** /sessions/{id}/

**Archetype** Readonly Document

**Methods** GET

**JSON Format Response**

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

**Beschreibung** Informationen zu einer bestimmten Session. Sessions sollen nicht direkt geändert werden, sondern nur über die entsprechenden Controller, der Grund dafür ist, dass im Hintergrund noch zusätzliche Operationen vorgenommen werden müssen.

**URI Path** /sessions/

**Archetype** Readonly Collection

**Filter Query**

**timeFrom** <int,timestamp>

**timeTo** <int,timestamp>

**Methods** GET

**JSON Format Response**

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

**Beschreibung** Liste aller Sessions. Neue Sessions können nur über den entsprechenden Controller erstellt werden.

**Workitems****URI Path** /session/{id}/workitems/{id}/**Archetype** Readonly Document**Methods** GET**JSON Format Response**

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

**Beschreibung** Ein zu einer bestimmten Session zugehöriges Workitem. Workitems können nicht direkt erstellt werden, sondern werden beim Erstellen einer Session anhand der Enforcements erstellt. Workitems gibt es nur, so lange eine Session nicht beendet wurde. Nach dem Ende der Session wird ein HTTP 404 Statuscode zurückgeliefert.

**URI Path** /session/{id}/workitems/**Archetype** Readonly Collection**Filter Query****type** <int,policy-type>**Methods** GET**JSON Format Response**

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

**Beschreibung** Eine Liste aller Workitems die zu einer bestimmten Session gehören. Workitems können nicht direkt erstellt werden, sondern werden beim Erstellen einer Session anhand der Enforcements erstellt. Workitems gibt es nur, so lange eine Session nicht beendet wurde. Nach dem Ende der Session wird eine leere Liste geliefert.

**URI Path** /session/{id}/workitems/{id}/result/

**Archetype** Document

**Request Parameter**

**recommendation** Resultat/Empfehlung für dieses Workitem.

**comment** Kommentar zum Resultat.

**Methods** GET, POST

**Response Statuscodes**

**201 Created** Resultat wurde erfolgreich gespeichert.

**409 Conflict** Resultat existiert bereits.

**JSON Format Response**

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

**Beschreibung** Auf dieser Ressource soll das Resultat des jeweiligen Workitems eingetragen werden. Diese Resultate werden beim Beenden der Session von den Workitems in die Session-Resultate übertragen.

## Results

**URI Path** /session/{id}/results/{id}/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format Response**

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

**Beschreibung** Einzelnes Resultat. Im Gegensatz zur Resultat-Resource unter der Workitem Resource enthält dieses Document auch einen Link zum zugehörigen Enforcement.

**URI Path** /session/{id}/results/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format Response**

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

**Beschreibung** Nach dem Beenden einer Session können die eingetragenen Resultate in dieser Collection abgefragt werden. Diese Collection ist readonly, damit die Session-Ergebnisse nicht nachträglich geändert werden können.

## 6.2 SWID Erweiterung

### 6.2.1 SWID Tags: Messung

**URI Path** /sessions/{id}/swid-measurement/

**Archetype** Controller

**Methods** POST

**Content-Type** application/json; charset=utf-8

**Request Parameter**

`softwareId` Software-IDs als JSON-Liste.

**Response Statuscodes**

**200 OK** SWID Tags der übermittelten Software-IDs sind eingetragen und wurden für die übermittelte Session eingetragen.

**404 Not Found** Session mit der spezifizierten ID wurde nicht gefunden.

**412 Precondition Failed** Es existieren nicht alle SWID Tags für die übertragenen Software-IDs. Als Payload werden die fehlenden Software-IDs übertragen.

**JSON Format Response**

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

**Beschreibung** Die strongSwan Komponente sendet eine Liste von Software-IDs an die strongTNC Schnittstelle. Die Software-IDs wurden zuvor auf dem Client gemessen und widerspiegeln die momentan installierten Software Pakete.

Wenn bereits SWID Tags für alle übertragenen Software-IDs bestehen, können diese direkt eingetragen werden.

### 6.2.2 SWID Tags: Erstellung

**URI Path** /swid/add-tags/

**Archetype** Controller

**Methods** POST

**Content-Type** application/json; charset=utf-8

**Request Parameter**

`xmlData` SWID Tag als JSON-Liste.

**Response Statuscodes**

**200 OK** SWID Tags wurden erfolgreich verarbeitet.

**400 Bad Request** Fehlerhafter Request. Details zum Fehler werden im Response-Body zurückgesendet.

**Beschreibung** Die übermittelten Tags werden gelesen und in die Datenbank gespeichert. Bereits vorhandene Tags werden übersprungen.

## 6.3 CRUD Ressourcen

### 6.3.1 Products

**URI Path** /products/{id}/

**Archetype** Document

**Methods** GET, PUT, PATCH

**JSON Format Response**

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

**URI Path** /products/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

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

**URI Path** /products/{id}/default-groups/

**Archetype** Collection

**Methods** GET, POST

**Request Parameter**

groupId group-id

**JSON Format Response**

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



### 6.3.2 Packages

**URI Path** /packages/{id}/

**Archetype** Document

**Methods** GET, PUT, PATCH

**JSON Format Response**

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

**URI Path** /packages/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

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

**URI Path** /packages/{id}/versions/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

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

### 6.3.3 Versions

**URI Path** /versions/{id}/

**Archetype** Document

**Methods** GET, PUT, PATCH, DELETE

**JSON Format 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 Format 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 Format Response**

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

**URI Path** /identities/

**Archetype** Collection

**Methods** GET, POST

**JSON Format 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 Format 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 Format 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 Format Response**

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

**URI Path** /device/{id}/sessions/{id}/results/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format Response**

[<doc,result>, ...]

**URI Path** /device/{id}/groups/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

[<doc,group>, ...]

**URI Path** /device/{id}/swid-tags/

**Archetype** Readonly Collection

**Methods** POST

**JSON Format Response**

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

### 6.3.6 Enforcements

**URI Path** /enforcements/{id}/

**Archetype** Readonly Document

**Methods** GET

**JSON Format 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 Format 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 Format 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 Format Response**

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

**URI Path** /groups/{id}/enforcements/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format Response**

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



### 6.3.9 Files

**URI Path** /files/{id}/

**Archetype** Document

**Methods** GET, PUT, PATCH

**JSON Format 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 Format 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 Format Response**

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

**URI Path** /directories/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

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

**URI Path** /directories/{id}/files/

**Archetype** Collection

**Methods** GET, POST

**JSON Format Response**

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

### 6.3.11 SWID Tags

**URI Path** /swid-tags/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format 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 Format 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 Format Response**

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

### 6.3.12 Entities

**URI Path** /swid-entities/{id}/

**Archetype** Readonly Document

**Methods** GET

**JSON Format Response**

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

**URI Path** /swid-entities/

**Archetype** Readonly Collection

**Methods** GET

**JSON Format 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 Format Response**

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

## Literatur

- [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).