ya tu aprendes

MikroTik

CERTIFIED TRAINER

TR-069 Sobrevivir al reset del cliente



WE INSPIRE KNOWLEDGE

¿ Quién soy ?

Soy Jorge Castellet



- Soy Profesor Certificado de MikroTik

MTCNA, MTCIPv6E, MTCRE, MTCTCE, MTCWE

MTCUME, MTCINE

- Soy Consultor

Operador Neutro

Soluciones de conectividad para ISP

j.castellet@yatuaprendes.com


- ¿ Hola ?
- ¿ Hola ?
- No me funciona el interné

El cliente llama al “call center” para quejarse (amablemente) que no tiene acceso a internet.

Tras una serie de pruebas (y mucha paciencia), determinamos que la incidencia viene provocada porque el router del cliente se ha restaurado a valores de fábrica.

Consultamos el parte meteorológico del día y vemos que es un día muy soleado, por lo que no ha podido ser una tormenta eléctrica.

Nos quedamos perplejos al ver que ha sido el cliente el que ha hecho renacer al router.

Es viernes tarde, y ….. tendremos que ir a su casa a recuperar el dispositivo.
(A lo lejos se oye al cliente decir que él no ha tocado nada, de nada)

Esta historia es una ficción y sólo sirve para plantear un problema



ya tu aprendes

Deseo …

- Un sistema con el que puedan comunicar mis dispositivos
- Que sea capaz de identificarlos y de enviarles la configuración a cada uno de ellos
- Saber si están configurados o no.

ya tu aprendes

Lo que necesitas es ...

- CPE Wan Management Protocolo (CWMP)

- Auto Configuration Server (ACS)

CWMP

- El TR-069 (Technical Report 069) es una especificación técnica que define la capa de aplicación para la gestión remota de un CPE conectado a una red IP.

- El standard TR-069 se desarrolló para la gestión y configuración automáticas de los dispositivos por medio del Servidor de Auto Configuración (ACS)

CWMP

- El protocolo está basado en SOAP/HTTP.
- Incluye entre otras cosas:
  - ✤ Autoconfiguración segura.
  - ✤ Funciones para el control de la gestión.
- Entorno integrado.

ya tu aprendes

CWMP

- A cualquier mensaje intercambiado durante la comunicación se le denomina *Sesión.*

- Unicamente los CPE son capaces de iniciar una sesión

- El CPE inicia una comunicación en respuesta a diferentes eventos:

  Arranque por primera vez, reinicio, intervalo`periódico, transferencia completa, etc

- El ACS puede solicitar al cliente que inicie una Sesión.

- Durante la sesión, cada una de las partes, llama a RPC's que se ejecutarán en el otro lado.

- El CPE siempre inicia una Sesión con un RPC de "*Inform*", que contiene la causa de la conexión, información sobre el dispositivo y algunos parámetros extra.

ya tu aprendes

CWMP

- Los parámetros son parejas de “*nombre = valor*”.
- Cada fabricante decide que parámetros soporta en su dispositivo.
- Al conjunto de parámetros soportados se le denomina modelo de datos (Data Model).
- Existen 3 modelos de datos predefinidos

  TR-098, TR-181:1 y TR181:2

- El fabricante debe basar su lista de parámetros soportados en los modelos de datos predefinidos.

ya tu aprendes

¿ Y la seguridad ?

- TR-069 nos proporciona varios mecanismos de seguridad:
- Autenticación
- Usuario / Contraseña
- Certificados SSL para los clientes
- Comunicación
- SSL
- Y, por supuesto, podemos implementar seguridad en los dispositivos (como la lista de direcciones del firewall)

ya tu aprendes

Mikrotik TR-069

- La implementación de RouterOS soporta tanto el protocolo HTTP como HTTPS en la dirección URL del ACS.

- Autenticación HTTP de usuario y contraseña para “logearse” al ACS

- Inform periódicos

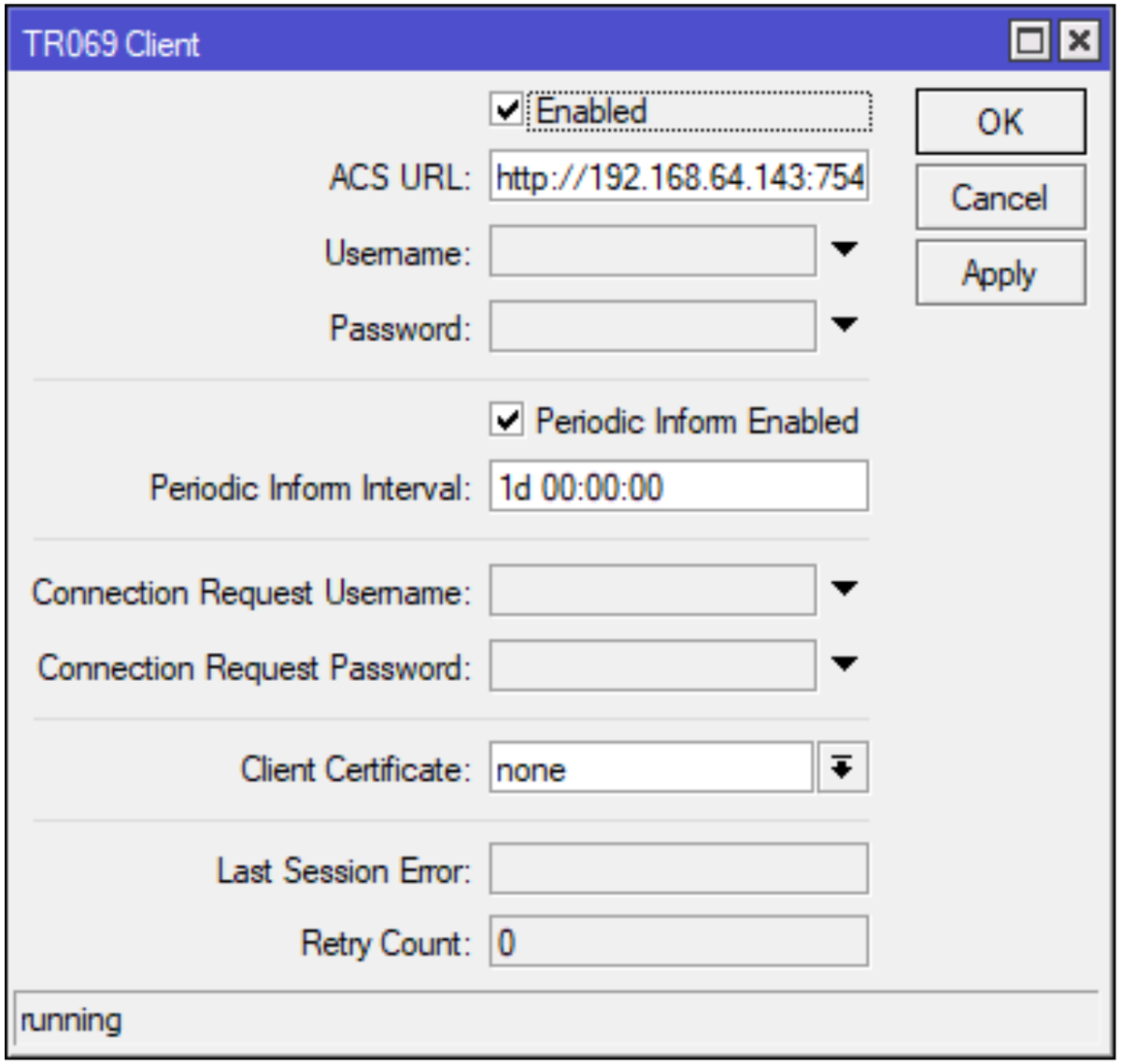
- Certificados de cliente (para una mayor seguridad)

- El modelo de datos de RouterOS está basado en el **TR-181 Issue2 Amendment 11**

ya tu aprendes

Mikrotik TR-069

Configuración mínima



ya tu aprendes

Mikrotik TR-069

- Por desgracia, después de un reset, la configuración del TR-069 se pierde (con nuestra ilusión)

- Necesitamos apoyarnos en netinstall para que podamos llegar a buen puerto y, sobrevivamos al cliente ;)

- Nuestra script por defecto únicamente necesita;

1.Instalar el certificado del ACS (en caso de que utilicemos protocolo HTTPS)

2.Asignar una dirección IP a un interfaz del dispositivo

3.Agregar las rutas necesarias en la tabla de enrutamiento

4.Configurar el cliente TR-069



Disponemos de una script de ejemplo en la wiki:
http://wiki.mikrotik.com/wiki/Tr069-best-practices

ya tu aprendes

ACS

- Mikrotik es compatible con diferentes ACS, por ejemplo:

  ✓ AVSystems

  ✓ Axiros

  ✓ Friendly Tech

  ✓ GenieACS (open source)

ya tu aprendes

GenieACS

- Es un Sistema de Autoconfiguración Rápido y Ligero.

- GenieACS es una solución open source para la gestión remota TR-069 con capacidad avanzadas de aprovisionamiento de dispositivos

- Está construido sobre Node.js y utiliza MongoDB como base de datos

ya tu aprendes

Instalación

Necesitaremos:

- Node.js: de la versión 6.x y la 8.x

  (Se recomienda la versión 8.x)

- MongoDB: de la versión 2.6 hasta la 3.4

- Las herramientas de compilación (build tools) y la librería libxml2

  (Las instalaremos desde el apt-get)



Para instalar Node.js en una distribución Debian/Ubuntu necesitaremos descargar la script pertinente. En este caso https://deb.nodesource.org/distributions/deb/setup_8.x

ya tu aprendes

“Una vez tenemos hecho el caldo, pasamos a la sopa”

(proverbio chino)

Instalación

- Realizaremos la instalación a través del npm:

```
npm install -g genieacs
```

  Si el anterior proceso de instalación nos diera problema, podemos limpiar el directorio e instalarlo desde repositorio del git para ello ejecutaremos los comandos:

```
git clone https://github.com/zaidka/genieacs.git
cd genieacs
git checkout $(git tag -l v1.1.* --sort=-v:refname | head -n 1)
npm install
npm run compile
```


ya tu aprendes

Instalación

- Una vez instalado, tenemos 3 ejecutables:

- **Genieacs-cwmp**

  Es el servicio por el que se comunica el CPE. Por defecto escucha en el puerto 7547 TCP.
  (Si cambiamos el puerto, hemos de cambiarlo en la configuración del mikrotik en ACS URL)

- **Genieacs-nbi**

  Este servicio exporta una REST API para el frontend GUI. Por defecto escucha en el puerto 7557 TCP

- **Genieacs-fs**

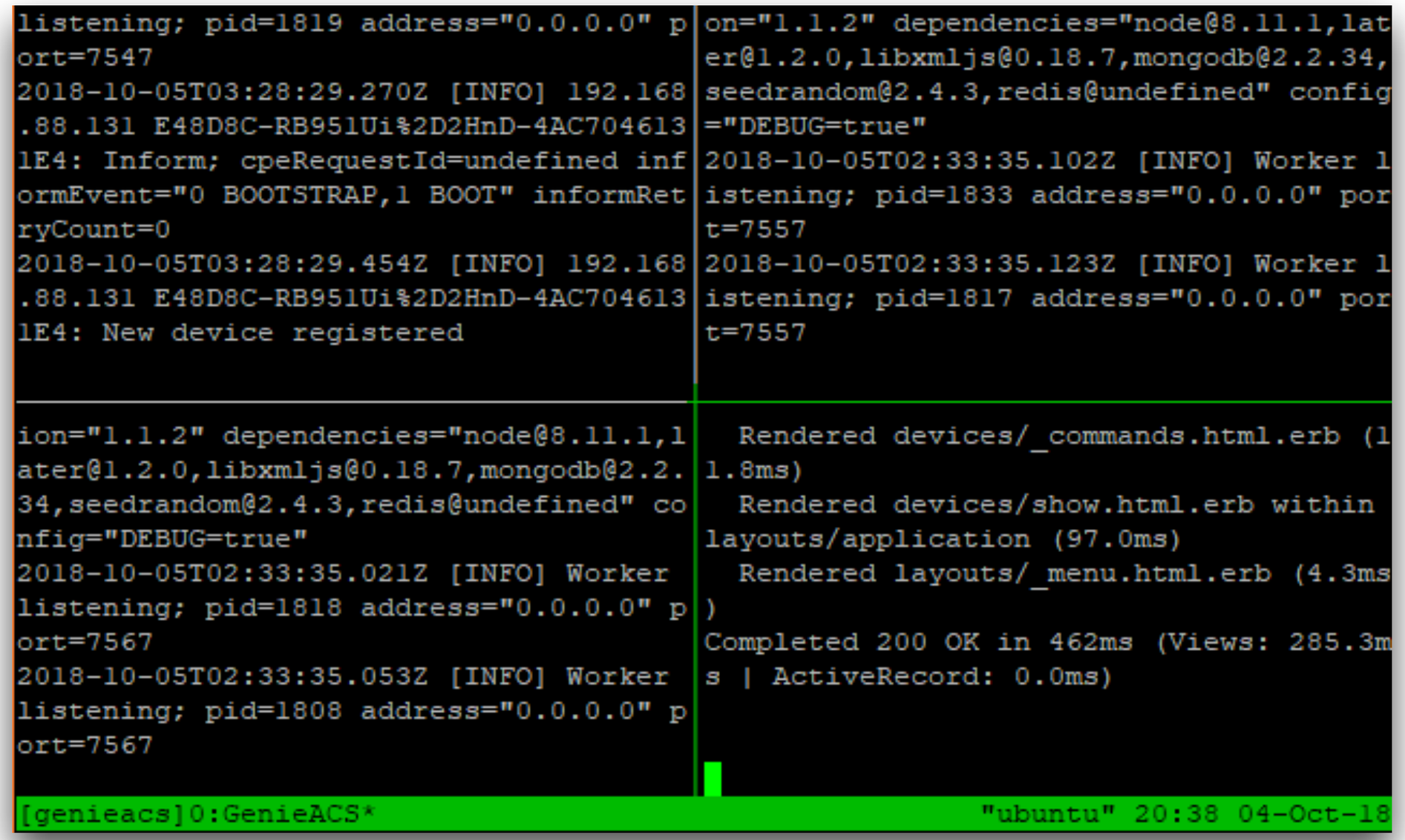
  Este servicio es el servidor de ficheros desde el que nuestro CPE se descargara las imágenes del firmware.



Para la modificar los valores por defecto y/o configurar el nombre de servidor de ficheros, editaremos el siguiente archivo del genieacs:
config/config.json

ya tu aprendes

Instalación



Ejecución de genieACS mediante el uso de tmux

“Sino quieres caldo, dos tazas”

Todavía nos falta instalar el frontend web



Instalación

- Hasta ahora hemos instalado el genieACS y hemos configurado nuestro CPE mikrotik que se ha conectado al ACS y ha hecho cositas, pero no podemos verlo de una forma bonita. Unicamente tenemos la salida por pantalla de la ejecución del genieACS y el log de nuestro mikrotik.

- Nos falta algo …. Un frontend con el que podamos ver las cosas bien.

Sep/25/2017 21:14:07	memory	tr069, warning	tr069 running in non-secure mode (HTTP)
Sep/25/2017 21:14:07	memory	tr069, debug	starting session, events: [0 BOOTSTRAP, 1 BOOT,]
Sep/25/2017 21:14:07	memory	system, info	tr069-client settings changed by admin
Sep/25/2017 21:14:07	memory	tr069, debug	send: Inform
Sep/25/2017 21:14:07	memory	tr069, debug	rcvd: InformResponse
Sep/25/2017 21:14:07	memory	tr069, debug	send: ""
Sep/25/2017 21:14:07	memory	tr069, debug	session finished ok
Sep/25/2017 21:14:07	memory	tr069, debug	scheduled next Periodic Inform after 86400 seconds

ya tu aprendes

Instalación

Para instalar el frontend (genieacs-gui) necesitamos:

- Ruby on Rails
  (Se recomienda una versión igual o superior a la 2.2.2)

- Bundler

ya tu aprendes

Instalación

- Clonaremos el repositorio del git:

```
git clone https://github.com/zaidka/genieacs-gui.git
```


ya tu aprendes

Instalación

Una vez clonado ejecutaremos:

```
cd genieacs-gui

cp config/graphs-sample.json.erb config/graphs.json.erb

cp config/index_parameters-sample.yml config/index_parameters.yml

cp config/summary_parameters-sample.yml config/summary_parameters.yml

cp config/parameters_edit-sample.yml config/parameters_edit.yml

cp config/parameter_renderers-sample.yml config/parameter_renderers.yml

cp config/roles-sample.yml config/roles.yml

cp config/users-sample.yml config/users.yml

bundle
```


ya tu aprendes

Y .... Ya está



ya tu aprendes

Hemos de manipular el genie.
Lo vemos mejor en una demo, ¿verdad?



Solución

Esto sucede porque las claves que espera el genieacs-gui no son las que envía Mikrotik o cualquier otro dispositivo basado en TR-181.

Para corregirlo, hemos de modificar los archivos:

- genieacs-gui/config/index_parameters.yml
- genieacs-gui/config/summary_parameters.yml

ya tu aprendes

Solución

- Al estar basado en TR-098 el genieacs-gui espera que la información esté en:

  `InternetGatewayDevice.DeviceInfo.*`

- Pero nuestro mikrotik (basado en TR-181) nos da la información en:

  `Device.DeviceInfo.*`

- Pero eso no es todo. A partir de la versión 1.1.x , genieacs exporta un objeto propio con la información “esencial” del cliente. Este objeto contiene la información:

  ✓ **SerialNumber.** Número de serie del dispositivo. En mikrotik es la Mac del interfaz

  ✓ **ProductClass.** Familia del producto. En mikrotik es el modelo de routerboard

  ✓ **OUI**. Identificador del fabricante.Es un identificador único por cada fabricante.

  ✓ **Manufacturer**. Nombre del fabricante.

ya tu aprendes

Solución

```
Serial number: _deviceId._SerialNumber
Product class: _deviceId._ProductClass
Software version: Device.DeviceInfo.SoftwareVersion
MAC: InternetGatewayDevice.WANDevice.1.WANConnectionDevice.1.WANIPConnection.1.MACAddress
IP: InternetGatewayDevice.WANDevice.1.WANConnectionDevice.1.WANIPConnection.1.ExternalIPAddress
WLAN SSID: Device.WiFi.SSID.1.SSID
```

index_parameters.yml

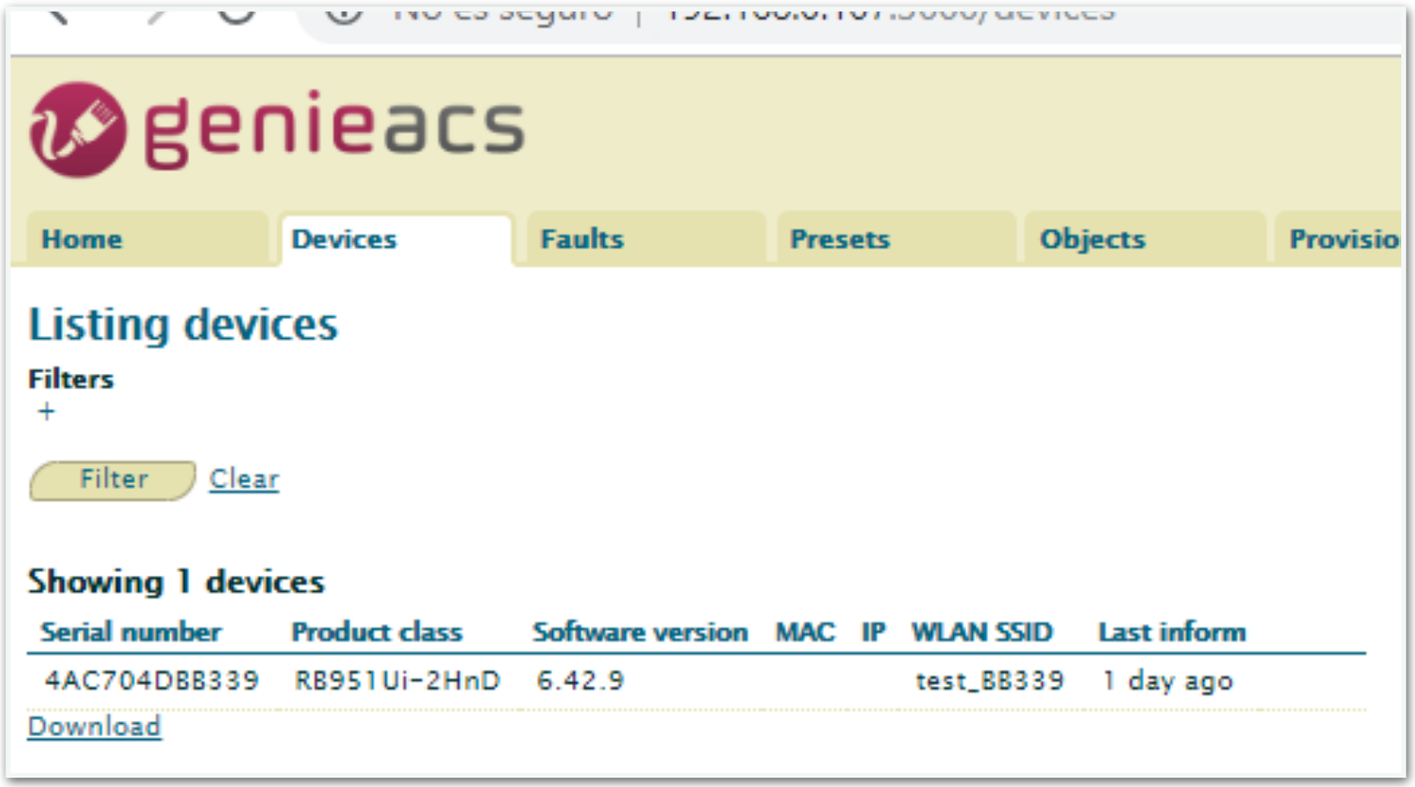
```
Serial number: _deviceId._SerialNumber
Product class: _deviceId._ProductClass
OUI: _deviceId._OUI
Manufacturer: _deviceId._Manufacturer
Hardware version: Device.DeviceInfo.HardwareVersion
Software version: Device.DeviceInfo.SoftwareVersion
MAC: InternetGatewayDevice.WANDevice.1.WANConnectionDevice.1.WANIPConnection.1.MACAddress
IP: InternetGatewayDevice.WANDevice.1.WANConnectionDevice.1.WANIPConnection.1.ExternalIPAddress
WLAN SSID: Device.WiFi.SSID.1.SSID
WLAN passphrase: InternetGatewayDevice.LANDevice.1.WLANConfiguration.1.KeyPassphrase
Hosts:
    _object: Device.Hosts.Host
    Host name: HostName
    IP: IPAddress
    MAC: MACAddress
```

summary_parameters.yml

ya tu aprendes

Solución

¡ Ahora si que salen las cosas !





Pero, Nuestro mikrotik todavía no se autoconfigura

ya tu aprendes

Autoprovision

- El genieACS es un software capaz de “hablar” con cualquier dispositivo. Aunque esté orientado a los dispositivos que usan el TR-098, se comunica perfectamente con nuestro mikrotik.

- Ya hemos hecho una modificación en la configuración del frontend para que nos muestre la información de nuestro mikrotik Ahora toca el genieACS

Presets

- Los presets son, como su nombre indica, configuraciones que aplicaremos al dispositivo.

- Es decir en un preset podemos asignar, por ejemplo, el valor de SSID, la frecuencia y la contraseña de la wifi que tendrá nuestro dispositivo.

Presets

- Los presets se componen de dos partes:
- **Precondiciones**: en esta parte podemos controlar cuando se ejecuta el preset. Pueden ser, por ejemplo:
  - ➡ OUI
  - ➡ Tag
  - ➡ Numero de serie
- **Configuraciones**: en esta parte aplicaremos las configuraciones. Podemos, por ejemplo:
  - ➡ Asignar valores a un parámetro
  - ➡ Agregar o quitar Tags
  - ➡ Refrescar un parámetro
  - ➡ Ejecutar una provisisión

ya tu aprendes

Preset

Adicionalmente podemos especificar:

- **Una planificación**: podemos definir la periodicidad con la que se ejecutará el preset.

- **Los eventos**: podemos definir que eventos del CPE dispararán el preset. Podemos especificar uno o una serie de ellos. Los eventos pueden ser:

  ✓ 0 BOOT

  ✓ 1 BOOTSTRAP

  ✓ 7 TRANSFER COMPLETE

  ✓ M DONWLOAD

  ✓ 2 PERIODIC

ya tu aprendes

Preset

- Los Tags, no son un parámetro del CPE, es un parámetro interno que nos ofrece el genieACS. Mediante el uso de Tags podemos definir un flujo de preset o identificar el estado del CPE.

- Un dispositivo puede contener más de un Tag.

- En nuestra demostración hemos definido los tags:

  - **PENDING**: este tag se asocia a los CPE que todavía no han iniciado el proceso.

  - **UPGRADING**: este tag se asocia al CPE para que se compruebe si necesita una actualización de firmware.

  - **UPGRADED**: este tag se asocia al CPE cuando ya está actualizado.

  - **PROVISIONING**: este tag se asocia al CPE cuando inicia el proceso de provisión de la configuración. Adicionalmente en este estado se utilizan los tags: INIT, WIFI y OTHER

  - **PROVISIONED**: este tag se asocia al CPE cuando está totalmente configurado.

ya tu aprendes

Preset

- A partir de los Tags anteriores definimos los presets:

Name	Channel	Weight	Events
00 NEW DEVICE		0	1 BOOT, -0 BOOTSTRAP
19 UPGRADED	upgrade	0	7 TRANSFER COMPLETE, M Download
20 PROVISIONING	provision	0	2 PERIODIC
21 PROVSIONING - WIFI	provision	0	
28 PROVISIONING - OTHER	provision	0	
10 UPGRADING	upgrade	0	
29 PROVISIONED	provision	0	

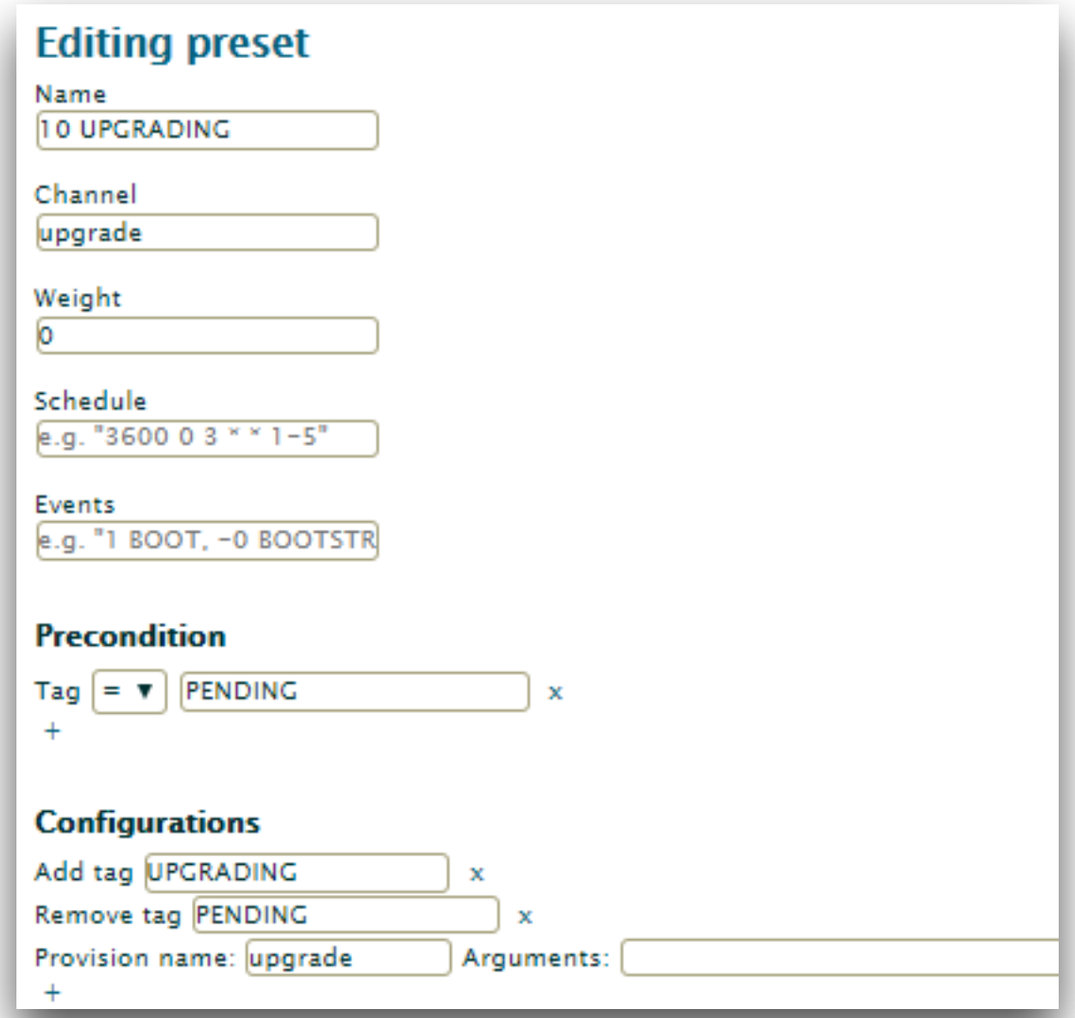


El peso (Weight) sirve para que en caso de que dos presets contengan el mismo elemento a configurar, prevalece la configuración del que tiene mayor peso

ya tu aprendes

Preset

- El preset de 10 UPGRADING sería:



ya tu aprendes

Provisión

- En el preset anterior habéis visto que en Configuration está la opción de “Provision”.

- ¿ Qué es una provisión ?

Una provisión es un programa que ejecutamos para realizar configuraciones en base a condiciones complejas.



Provision

- Las Provisiones se introducen en la versión 1.1
- Son programas de javascript que se ejecutan en un sandbox.
- Ofrece una serie de funciones definidas por genieACS
- Podemos pasar parámetros que estarán disponibles a a través del array *args*

ya tu aprendes

Provision

Hemos definido dos provisiones:

- **Upgrade**: se comprueba la versión del routerOS y si es menor de la 4.20.9 actualiza el mikrotik.

- **Wifi**: establece el ssid a la cadena test_ + los últimos cinco dígitos de la MAC y configura el modo ap_bridge

ya tu aprendes

Provision

- La script de upgrade sería:

```
let version=declare("Device.DeviceInfo.SoftwareVersion",{value:1}).value[0];
log('device version: ' + version);
if (version=="6.42.9"){
  log('No upgrade needed');
  declare("Tags.UPGRADED",null,{value:true});
  declare("Tags.UPGRADING",null,{value:false});
} else {
  log('upgrading firmware');
  declare("Downloads.[FileType:1 Firmware Upgrade Image]",
        {path: 1}, {path: 1});
  declare("Downloads.[FileType:1 Firmware Upgrade Image].FileName",
        {value: 1}, {value: "upgrade-mipsbe-6.42.9.xml"});
  declare("Downloads.[FileType:1 Firmware Upgrade Image].Download",
        {value: 1}, {value: Date.now()});
}
```


ya tu aprendes

Files

- En la script hemos definido un Download especificando el tipo de archivo a “1 Firmware Upgrade Image”

- El archivo que descarga nuestro mikrotik es un archivo xml.

- Mikrotik especifica que para actualizar la versión del routerOS desde el ACS hay que enviar un archivo xml que contiene los enlaces a los diferentes npk que vamos a actualizar.

ya tu aprendes

Files

Contenido del archivo xml:

```
<upgrade version="1" type="links">
    <config/>
    <links>
        <link>
            <url>http://192.168.0.107:7567/routeros-mipsbe-6.42.9.npk</url>
        </link>
        <link>
            <url>http://192.168.0.107:7567/tr069-client-6.42.9-mipsbe.npk</url>
        </link>
    </links>
</upgrade>
```

Fijarse que también hemos agregado el **paquete del tr069**. Ya que sino lo hacemos, tras actualizar la versión, ya no estará disponible el servicio y nuestro mikrotik dejará de comunicar con el ACS.

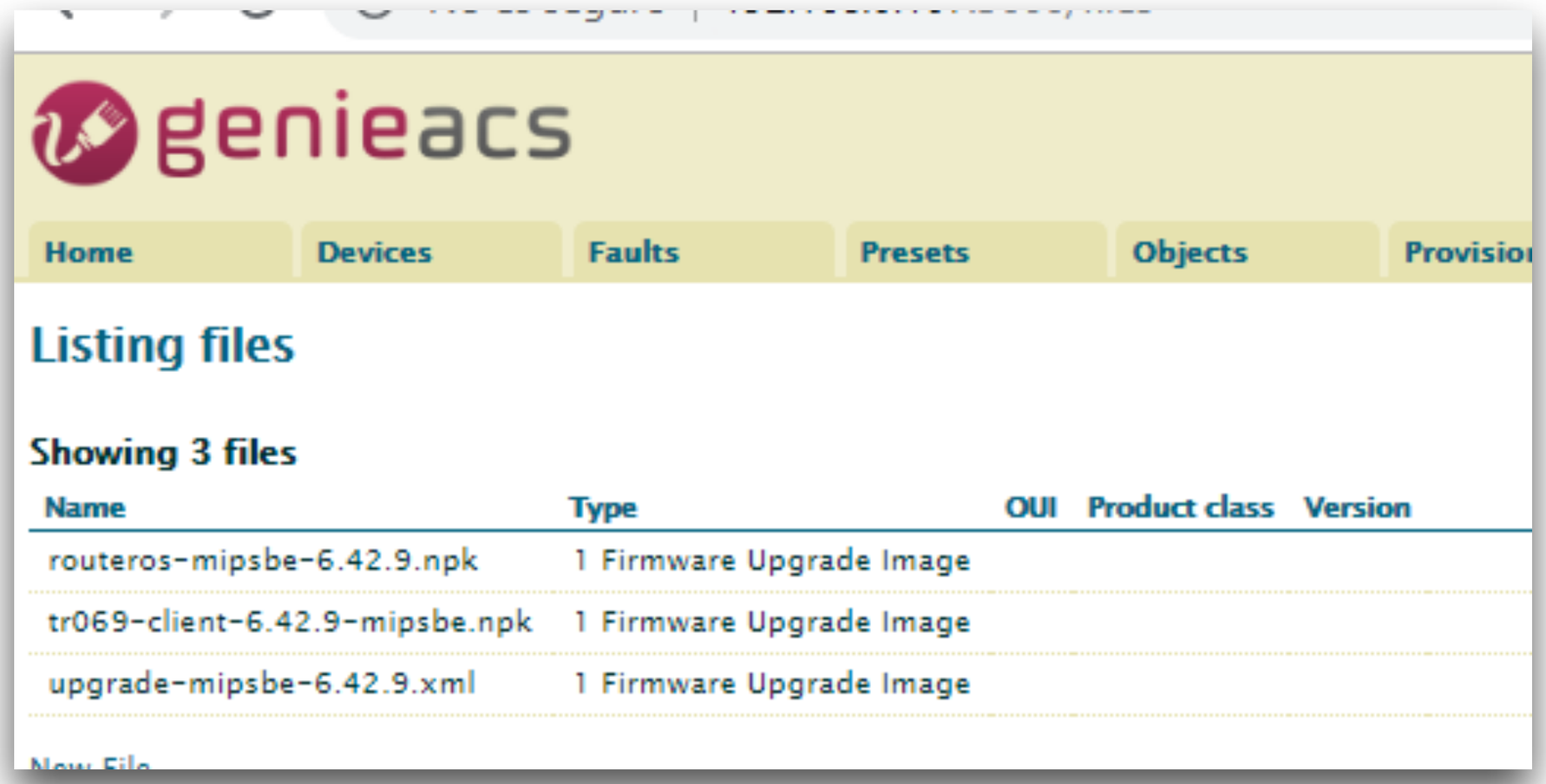
Si hacemos servir otros paquetes adicionales, también hay que agregarlos.



ya tu aprendes

Files

Así pues, en nuestro sistema tendremos los siguientes archivos:



ya tu aprendes

Files



He hecho todo lo que me has enseñado y no me actualiza.

Me da un error de “unresolved no se que”.

¿ Qué hago ?

ya tu aprendes

Files

Ay ! Alma de cántaro!

- No has configurado bien el nombre del servidor de ficheros en el archivo de configuración **config.json** de genieacs.

- La entrada del archivo en cuestión es:

  FS_HOSTNAME

- En ella especificaremos la ip o el nombre del servidor donde se está ejecutando el genieACS, mas concretamente, el proceso genieacs-fs.

Hay que hacer notar, que esto es para descargar el archivo xml. Los enlaces que hay dentro del archivo xml pueden hacer referencia a cualquier servidor (incluso el de downloads de mikrotik ;)



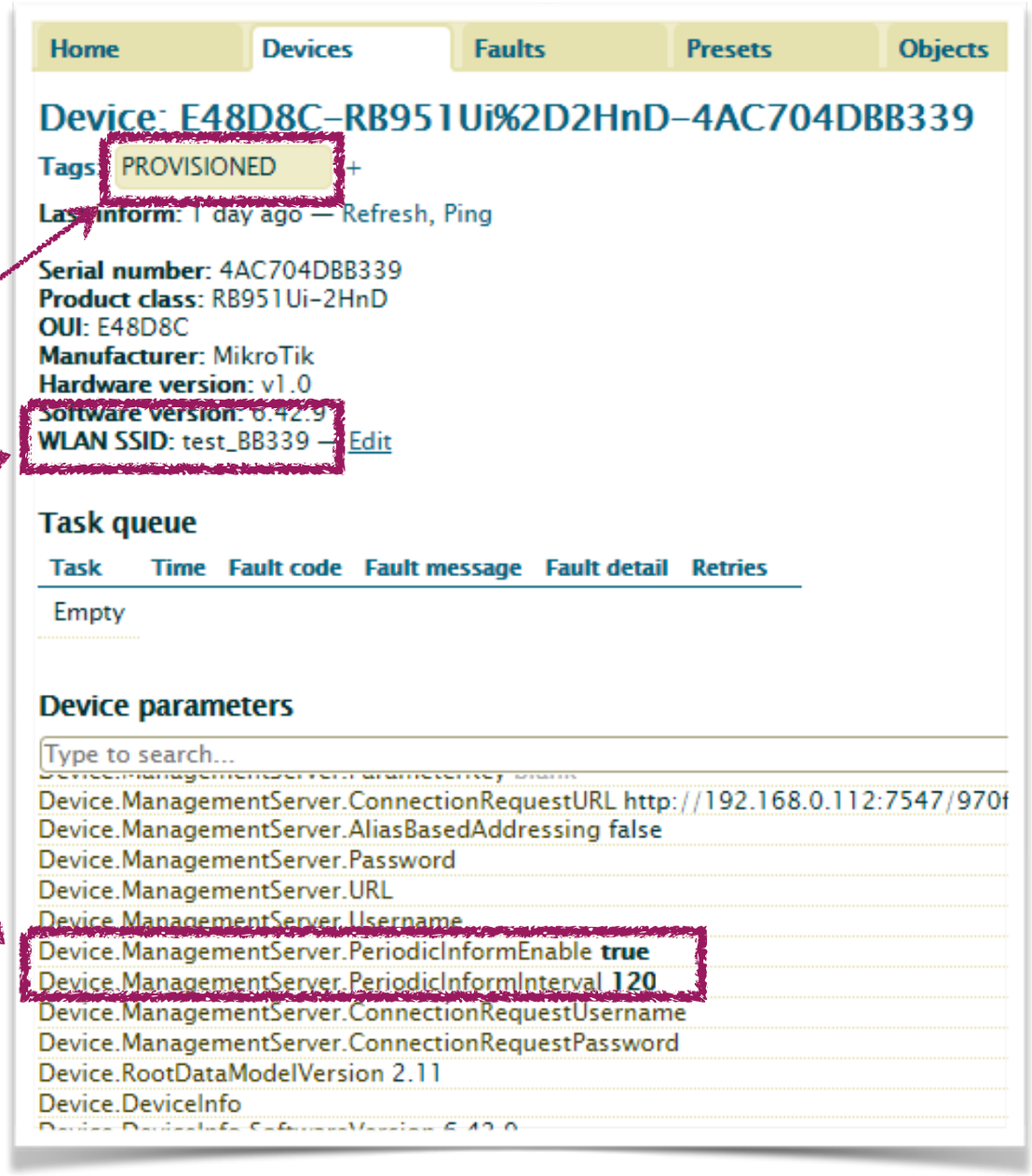
ya tu aprendes

Resultado

Una vez configurado el genieACS, apagamos y encendemos nuestro CPE y ….

¡¡ Eureka !!

¡ Ha funcionado !



ya tu aprendes

Si queréis obtener una copia de la máquina virtual que se ha utilizado para la presentación,

por favor, enviamos un correo a:

j.castellet@yatuaprendes.com

Y te enviaremos en enlace de descarga.

Gracias por vuestra atención.



Mas información

- https://www.broadband-forum.org/technical/download/TR-069_Amendment-5.pdf
- https://wiki.mikrotik.com/wiki/Manual:TR069-client
- https://wiki.mikrotik.com/wiki/Tr069-best-practices
- https://wiki.mikrotik.com/tr069ref/current.html
- Hannes Willemse presentation at ZA17
- https://mum.mikrotik.com/presentations/ZA17/presentation_4990_1512109593.pdf
- https://genieacs.com

ya tu aprendes