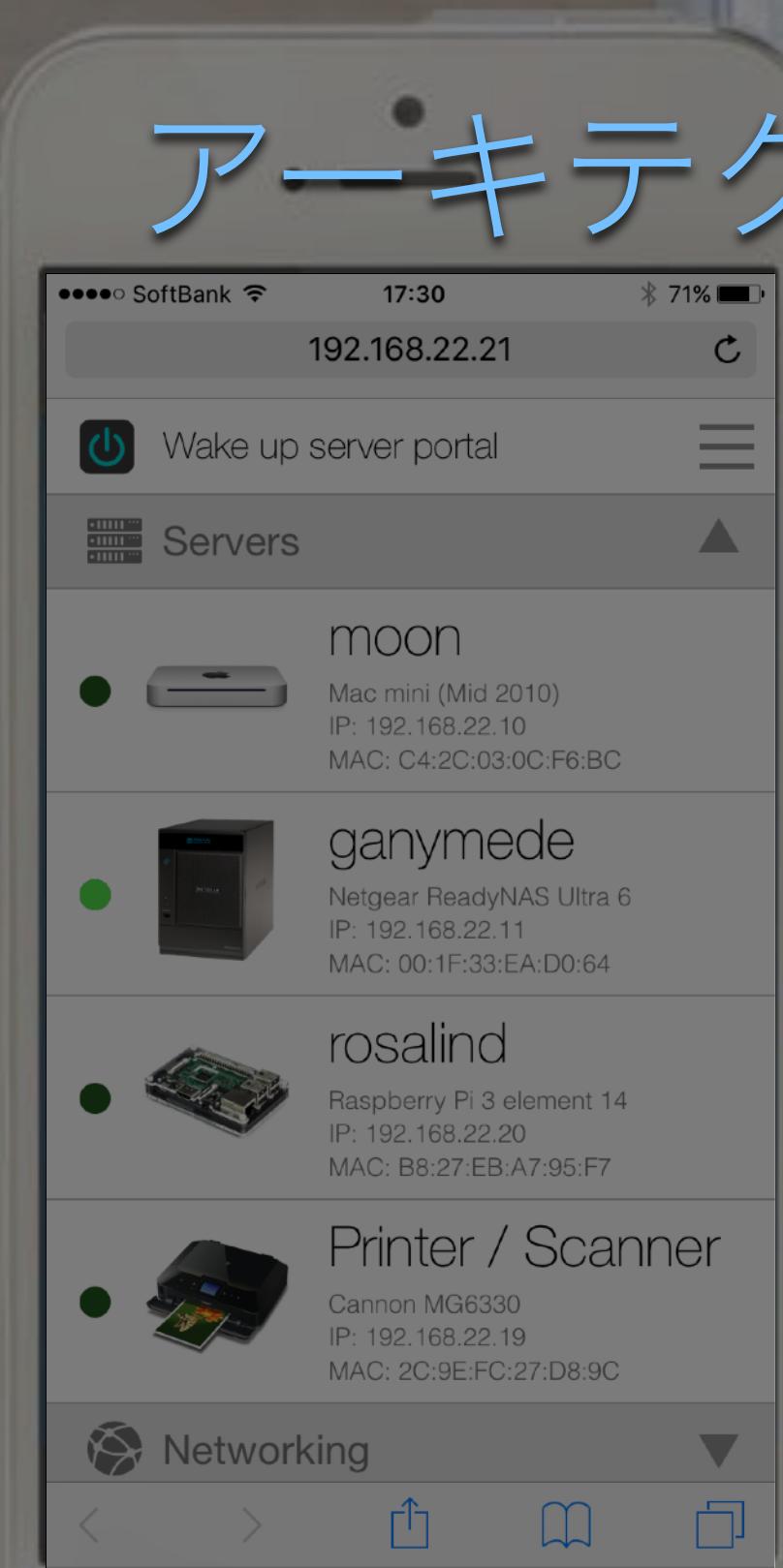


wakeserver

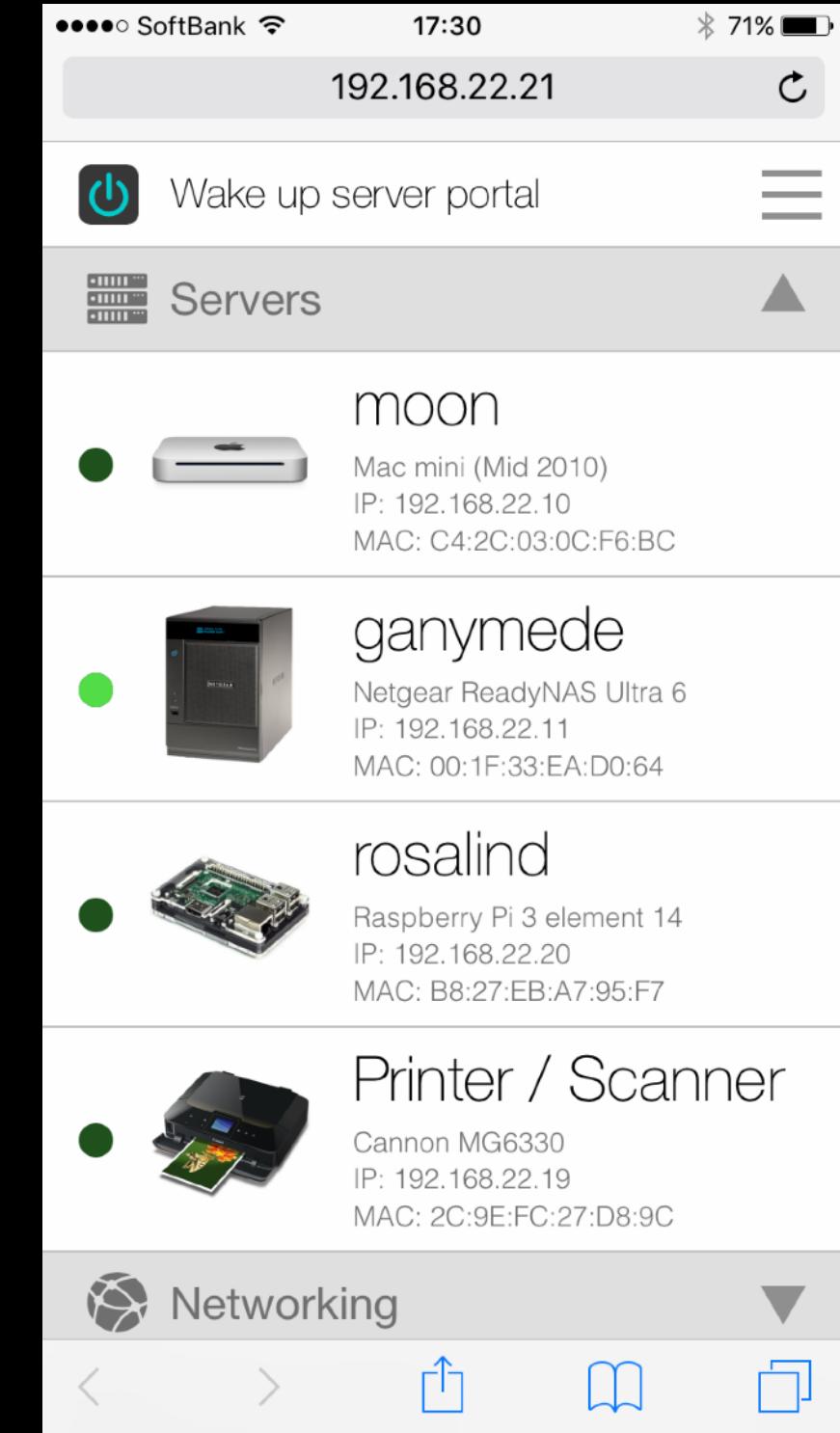
アーキテクチャ & Plugin仕様



2016-12-30
opiopan@gmail.com

wakeserverとは

- ネットワーク家電の制御を行うホームオートメーションハブ
- Raspberry Pi上での動作を想定して作成
- UIはWebアプリケーションとして提供
- Apple Home Kit 対応ブリッジとしても動作



<https://github.com/opiopan/wakeserver>

インストール手順

1. ソースダウンロード

```
$ git clone https://github.com/opiopan/wakeserver.git
```

2. パーソナライズ

本資料を参考に wakeservr/personal/ユーザ名/ 配下に設定ファイル
プラグイン等を配置する。

3. ビルド&インストール

```
$ sudo make PERSONAL=ユーザ名 install
```

使用方法

- Web UIで利用する

Webブラウザでポート8080に接続する

http://インストール先サーバのIPアドレス:8080/

- HomeKit連携で利用する

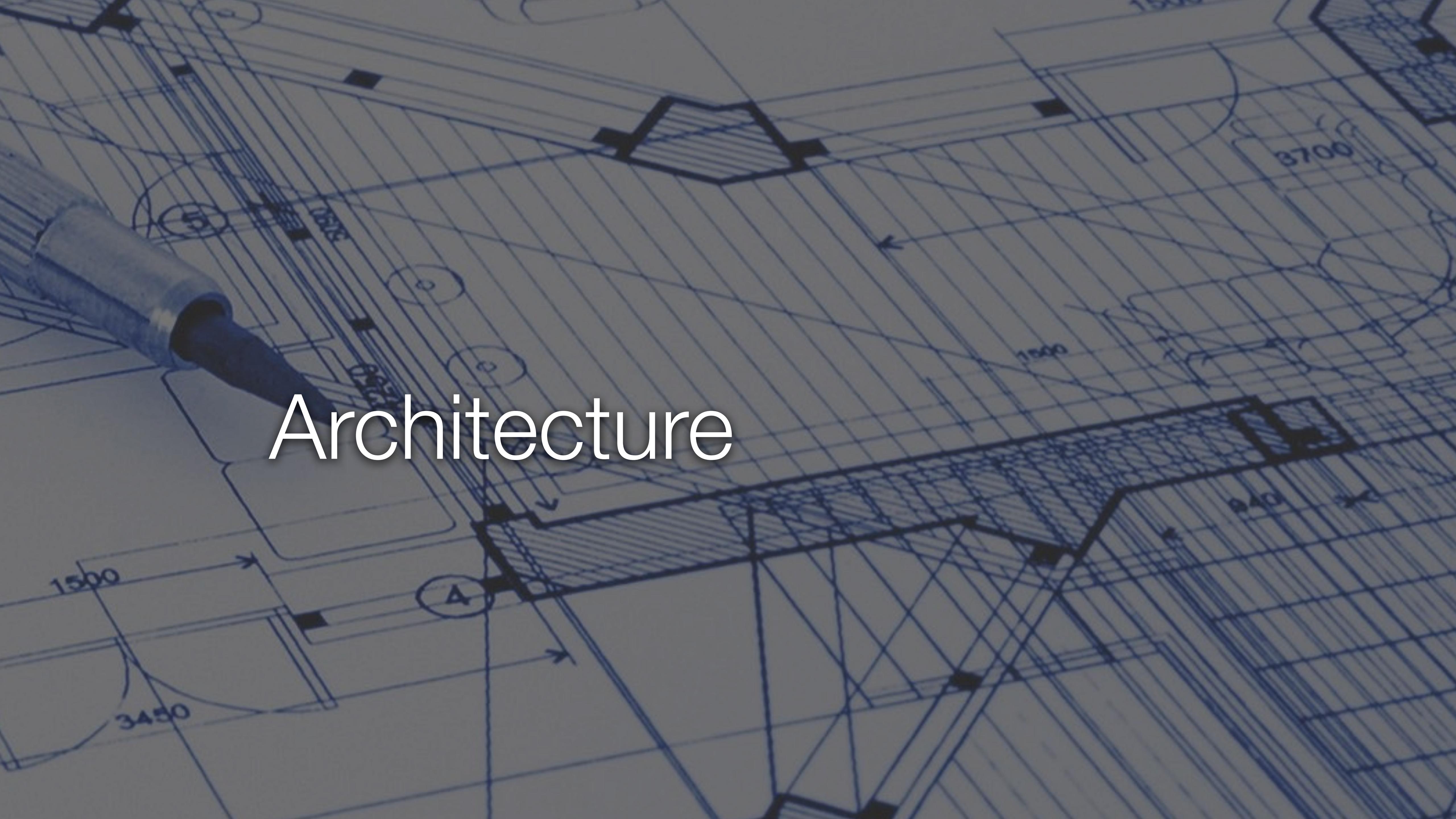
1) iPhone、iPadのホームアプリを起動する

2) アクセサリ追加ボタンを押す

3) リストからWakeserverを選択し追加する

(PINはデフォルトでは123-45-678)

Architecture



wakeserver 全体構成 : ■ + ■ = wakeserver

- wakeserverを構成するプログラム
- 設定ファイル
- 外部プログラム

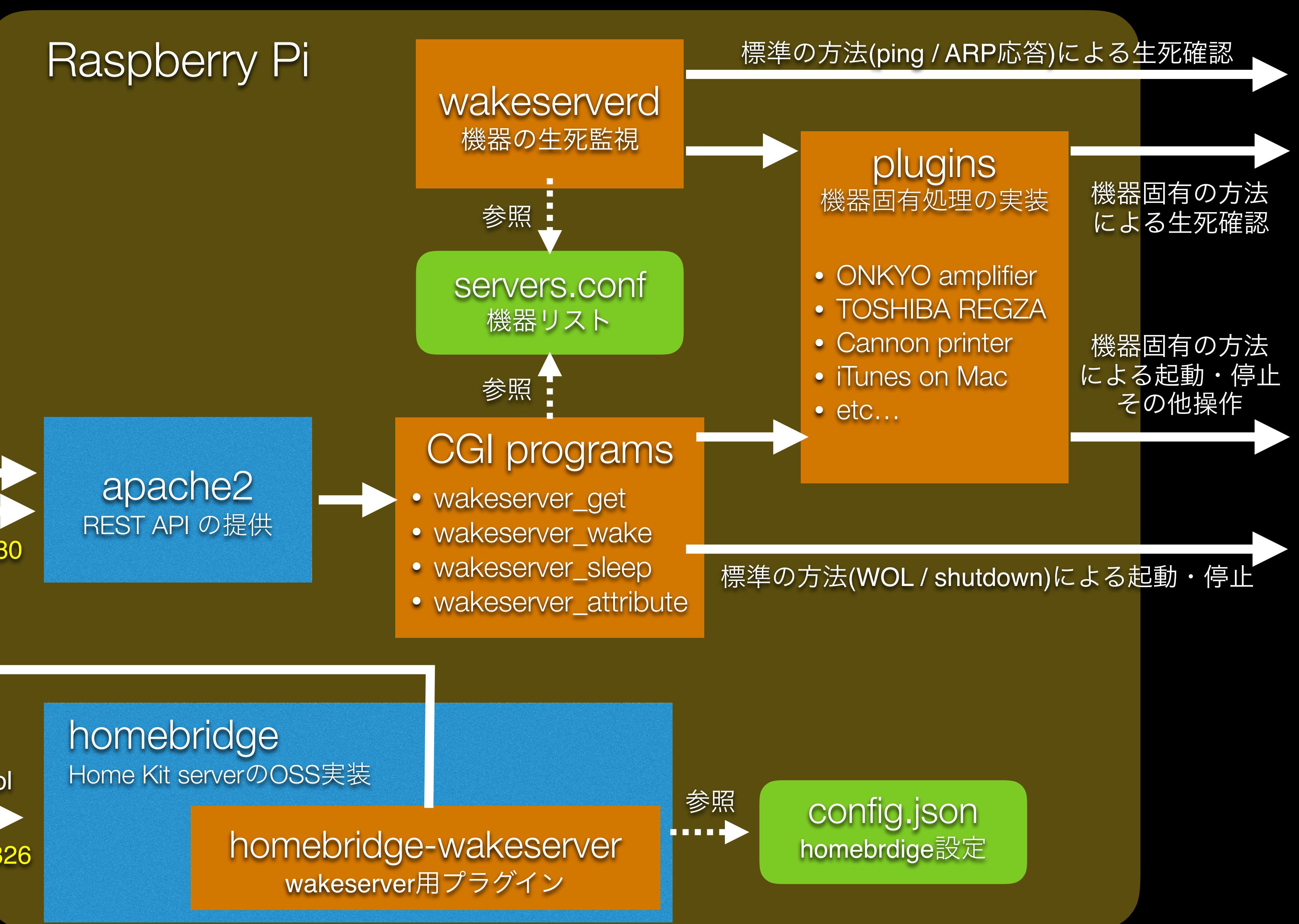


iOS devices / Apple TV



Home Kit server protocol

tcp/51826



pstreeで見るデーモン構成



ソースファイル

カテゴリ	ファイル	ソースディレクトリ	インストール先
client programs	index.html style.css index.js	html/	/var/www/wakeserver/html/
wakeserverd	wakeserverd	daemon/	/var/www/wakeserver/daemon/
CGI programs	wakeserver_get wakeserver_wake wakeserver_sleep wakeserver_attribute	cgi-bin/	/usr/lib/cgi-bin/
Plugins	多数	personal/ユーザ名/plugin/	/var/www/wakeserver/plugin/
機器リスト	servers.conf	personal/ユーザ名/	/var/www/wakeserver/
homebridge plugin	index.js utils/webwether.js	homebridge/homebridge-wakeserver/	/usr/lib/node_modules/homebridge-wakeserver/
homebridge設定ファイル	config.json	personal/ユーザ名/homebridge	/var/homebridge

実装言語：いろいろ使用

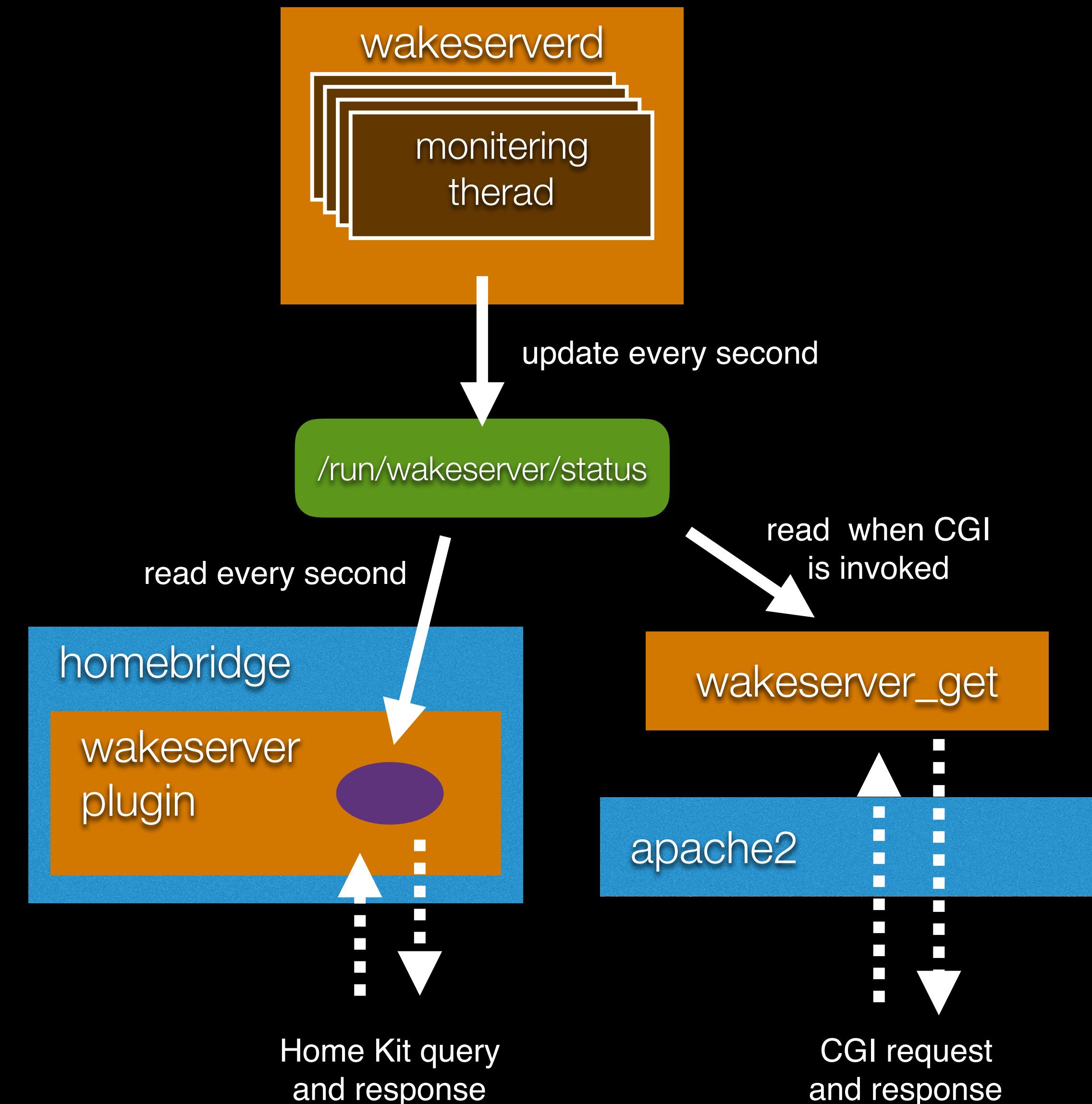
カテゴリ	言語	理由
client programs	HTML5 + CSS3 JavaScript	Webアプリケーションなのでこれ以外選択の余地なし
wakeserverd	Python	デーモンを実装できる能力を持つ十分な能力と軽量かつJSONの取り回しができること、という条件で選択
CGI programs	Python shell script	HTTP request / responseの取り回しのしやすさを基準にPythonを選択（今更PHPでもなし） wakeserver_getだけは処理が単純なのでshell scriptで実装
Plugins	shell script Python	各機器を制御するのに適しているかという基準で選択 多くの場合はshell scriptで十分、ONKYOのアンプはバイナリプロトコルの実装が必要だったのでPythonを使用
homebridge plugin	node.js	Home Kit server protocolのOSS実装を探したところnode.js (サーバサイドJavaScript) のライブラリを見つけたのでnode.jsを採用 どうせnode.jsをインストールするのであれば、サーバ側は全てnode.jsで統一すればよかったと今になって思う…

REST API

URL (相対)	protocol	パラメタ	説明
cgi-bin/wakeserver-get.cgi	POST	type=full simple	機器の状態リストを取得する。 type=fullが指定した場合はservers.conf様の機器の各種属性も合わせたリストがJSONで返り、 type=simpleを指定した場合は状態だけを含むオブジェクトの配列がJSONで返る。
cgi-bin/wakeserver-wake.cgi cgi-bin/wakeserver-sleep.cgi cgi-bin/wakeserver-reboot.cgi	POST	target=機器名	機器の起動、機器の停止、機器のリブートを指示する。 パラメタtargetでservers.confに記載した機器の名前(name属性) を指定する。 つまり、 servers.confに指定する各機器の名前は一意でなければならない。
cgi-bin/wakeserver-attribute.cgi	POST	target=機器名	機器の属性の取得または設定する。 パラメタvalueを省略した場合は属性を変更せず取得動作となる。 現在の定義済み属性はvolume (音量)、 tvchannel (TVチャンネル)、 tvband (TVチャンネルのバンド種別) 。
		attribute=属性名	利用者が自由に属性を定義しHome Kitアクセサリとしてコントロールすることも可能である。 筆者はairplayという属性を使用してMac上のiTunesの再生・停止とAirplay接続先をSiriからコントロールしている。
		value=値	

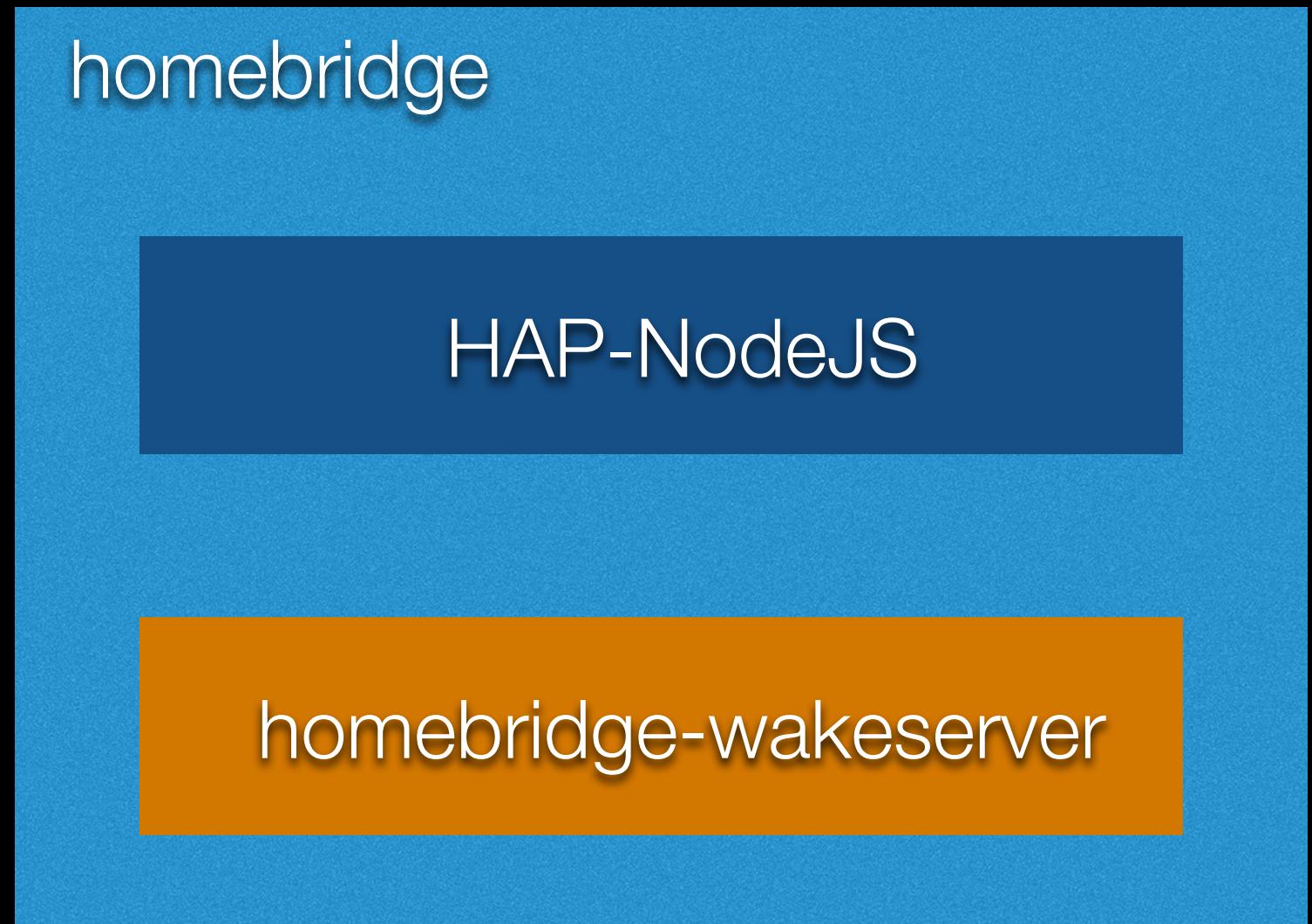
wakeserverdによる生死監視

- 各機器が一定時間内の状態確認を保証できるだけの数の監視スレッドを生成
 - On/Offできる機器は3秒
 - それ以外は7秒
- wakeserverdはRAMファイルシステム(tmpfs) 上に配置されたJSON形式の各機器の状態配列ファイルを1秒毎に更新
- CGI (wakeserver_get) リクエスト毎にこのファイルを依頼元に返却
- homebridge pluginはこのファイルを1秒毎にメモリに展開、iOSデバイスやApple TVから要求があった場合は、メモリに展開された状態を返却



Apple Home Kit連携の仕組み

- homebridgeというnode.jsで動作するHome Kit のブリッジデバイスをエミュレートするOSSを利用
<https://github.com/nfarina/homebridge>
- ところが、homebridge自身はいかなるHome Kit アクセサリのエミュレートもしないし、Home Kit Server protocolを実装しているわけでもない
- Home Kit Server protocolはHAP-NodeJSというライブラリが実装している
<https://github.com/KhaosT/HAP-NodeJS>
- 一方、Home Kitアクセサリ（各デバイス）のエミュレートはhomebridgeのプラグイン側で行う。wakeserverではhomebridge-wakeserverというプラグインをhomebridgeに登録している
- つまりhomebridgeはHAP-NodeJSを使用したHome Kitアクセサリのエミュレータをプラグインという形で記述できるようにしたプラットフォームと言える
- 何が言いたいかというと、homebridgeのソースだけ読んでもプラグインの書き方は理解できないのでHAP-NodeJSのソースも読みましょう

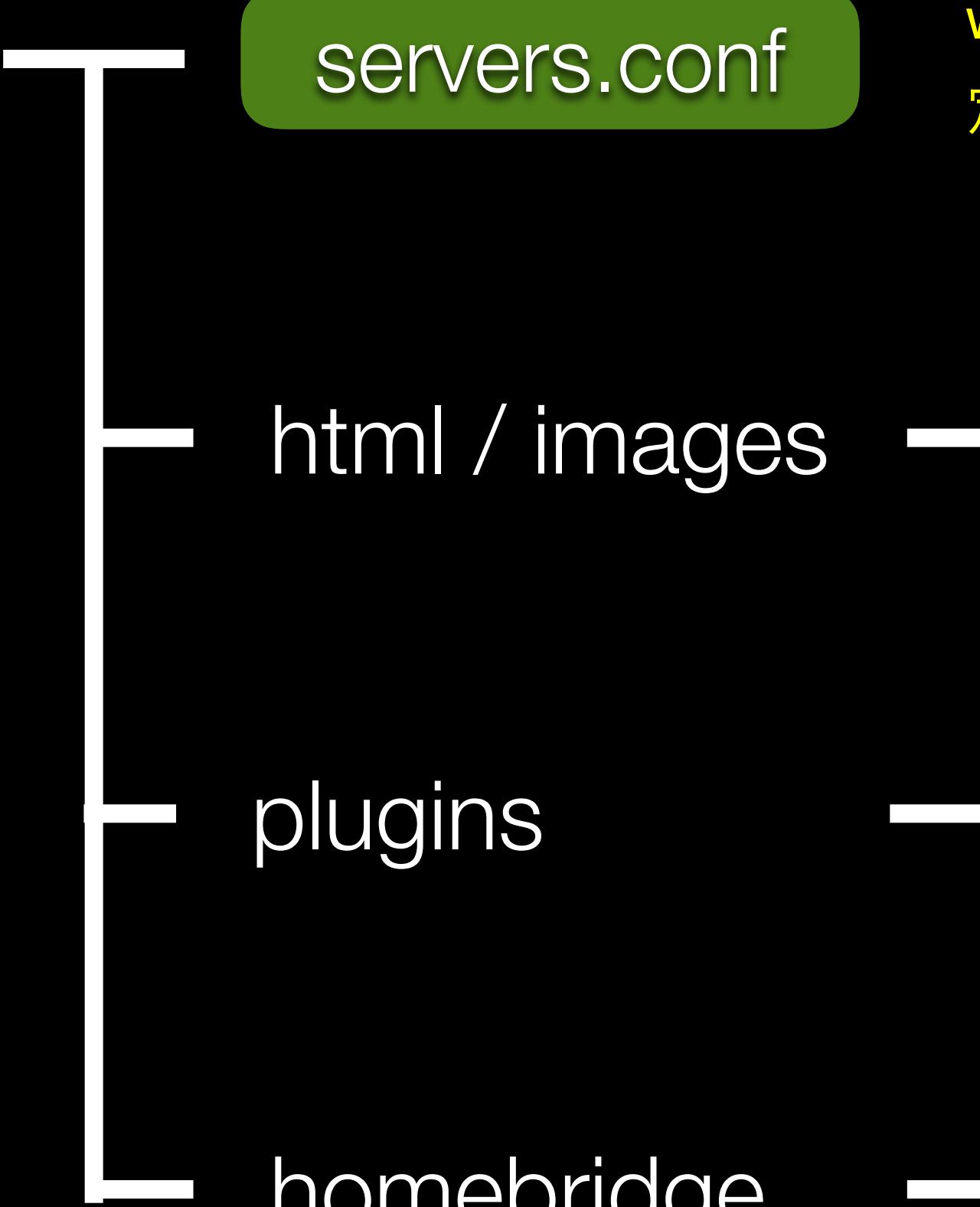




Personalize

個人環境毎に作成が必要なファイル

personal / ユーザ名



wakeserverで管理する機器の一覧を定義する設定ファイル

画像ファイル

servers.confで定義する機器の画像
Web UIで表示される

プラグイン

機器毎固有の生死判定や起動・停止
手順が必要な場合はプラグインプログラムを作成する

config.json

homebridge用の設定ファイル
通常はパーソナライズは不要であるが
servers.confに記述された機器の一部をHome Kitでの管理下から除外する場合や
仮想的なアクセサリを定義する場合は環境に合わせて記述を変更する

servers.conf : 概要

- wakeserverで管理する機器をJSONオブジェクトとして記述する
- 機器の配列とそれらをグルーピングしたオブジェクトの配列という階層構造から成る
- グループの定義ではグループ名、アイコン、接続時に最初からグループを展開表示するか否かを指定する
- アイコンを追加する場合はSVG形式で画像ライブラリファイルに追加すること
[html/images/svglib.xml](#)

```
[{  
    "groupName": "Servers",  
    "initial" : "close",  
    "icon": "server",  
    "servers": [  
        {  
            "label": "機器定義 (後述)"  
        },  
        {  
            "label": "機器定義 (後述)"  
        }  
    ]  
    "groupName": "Networking",  
    "initial" : "open",  
    "icon": "network",  
    "servers": [  
        {  
            "label": "機器定義 (後述)"  
        }  
    ]  
}]
```

servers.conf : 機器定義 - 例1 標準手順を使用する場合

```
{  
    "name":      "ganymede",  
    "ipaddr":    "192.168.22.11",  
    "macaddr":   "00:1F:33:EA:D0:64",  
    "maker":     "Netgear",  
    "comment":   "Netgear ReadyNAS Ultra 6",  
    "icon":      "images/rnu6.png",  
    "scheme": {  
        "type":      "unix",  
        "user":      "opiopan",  
        "diag":      "ping",  
        "on":        "wol",  
        "off":       "sudo-shutdown",  
        "ruser-off": "poff",  
        "services": [  
            {"type": "webui", "suffix": "admin"},  
            {"type": "ssh"},  
            {"type": "smb"},  
            {"type": "afp"}  
        ]  
    }  
}
```

servers.conf : 機器定義 - 例2 プラグインを使用する場合

```
{  
    "name":      "TV living@1",  
    "ipaddr":    "192.168.22.16",  
    "macaddr":   "E8:9D:87:57:69:EE",  
    "maker":     "TOSHIBA",  
    "comment":   "TOSHIBA REGZA 47Z2",  
    "icon":      "images/regza47Z2.jpg",  
    "scheme": {  
        "type":      "regza",  
        "diag":      "custom",  
        "on":        "custom",  
        "off":       "custom",  
        "volume":    "relay:AV amplifier",  
        "tvchannel": "custom"  
    }  
}
```

servers.conf : 機器定義 - 文法

パラメタ	条件	型	説明
<i>name</i>	required	文字列	機器名 機器名はservers.conf内で一意でなければならない
<i>ipaddr</i>	required	文字列	IPアドレス
<i>macaddr</i>	required	文字列	MACアドレス
<i>maker</i>	optional	文字列	製造者名
<i>comment</i>	required	文字列	自由なコメント項目として用意したが、装置の製品名（型名）が指定されることを想定している。
<i>icon</i>	required	文字列	機器の画像ファイルのパス 本パラメタで指定したファイルはpersonal/ユーザ名/html/imagesに配置すること。
<i>hidden</i>	required	真偽値	本パラメタが真 (true) の場合、Web UIで表示しない。 Home Kit連携でのみ利用する機器を登録する場合に指定する。
<i>scheme</i>	required	オブジェクト	生死確認の方法や起動・停止方法、また機器が持つcapabilityを指定する。 以下はpingによる生死確認のみをサポートする機器のscheme定義である。 <pre>{ "type": "general", "diag": "ping" }</pre> schemeオブジェクトの記述方法の詳細は後述する。

servers.conf : 機器定義 - schemeオブジェクト文法 ①

パラメタ	条件	型	説明
<code>type</code>	required	文字列	機器の種別を指定する。定義済みの種別は、"osx" と "unix"の2種類。この2種類のどちらかを指定した場合は、標準の方法(sshで停止コマンドを実行)で機器を停止することができる。それ以外の値を指定した場合、機器を停止するためにはpluginモジュールを作成する必要がある。 <code>plugin</code> パラメタを省略した場合、pluginモジュールはここで指定した名前と同名のファイルとして作成しなければならない。
<code>plugin</code>	optional	文字列	pluginモジュールの名前を指定する。本パラメタを省略した場合は <code>type</code> に指定した名前と同名のpluginモジュールを使用する。
<code>user</code>	optional	文字列	<code>type</code> に "osx"もしくは"unix"を指定した場合のみ本パラメタは有効。 機器の停止またはリブートを標準のsshで停止コマンドを投げる方法を使う際に、ローカル側(Raspberry側) のどのユーザ権限で実行するかを指定する。 ここで指定したユーザの公開鍵をリモートの利用者環境に登録しsshがパスワードなしでログインできる状態にしておく必要がある。
<code>ruser-off</code>	optional	文字列	<code>type</code> に "osx"もしくは"unix"を指定した場合のみ本パラメタは有効。 機器の停止またはリブートを標準のsshで停止コマンドを投げる方法を使う際に、リモートのどのユーザでloginするかを指定する。 <code>off</code> や <code>reboot</code> の値に"sudo-shutdown"を指定した場合はリモートでsudoコマンドによりshutdownコマンドを実行するが、この時sudoがパスワード指定なしで実行できることをwakeserverは期待する。ここで指定したユーザが少なくともshutdownコマンドの実行をパスワードなしでできるよう/etc/sudoersを設定すること。

servers.conf : 機器定義 - schemeオブジェクト文法 ②

パラメタ	条件	型	説明
<i>diag</i>	required	文字列	機器の生死監視の方法を指定する。 ping pingの応答がある場合に動作中と判断 arp ARPの応答がある場合に動作中と判断 alwayson 常にon normallyoff 常にoff custom プラグインプログラムを使用
<i>on</i>	optional	文字列	機器が起動能力を持つ場合に起動方法を指定する。 wol Wake on LAN により起動 custom プラグインプログラムを使用
<i>off</i>	optional	文字列	機器が停止能力を持つ場合に停止方法を指定する。 sleep sshでMacのpmsetコマンドを投入しスリープする typeに"osx"を指定した場合のみ指定できる sudo-shutdown sshでsudoコマンド経由でshutdownコマンドを投入し停止する typeに"osx"または"unix"を指定した場合のみ指定できる custom プラグインプログラムに状態を問い合わせる
<i>reboot</i>	optional	文字列	機器が再起動能力を持つ場合に再起動方法を指定する。 sudo-shutdown sshでsudoコマンド経由でshutdownコマンドを投入し再起動する typeに"osx"または"unix"を指定した場合のみ指定できる custom プラグインプログラムに状態を問い合わせる

servers.conf : 機器定義 - schemeオブジェクト文法 ③

パラメタ	条件	型	説明
<i>volume</i>	optional	文字列	<p>機器が音量調整能力を持つ場合に調整方法を指定する。</p> <p>custom プラグインプログラムを使用</p> <p>relay:機器名 他の機器の音量調整機能を利用</p> <p>“relay:機器名”の指定は、例えばTVとAVアンプが接続されておりTVの音量調整をAVアンプに指示して行う場合などを想定している。</p> <p>現在このパラメタを理解できるのはHome Kit連携のみ。WebUIには影響を与えない。</p>
<i>tvchannel</i>	optional	文字列	<p>機器がTVチューナー機能を持つ場合にチャネル選択方法を指定する。</p> <p>custom プラグインプログラムを使用</p> <p>relay:機器名 他の機器のチャネル選択機能を利用</p> <p>“relay:機器名”の指定は、例えばTVとケーブルテレビ等のセットトップボックスが接続されている環境でセットトップボックス側のチャネル選択機能を利用する合などを想定している。</p> <p>現在このパラメタを理解できるのはHome Kit連携のみ。WebUIには影響を与えない。</p>
<i>services</i>	optional	配列	<p>機器が提供するサービスを表現するオブジェクトの配列を指定する。</p> <p>なお、サービスとは"http://"や"ssh://"等、URLで表現可能なものを指す。</p> <p>以下はtcp/80でhttpとして提供されるWebUIを記述したserviceオブジェクトの例である。</p> <pre>{"type": "webui"}</pre> <p>serviceオブジェクトの詳細は次ページ以降で示す。</p>

servers.conf : 機器定義 - serviceオブジェクト文法

パラメタ	条件	型	説明
<code>type</code>	required	文字列	<p>サービス種別を指定する。これはURLスキームを指定することにもなる。</p> <p>webui 機器が提供するWebUI (httpスキーム) ssh 機器のsshサーバ機能 (sshスキーム) vnc 機器のvncサーバ機能 (vncスキーム) smb 機器のCIFSによるファイル共有機能 (smbスキーム) afp 機器のAFPによるファイル共有機能 (afpスキーム) config-app 機器専用設定アプリを起動 (<code>prefix</code>でスキームを指定する必要あり)</p>
<code>prefix</code>	optional	文字列	<p><code>type</code>で指定したサービス種別と対応するURLスキームをオーバライドする。</p> <p>次のserviceオブジェクトは、 "apmanage://アドレス/" というURLを生成する。</p> <pre>{"type": "config-app", "prefix": "apmanage"}</pre>
<code>suffix</code>	optional	文字列	<p>URLの後ろに付加するパスサフィックスを指定する。</p> <p>次のserviceオブジェクトは、 "http://アドレス/admin" というURLを生成する。</p> <pre>{"type": "webui", "suffix": "admin"}</pre>
<code>port</code>	optional	文字列	<p>URLでポート番号を明示する場合に指定する。</p> <p>次のserviceオブジェクトは、 "http:8080//アドレス/" というURLを生成する。</p> <pre>{"type": "webui", "port": "8080"}</pre>

Plugin仕様：概要

Pluginプログラムとは

- service.conf機器設定において、schemaオブジェクトで“custom”と指定した機器の制御操作を実装するプログラム
- CGIプログラムから起動される

ファイル名

- schemaオブジェクトのtypeパラメタで指定した名前と同じファイル名の実行可能ファイルを作成する
- ただしschemaオブジェクトのpluginパラメタを指定した場合はpluginパラメタの値がファイル名となる
- ソースのpersonal/ユーザ名/pluginsに配置すること

コマンド syntax

以下のように4～6個の引数とともに起動される。

```
plugin type ipaddr macaddr interval [attribute [value]]
```

- *type* : 動作種別 (diag | on | off | reboot | attribute)
- *ipaddr* : IPアドレス
- *macaddr* : MACアドレス
- *interval* : 最低sleep時間
- *attribute* : 属性名 (第一引数がattributeの場合のみ指定される)
- *value* : 属性の値 (第一引数がattributeの場合のみ指定される)

引数の詳細は次頁以降で説明する。

Plugin仕様：コマンド引数詳細

順序	引数名	説明
1	<i>type</i>	Pluginコマンドの動作モードを指定する。 diag 機器の生死状態を返却する 機器が動作している場合は0を、停止している場合は0以外を終了コードとして返す on 機器を起動する off 機器を停止する reboot 機器を再起動する attribute 機器の属性値の取得・設定をおこなう 標準出力に属性値を出力する (詳細はattribute引数の説明を参照)
2	<i>ipaddr</i>	対象となる装置のIPアドレス
3	<i>macaddr</i>	対象となる装置のMACアドレス
4	<i>interval</i>	<i>type</i> が“diag”的場合は、本引数に指定された秒数待ってからコマンドを終了する
5	<i>attribute</i>	<i>type</i> が“attribute”的場合、本引数に属性名が指定される。以下の属性はwakeserver側で定義済。 volume 音量コントロールに使用、0から100までの数値 tvchannel TVのチャネル選択に使用、0から100までの数値 tvband TVのチャネル種別の切替えに使用、terrestrial(地デジ)、bs、csのいずれかの文字列 それ以外の属性を自由に定義することも可能。現在ユーザ定義属性はHome Kit連携でのみ利用可。
6	<i>value</i>	<i>type</i> が“attribute”的場合、本引数が属性値となる。本引数が省略された場合は属性値を変更しない。

Home Kit連携：config.jsonの設定方法

- 通常は右の基本テンプレートの内容を変更する必要はない
右の設定を利用してすることでWeb UIで管理できる機器が全て Home Kitで管理できるようになる
- pinにiOSデバイスにwakeserverを登録する際に入力するPINを設定する
- On / Off capabilityを持つ機器のみHome Kit管理下に置きたい場合はenableOnlySwitchをtrueにする
- servers.confで定義した機器の一部をHome Kitの管理下から除外したい場合は、ignoreServersに機器名の配列を指定する
- wakeserverはいくつかの擬似的なアクセサリを提供する
擬似アクセサリを利用する場合はアクセサリ定義をaccessoriesに追加する
擬似アクセサリの定義方法は次頁を参照

```
{  
  "bridge": {  
    "name": "Wakeserver Portal",  
    "username": "CC:22:3D:E3:CE:21",  
  
    "manufacturer": "opiopan",  
    "model": "Wakeserver brigde",  
    "serialNumber": "00:00:00:00:00:00",  
  
    "port": 51826,  
    "pin": "123-45-678"  
  },  
  
  "platforms": [  
    {  
      "platform": "Wakeserver",  
      "name": "Wakeserver"  
      "enableOnlySwitch": false,  
      "ignoreServers" : [  
      ],  
      "accessories" : [  
      ]  
    }  
  ]  
}
```

Home Kit連携：擬似アクセサリの設定方法

- wakeserverのhomebridgeプラグインは servers.confに登録された機器をHomeKitのアクセサリとしてエクスポートするだけではなく、擬似的な機器の振る舞いをするアクセサリをエクスポートすることができる
- これら擬似アクセサリを使用する場合は config.jsonのwakeserverプラグインの設定で、**accessories**属性に擬似アクセサリ定義の配列を指定する
- 現在実装している擬似アクセサリとその設定方法は次ページ以降を参照

```
{  
  "bridge": {  
    省略  
  },  
  "platforms": [  
    {  
      "platform": "Wakeserver",  
      "name": "Wakeserver"  
      "enableOnlySwitch": false,  
      "ignoreServers": [  
        ],  
      "accessories" : [  
        {  
          擬似アクセサリ定義  
        },  
        {  
          擬似アクセサリ定義  
        }  
      ]  
    }  
  ]  
}
```

Home Kit連携：アメダス温度計 擬似アクセサリ

- 気象庁のWebページからアメダスデータを抽出し温度計として振る舞うアクセサリ
- nameにアクセサリの名前を指定する
- typeに"psensor"を指定する
- webに参照するアメダスのロケーションを表すareaCodeとgroupCodeを指定する
- areaCodeとgroupCodeの値は、下記サイトから目的の地区のページにアクセスし、そのURLのパラメタから抽出できる
http://www.jma.go.jp/jp/amedas_h/index.html

```
{  
  "bridge": {  
    省略  
  },  
  "platforms": [  
    {  
      "platform": "Wakeserver",  
      省略  
    },  
    "accessories" : [  
      {  
        "name": "Outer Temperature",  
        "type": "psensor",  
        "web": {  
          "areaCode": "000",  
          "groupCode": "35"  
        }  
      }  
    ]  
  ]  
}
```

Home Kit連携：属性 擬似アクセサリ (switch)

- pluginプログラムで実装した任意の属性をコントロールするアクセサリを定義できる
ここではswitchタイプのアクセサリを定義する例を示す
- nameにアクセサリ名を指定する
- typeに"attribute"を指定する
- serverに属性を実装している機器の名前を指定する
(servers.confで定義した名前)
- atypeに"switch"を指定する
- on、offにスイッチをonにする場合とoffにする場合に設定する属性の名前と値を指定する
- stateにスイッチの状態を取得するための属性の名前とonと判定する値を指定する（指定形式はon、offと同じ）
右の例のようにstateの指定を省略した場合は、スイッチの状態は常にoffとなる

```
{  
  "bridge": { 省略 },  
  
  "platforms": [  
    {  
      "platform": "Wakeserver",  
      省略  
    },  
    {  
      "accessories": [  
        {  
          "name": "Jukebox@living",  
          "type": "attribute",  
          "server": "moon",  
          "atype": "switch",  
          "on": {  
            "attribute": "airplay",  
            "value": "living"  
          },  
          "off": {  
            "attribute": "airplay",  
            "value": "off"  
          }  
        }  
      ]  
    }  
  ]  
}
```

Home Kit連携：属性 擬似アクセサリ (温度計)

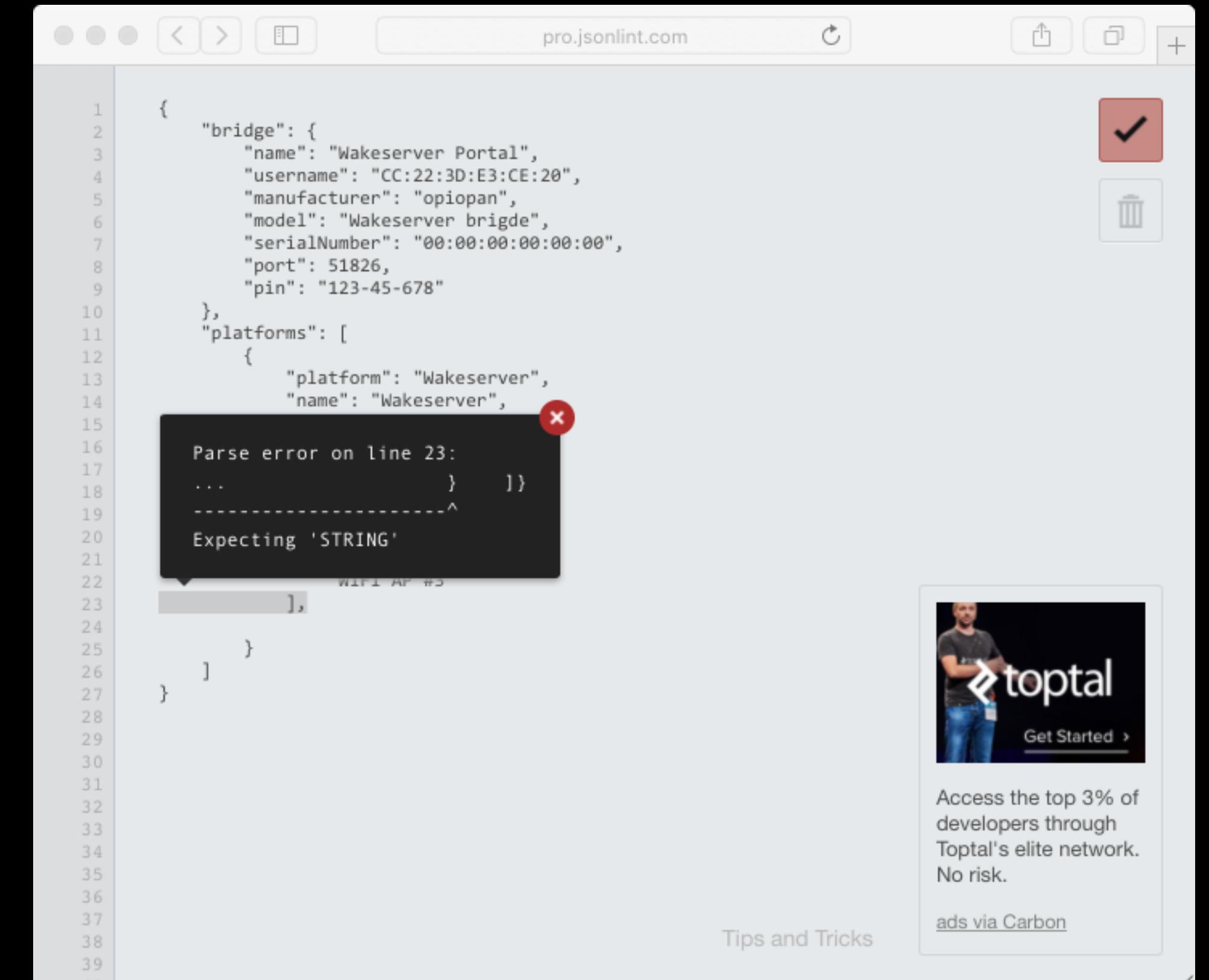
- pluginプログラムで実装した任意の属性をコントロールするアクセサリを定義できる
ここでは温度計タイプのアクセサリを定義する例を示す
- nameにアクセサリ名を指定する
- typeに"attribute"を指定する
- serverに属性を実装している機器の名前を指定する(servers.confで定義した名前)
- atypeに"temperature"を指定する
- valueに温度を返却する属性名を指定する

```
{  
  "bridge": { 省略 },  
  "platforms": [  
    {  
      "platform": "Wakeserver",  
      省略  
      "accessories" : [  
        {  
          "name": "Temperature@living",  
          "type": "attribute",  
          "server": "mserver",  
          "atype" : "temperature",  
          "value":{  
            "attribute": "temperature",  
          }  
        }  
      ]  
    }  
  ]  
}
```

Debugging tips

services.conf、 config.jsonのsyntaxチェック

- services.confやconfig.jsonにsyntaxエラーがある場合、wakeserverdやhomebridgeの起動が失敗する
- しかし、両デーモンとも自動再起動されるので定義ファイルにエラーがあることに気がつけないことが多い
- 定義ファイルを編集した場合は、下記のようなJSON文法チェッカーWebサービスなどを使用して、文法が正しいことをチェックしておくことを推奨する
<http://pro.jsonlint.com>



The screenshot shows a web browser window with the URL 'pro.jsonlint.com'. The page displays a JSON configuration file with line numbers from 1 to 40. A tooltip is overlaid on line 23, indicating a 'Parse error on line 23: ... Expecting 'STRING''. The JSON code includes sections for 'bridge' and 'platforms', with the error pointing to a line within the 'platforms' array. The browser interface includes standard navigation buttons and a toolbar with icons for file operations.



Access the top 3% of developers through Toptal's elite network. No risk.

[ads via Carbon](#)

Tips and Tricks

デーモンのエラー確認

- services.confやconfig.jsonがJSONの文法に適合していてもwakeseserverの仕様に適合していない場合、wakeserverdやhomebridgeの起動が失敗したり途中終了や処理異常となる場合もある
- このような場合はサービスとしてデーモンを停止し、端末上で起動しエラーメッセージを可視化すると原因特定につながりやすい
- なおデバッグ完了後サービスを起動するのはmake installしなおすのが早い

wakeserverdを端末で起動する手順

```
$ sudo systemctl stop wakeserver.service  
$ sudo /var/www/wakeserver/daemon/wakeserverd
```

homebridgeを端末で起動する手順

```
$ sudo systemctl stop homebridge.service  
$ BUG=* sudo -u homebridge homebridge -D -U /var/homebridge
```

プラグインのデバッグ

- まずはプラグインコマンド単体で動作させ問題なく動作することを確認しましょう
- 次にREST API越しに正しく動作するか確認しましょう。
うまく動作しない場合はresponseに着目してデバッグします。
CGIレベルで正しく動作できている場合はbodyがJSONデータとなっており、message属性の内容で原因が判定できる場合も多いです。
CGIが実行できない問題が発生している場合は、response headerの内容から原因を推測します。CGIの起動と結果の取得はcurlコマンドを利用するのが便利です。
以下は wakeserver-attribute.cgiを実行する例です。

```
$ curl -F'value=off' -F'target=moon' -F'attribute=airplay' http://localhost:8080/cgi-bin/wakeserver-attribute.cgi
{"message": "Succeed", "result": true, "value": "off"}
```

A wide-angle photograph of a modern, two-story house at dusk or night. The house features large glass windows and doors that reflect the warm interior lights and the dark sky. The exterior is dark, possibly black or dark grey. In the foreground, there's a paved area with some outdoor furniture and a small potted plant. The background shows a dark sky with a few stars and a distant city skyline with lights.

THANK YOU