APX UI

New UI

1. Footprint reduction
2. Multi-theme(run-time) support.
3. Ease of rebranding – logos etc. should be changeable without changing backend or front end code.

The Basics from UI's point of view

1. JavaScript – which makes the browser work like an application.
2. jQuery – a JS library. Simpler, faster and more powerful.
3. AJAX – technology that allows JS functions to request information from the server.

APX UI – Communication with Backend

- The backend uses CGIs that communicate with the frontend using predefined XMLs.
- Getting the Configuration –

  HTTP GET is used to get configuration information specific to a module in an XML format.

  Request:

GET /cgi/xml_action.cgi?method=get&module=duster&file=wlan_basic_settings HTTP/1.1

Response for this would be:

```
<?xml version="1.0" encoding="US-ASCII" ?>
    <RGW>
        <wlan_settings>
            <mac>00:50:43:20:cb:2a</mac>
            <channel>0</channel>
            <bandwidth></bandwidth>
        </wlan_settings>
    </RGW>
```


MARVELL

APX UI – Communication with Backend

- Changing the configuration

HTTP POST is used to set/change the configuration.

Request:

POST /cgi/xml_action.cgi?method=set&module=duster&file=wlan_basic_settings HTTP/1.1

Response for this would be:

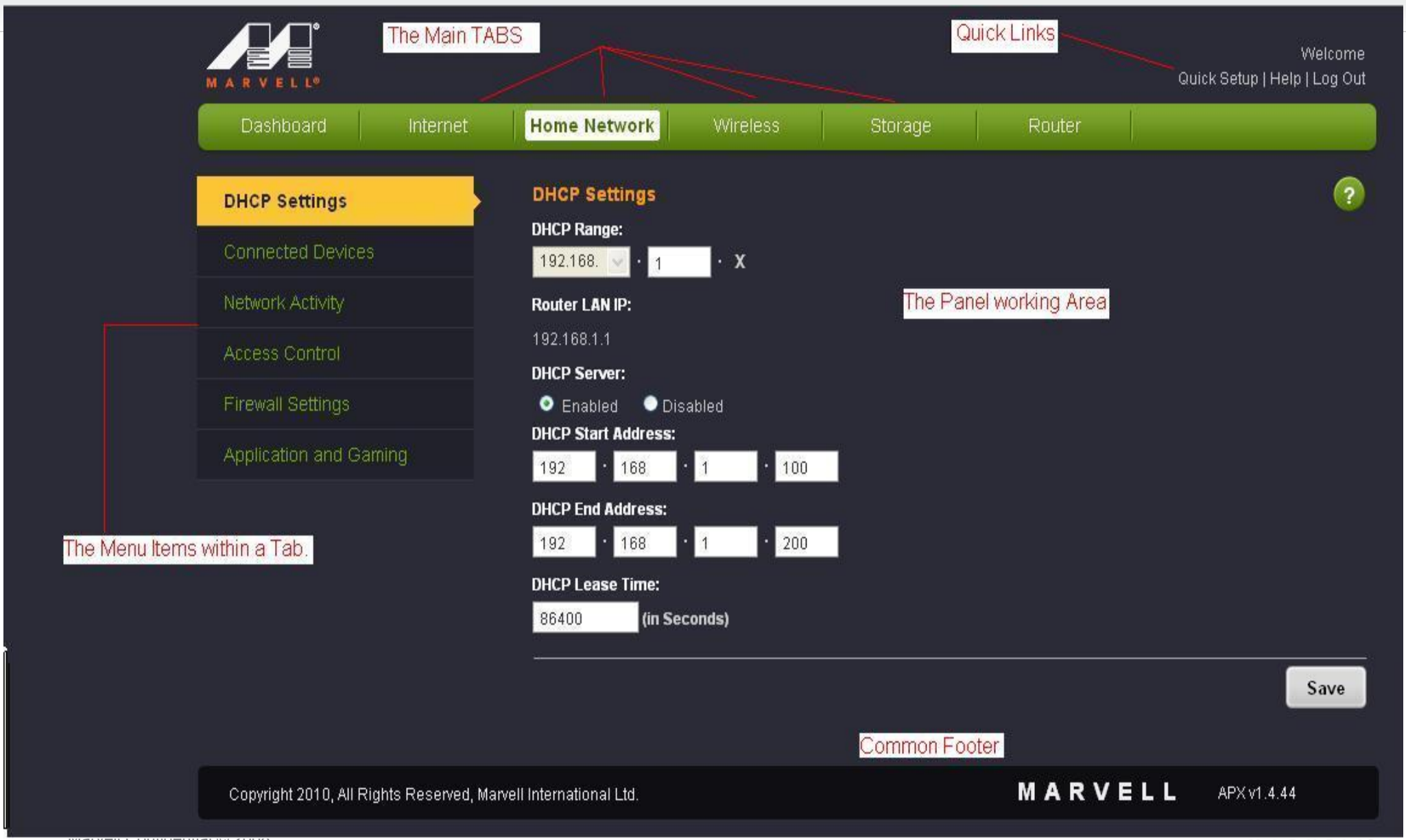
```
<?xml version="1.0" encoding="US-ASCII" ?>
    <RGW>
        <wlan_settings>
            <channel>2</channel>
            <bandwidth>2</bandwidth>
        </wlan_settings>
    </RGW>
```


MARVELL

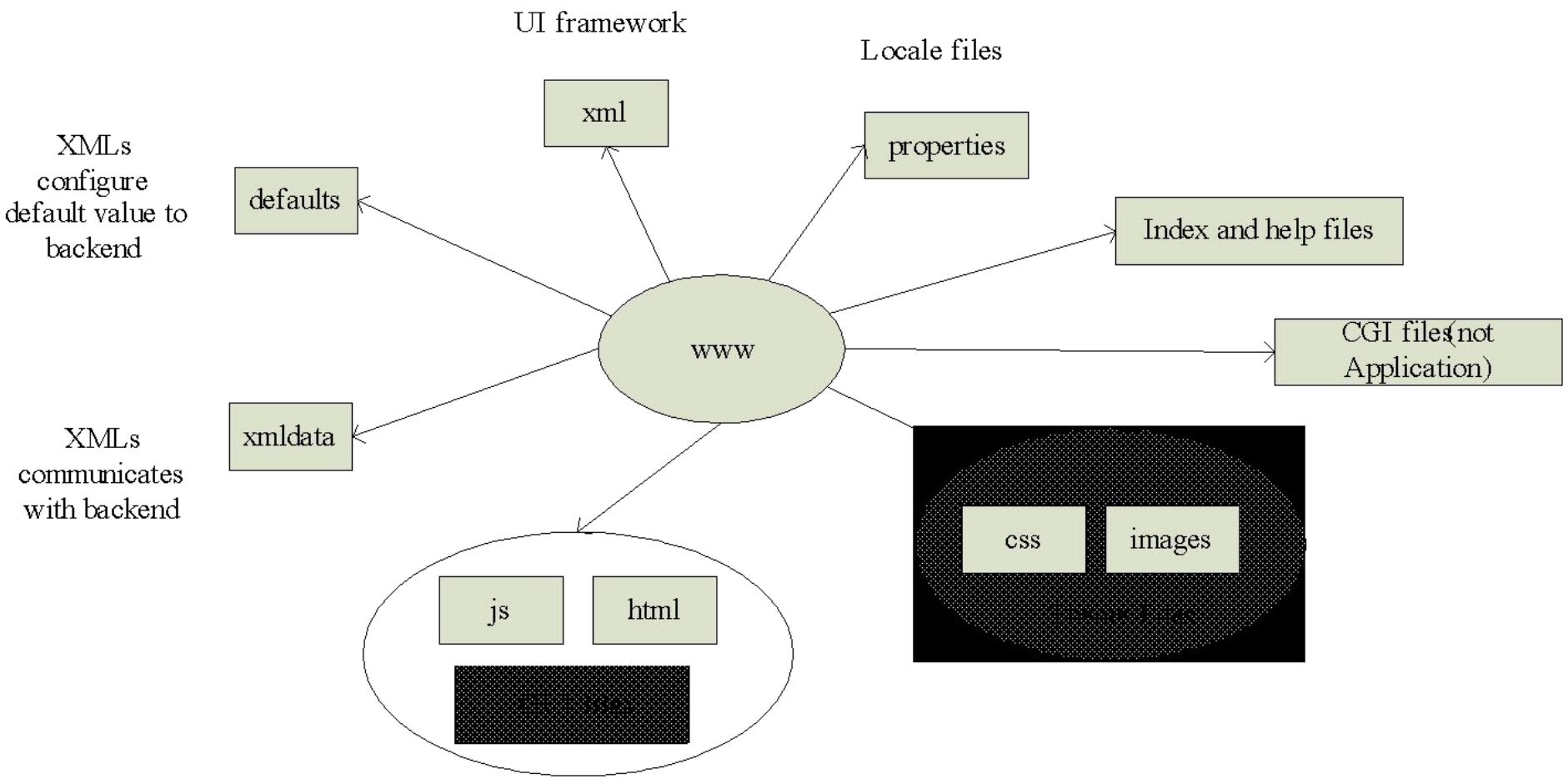
APX UI –Backend Interface

- All the XML GET/POST are forwarded by the webserver(mongoose) to the CGI task. The CGI task invoke the duster_parse to dispatch the XML GET/POST request to the back-end modules.
- The XML document hierarchy mirrors the hierarchy maintained by the persistent storage manager(PSM)
- For get request will retrivers variables from PSM and returns values in an XML format and invoke registered GET API.
- For set request will save the value to PSM and invoke registered POST API.

APX UI - Overview



APX UI- Directory Structure



APX UI Key Files

1. www/xml/ui.xml
   Tab definitions. Top level user interface layout.
2. www/index.html
   New javescript files need to be inclued in the index.html to load by the client. HTML files are loaded by the javascript files when the javascript is invoked.
3. www/properties/Messages_<localed>.properties
   Locale(e.g en) can be used for development. New property strings nend to be tanslated and added to other supported locales as well.
4. www/js/panel/<newapp>/<function>.js
   The source code layout convention is to put javascript code for each application in a separate directory with a separate javascript file for each major feature.
5. www/html/<newapp>/<function>.html
   Any static html markip associated with the tan panel needs to be created.
6. www/help_<locale>.html
   Add sections to the help file for each new application feature. It need tanslated and added to each locale specific help files.

APX UI - Sample Screen

1.Entry in App Framework(ui.xml)

```
</Tab>
<Tab Name='tHome_Network' type='submenupresent'>
    <Menues>
        <Menu id='mDHCP_Settings' implFunction='objDHCP_Settings' xmlName='lan' />
        <Menu id='mConnected_Devices' implFunction='objConnectedDev' xmlName='device_management_all' />
        <Menu id='mAccess_Logs' implFunction='objAccess_Logs' xmlName='detailed_log' />
        <Menu id='mAccess_Control' implFunction='objAccess_Control' xmlName='internet_access_control' />
        <Menu id='mFirewall_Settings' implFunction='objFirewall_Settings' xmlName='firewall' />
        <Menu id='mApp_Gaming' implFunction='objApp_Gaming' xmlName='port_forward_trigger' />
    </Menues>
</Tab>
<Tab Name='tWireless' type='submenupresent'>
    <Menues>
        <Menu id='mWire_Set' implFunction='objWire_Set' xmlName='wlan_basic_settings' />
```

name of the xml

Resource ID

jquery class which implements the panel

2. The HTML file of the module
3. JS file – the jQuery class implementation.
4. Entry in properties file.

MARVELL

APX UI – Additional Stuff

* Locales Support

1. Properties file corresponding to each locale
2. One place modification for changing label text, popup text etc.

* Multiple Themes

1. Changing the theme doesn't require any code change in HTML or in JS
2. "css" and "images" folders corresponding to each theme
3. The "theme" duster module for communicating to the backend the change of theme.

APX UI – Adding a module

1)Depending on where you want to include the new panel, add an entry in ui.xml inside the corresponding “Tab” and “Menues” tags.
2)Create the html file in specific folder within the html directory.
3) Create the js file in specific folder.
Responsibilities of each js class ->
1. Implement the onLoad fun that populates the DOM elements as per the XML fetched.
2. Localization, validation etc as per the screen requirement.
3. Implement the onPost which fetches modified field values and invokes the base API to send data to backend.
4. Implement the setXMLName fun which is called by the
4)Include all the label texts, button texts etc. in the properties files of both the locales.
5)Don’t forget to include the newly created js file in index.html using script tag.

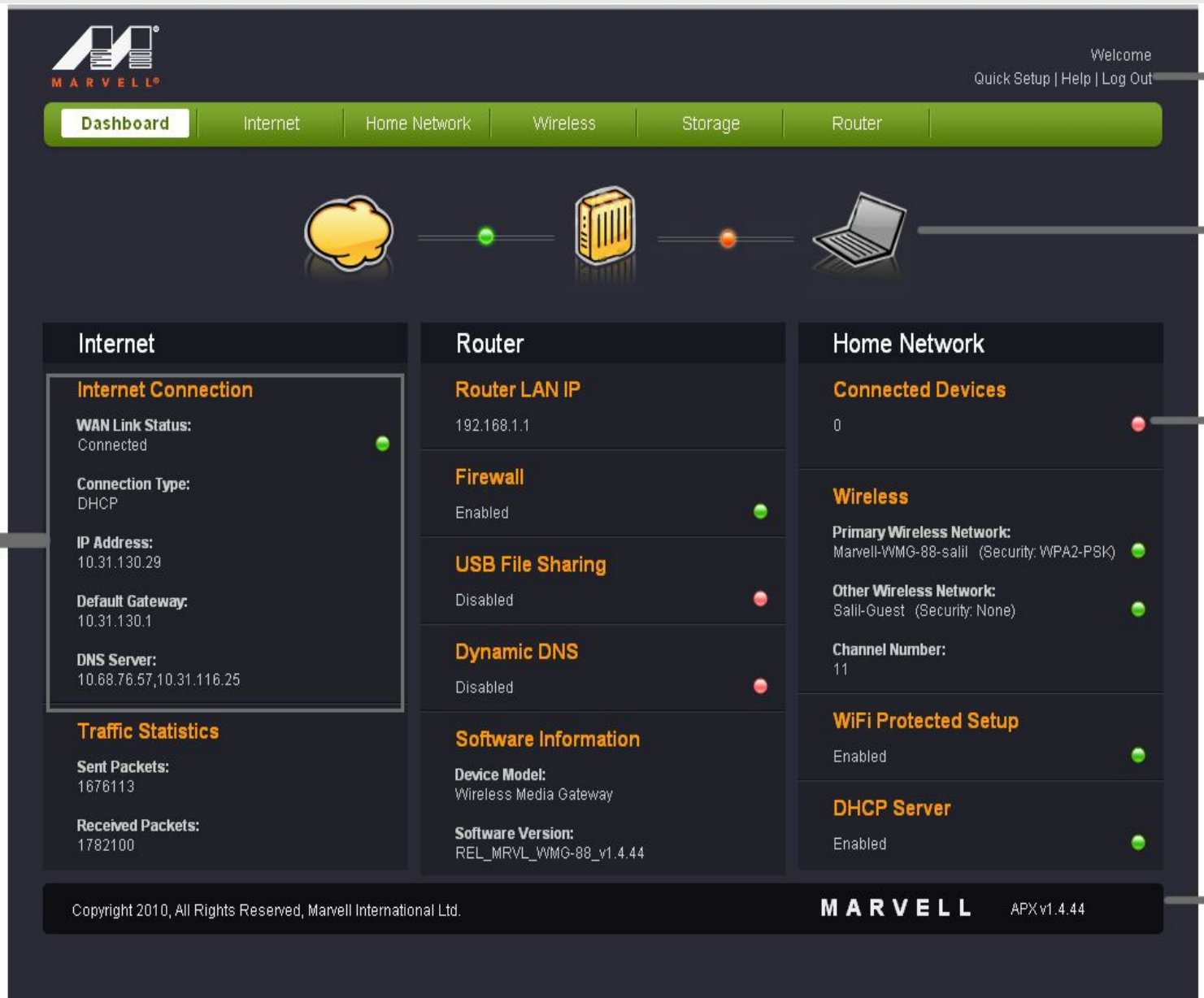
APX UI – Adding a module

* Commonly required APIs in the jQuery code –

- getData(xmlName) – invokes the AJAX call for a GET from backend on the specified xml name.
- lableLocalization(), buttonLocalization() – pass an array of elements returned from document.getElementById
- putMapElement(2-D array, index, path-of-xml-node, value-of-that-node)

* The jQuery class must implement the loadHTML function. In it the callProductHTML API needs to be used to specify the html file corresponding to this particular panel.

APX UI – Dashboard



APX UI – Quick Access Links

* Quick Setup

1. Behavior after Log-In
2. Skip setup option
3. Functionality change in Quick setup behavior than previous UI.

* Help Page

1. Opens in new Tb.
2. Context specific help for each panel/screen.

* Log Out

MARVELL

APX UI – Commonly offered widgets

- A bunch of commonly required widgets are provided in the controls/ directory within the js code.
- For example – enabled_disabled control, ip_address control etc.
- These widgets expose a set of APIs which can be directly used.