# OMV 3. X  DIY Plugin Guide - Release 1.0

## Preface

Use this guide at your own risk. It is for all users looking to develop their own plugin for OpenMediaVault.  Knowledge of php, javascript, json, shell scripting, and creating debian packages is recommended. Do NOT develop on your main OMV machine. It is highly recommended to develop on a virtual machine. Please refer to the online documentation for more information:
http://docs.openmediavault.org/3.x/index.html
View the source code for OpenMediaVault and plugins from the core developer Volker Theile here:
https://github.com/openmediavault/openmediavault
View the source code from OpenMediaVault Plugin Developers here:
https://github.com/OpenMediaVault-Plugin-Developers
These can also be viewed as examples on how the plugins are constructed.

## Getting Started

### Coding Guidelines

Please take a moment to familiarize yourself with the coding guidelines to be used when developing a plugin for OpenMediaVault. Following these guidelines will make it easier to understand how a plugin works and allow others to contribute to them.

- PSR-2 Coding Format http://www.php-fig.org/psr/psr-2/
- Official OMV Coding Guidelines
  http://wiki.openmediavault.org/index.php?title=Coding_Guidelines

### How plugins are created

Before we go into detail on each section of the plugin development, we want to give you a rough overview of what's necessary from the start to end for an OpenMediaVault plugin.

#### The Webgui

A plugin starts by creating a Plugin.js file which registers the plugin on OpenMediaVault. This file allows a plugin to show up as a node in one of the sections (System, Services, etc...) in the Webgui. See figure 1 below.

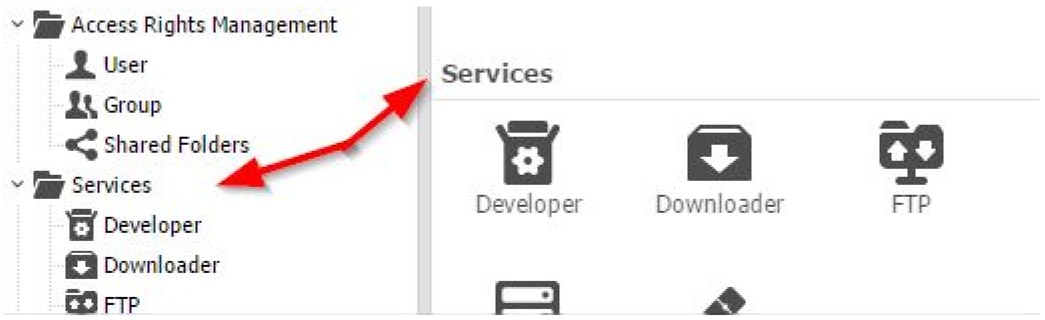
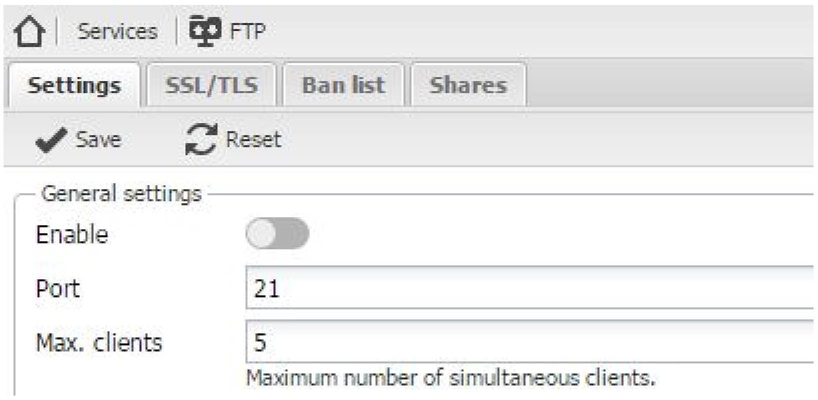


Figure 1. - Registered nodes under Services. The Treeview is on the left and main view on the right.

Each plugin node then can have multiple sections also known as tabs. Those sections define the visual content in the Webgui. For example, the enable/disable option, or a text field to define which port your plugin should listen on. This will diverse from plugin to plugin, where some plugins may only have a single enable button, while others may have a dozens of options to customize. See figure 2 below. As you can see, there are many options available for the FTP plugin that also includes other sections or tabs.



General settings

Enable

Port 21

Max. clients 5

Maximum number of simultaneous clients.

Figure 2. - Multiple options in a section.

## Datamodels

Datamodels was introduced into OpenMediaVault after 3.0.13 beta. It uses json files to make it easier to add or change functionality in your plugin. These are the parameters that are defined in the Webgui and it can be created, modified, or deleted from your Webgui options. They are also used to verify the parameters are the correct when it is used in a function that writes to the config.xml.   We will go into more detail about using datamodels later on in this guide.

## The RPC Service

RPC stands for remote procedure call. This will initiate functions when they are called upon by an action such as enabling the plugin. In order to store the settings of your plugin a file called config.xml has to modified. The config.xml file contains all parameters that the Webgui holds. The RPC files are the link between the Webgui and config.xml. The configuration file is located in

/etc/openmediavault/config.xml. There's also a template located in /usr/share/openmediavault/templates/config.xml. It is a good idea to make a backup of the config.xml before testing your plugin or any changes that have been made.

### Modules

Modules are used to monitor and execute changes made to your plugins configuration. Modules start and stop services and they also update the status of the service in the web interface.

### MKCONF

Last but not least you need a mkconf file. This writes the configuration file for the service of your plugin. This is a shell script which reads the variables from the config.xml and writes it into the configuration files for your plugin application. This is the last step to have a working plugin, not including the packaging of the plugin yet. See figure 3 below which shows the process flow.

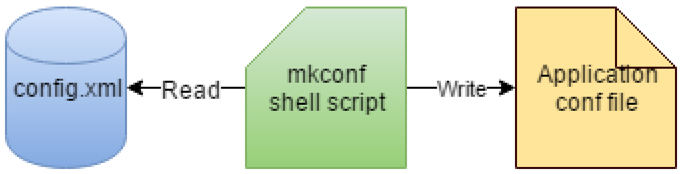


Figure 3. The shell script reads the data in config.xml and writes it to the application conf file.

## A plugin walkthrough - minidlna

So now that you got a rough idea of how a OpenMediaVault plugin works, let us walk through a plugin creation from the beginning to the end. We choose the minidlna plugin because it's simple and should show you the basics for all parts of the plugin development. You can view the source code here: https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna

Create a working directory such as /home/myname/development/minidlna. You can replace minidlna with the name of your plugin. You will also need to create a subfolder named debian in this directory. This is required for the packaging files needed later on. An introduction to Debian pacakaging can be found here: https://wiki.debian.org/IntroDebianPackaging Additional details will be covered at the end of this guide. This guide assumes you will use this as your current working directory and all paths mentioned below are contained in this directory.

### The Webgui

The Webgui uses Ext JS (https://www.sencha.com/products/extjs/#overview) as the base system. You can find all the classes here http://docs.openmediavault.org/3.x/group__webgui.html that will allow you to create the interface for your plugin. There are 3 sections of the GUI you can create an interface for. They are admin, user, and public. Each location allows specific access to the plugin. For example, only the admin can see plugins under the admin folder. If you want to make a plugin for users, then place it under users. Public will allow both the admin and users to see the plugin. They are located in the following paths below.

/var/www/openmediavault/js/omv/module/admin/
/var/www/openmediavault/js/omv/module/user/
/var/www/openmediavault/js/omv/module/public/

## Create the folder for the Webgui

In this example we will place the minidlna plugin under the services section. From the command line, create the folder using the path below. Create the complete folder structure within your working directory and remember to leave out the trailing slash.

var/www/openmediavault/js/omv/module/admin/service/minidlna
*note: Do not add the trailing slash.

## Create the node Minidlna.js

A node is a single object in the navigation tree view that appears on the left. Each node is declared in a separate JavaScript file. Creating this node will show in the tree view and the main view as DLNA. See figure 4 below that shows the node in the tree view for reference. We will show you how you define how a plugin is named in the GUI in the code below.

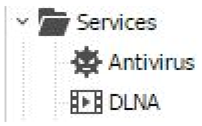


Figure 4. - The minidlna node in the tree view.

Next we will walk through the code in Minidlna.js. We will add comments in the code here to describe what is taking place. You can write the code on your own or use the code on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/var/www/openmediavault/js/omv/module/admin/service/minidlna/Minidlna.js

### Licensing

Please keep the licensing header intact on all files. You may add/modify yourself as the author and the copyright for your own plugin. All files must include the GPL Version 3 licensing information.

Code
```
/**
 * @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
 * @author    Volker Theile <volker.theile@openmediavault.org>
 * @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
 * @copyright Copyright (c) 2009-2013 Volker Theile
 * @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
// require("js/omv/WorkspaceManager.js") The node has a main view that is called a workspace.
OMV.WorkspaceManager.registerNode({ Register the node in the Workspace Manager.
   id      : "minidlna", This is the ID of the node. This must be unique.
   path    : "/service", The node will be located under services. See figure 1 and 4 above.
   text    : _("DLNA"), This is the text to display for the name of the node. Note the text is wrapped in _(). This means it can be translated. See the information below about text and translations.
   icon16  : "images/minidlna.png", This is the image to display in the navigation tree menu.
   iconSvg : "images/minidlna.svg" This is the image to display in the main menu.
});
```

### Images

Please note the images, as you will need to add a 16x16 pixel png file to be used in the tree menu and a svg file for the main view. Place them in the images folder located at var/www/openmediavault/images/.
You can create your own or search the internet. Be sure to include any copyright information.

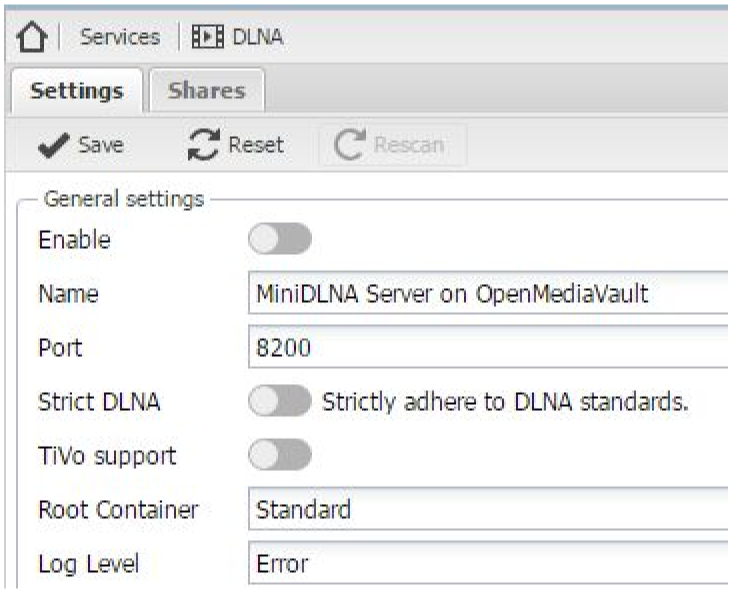
### Text and Translations

If you want your text to be translated, make sure the text option is wrapped in _(). Text that you don't want translated, just use quotation marks. Translations can be found in usr/share/openmediavault/locale/. Review the locale folder on github for more details.
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/tree/master/usr/share/openmediavault/locale

## Create the workspace Settings.js

The workspace allows you to design how your interface will look. In this example we will show the most commonly used for a plugin, OMV.workspace.form.Panel. This is an extension of the primary workspace class. A couple of features of this extension include a built in save button, reset button, and it can be used as a tab. Additional documentation can be found here:

http://docs.openmediavault.org/3.x/classOMV_1_1workspace_1_1form_1_1Panel.html See figure 5 below as this will be the result of creating the Settings.js file. Note the shares tab which will be covered in the next section.



Enable
Name: MiniDLNA Server on OpenMediaVault
Port: 8200
Strict DLNA: Strictly adhere to DLNA standards.
TiVo support
Root Container: Standard
Log Level: Error

Figure 5. - The Settings.js tab.

Now we will walk through the code in Settings.js file. We will add comments in the code here to describe what is taking place. Please note that not all of the code will be commented on. Any code that is not commented on should be considered easy to understand what is taking place. You can write the code on your own or use the code on Github located here: https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/var/www/openmediavault/js/omv/module/admin/service/minidlna/Settings.js

Code

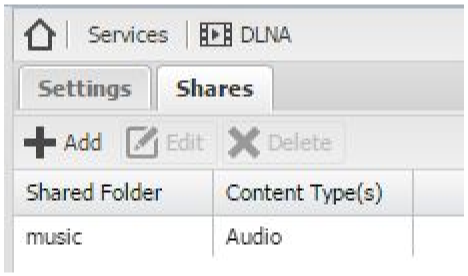
```
/**
 * @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
 * @author    Volker Theile <volker.theile@openmediavault.org>
 * @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
 * @copyright Copyright (c) 2009-2013 Volker Theile
 * @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
// require("js/omv/WorkspaceManager.js")
// require("js/omv/workspace/form/Panel.js")
Ext.define("OMV.module.admin.service.minidlna.Settings", { Define the workspace structure for this extension.
  extend : "OMV.workspace.form.Panel", Extend the workspace using the Panel class extension

  rpcService   : "MiniDlna",  The RPC service class to use. This must be unique.
  rpcGetMethod : "getSettings", The RPC function in the class above to get the settings in config.xml.
  rpcSetMethod : "setSettings", The RPC function in the class above to set the settings in config.xml.
  plugins : [{ Include the plugin linkedfields.
    ptype       : "linkedfields",
    correlations : [{ Look for the items in the getFormItems function below.
      name       : [ The item name object.
        "enable" The item value to match.
      ],
      conditions  : [
        { name : "enable", value : true } Check to see if the plugin is enabled. If it is then enable the rescan button.
      ],
      properties : function(valid, field) { If the plugin is disabled, then set the rescan button as disabled.
        this.setButtonDisabled("rescan", !valid);
      }
    }]
  }],

  getButtonItems : function() { This is the  function for the rescan button.
    var me = this;
    var items = me.callParent(arguments);  This calls the correct function in the main class.
    items.push({ Add these items.
      id      : me.getId() + "-rescan",
      xtype    : "button",
      text     : _("Rescan"),
      icon     : "images/reboot.png",
      iconCls  : Ext.baseCSSPrefix + "btn-icon-16x16",
      disabled : true,
      scope    : me,
      handler  : function() { Do this when the button is pressed.
        // Execute RPC.
        OMV.Rpc.request({ Connect to the RPC service.
          scope       : this,
          callback    : function(id, success, response) {
            this.doReload(); Reload the Webui after completed.
          },
          relayErrors : false,
          rpcData     : {
            service  : "MiniDlna", The RPC service class to use.
```

```
                method   : "doRescan" The RPC function to use in the above class.
              }
            });
          }
        });
        return items;
      },

      getFormItems : function () { This is the function for the form in the main view.
        return [{
          xtype        : "fieldset", This creates the box border around the form items.
          title        : _("General settings"), The name of this box.
          fieldDefaults : {
            labelSeparator : "" This creates a blank space.
          },
          items : [{
            xtype     : "checkbox", The item type.
            name      : "enable", The item name.
            fieldLabel : _("Enable"), The item text to display to the left of the textfield.
            checked    : false The item default value.
          },{
            xtype     : "textfield",
            name      : "name",
            value     : _("MiniDLNA on OpenMediaVault"), The default value in the textfield.
            fieldLabel : _("Name")
          },{
            xtype        : "numberfield",
            name         : "port",
            fieldLabel   : _("Port"),
            vtype        : "port",
            minValue     : 1,
            maxValue     : 65535,
            allowDecimals : false,
            allowBlank   : false,
            value        : 8200
          },{
            xtype     : "checkbox",
            name      : "strict",
            fieldLabel : _("Strict DLNA"),
            boxLabel   : _("Strictly adhere to DLNA standards."), The text to display next to the checkbox.
checked    : false
          },{
            xtype     : "checkbox",
            name      : "tivo",
            fieldLabel : _("TiVo support"),
            checked    : false
          },{
            xtype        : "combo", A combobox containing the options listed below.
            name         : "rootcontainer",
            fieldLabel   : _("Root Container"),
            queryMode    : "local", Query the data in the array listed below.
            store : [
              [ ".", _("Standard") ],
              [ "B", _("Browse") ],
              [ "M", _("Music") ],
              [ "P", _("Pictures") ],
              [ "V", _("Video") ]
            ],
            editable     : false, Users cannot change the data in the combobox.
            triggerAction : "all", Take action on any item selected.
            value        : "."
          },{
            xtype      : "combo",
            name       : "loglevel",
            fieldLabel : _("Log Level"),
            mode       : "local", Same as queryMode above but in a shorter format
            store      : new Ext.data.SimpleStore({ Similar to the store array as above but as a function.
              fields  : [ "value", "text" ],
              data    : [
                [ "off", _("Off") ],
                [ "fatal", _("Fatal") ],
                [ "error", _("Error") ],
                [ "warn", _("Warn") ],
                [ "info", _("Info") ],
                [ "debug", _("Debug") ]
              ]
            }),
            displayField  : "text",
            valueField    : "value",
            allowBlank    : false,
            editable      : false,
            triggerAction : "all",
            value         : "error"
          },{
            xtype      : "textarea", A simple text box.
            name       : "extraoptions",
            fieldLabel : _("Extra options"),
            allowBlank : true
          }]
        }];
      }
});

OMV.WorkspaceManager.registerPanel({ Register this workspace.
      id      : "settings", The id of this workspace.
      path    : "/service/minidlna", The location where this workspace will reside.
```

```
    text      : _("Settings"), The name of this panel.
    position  : 10, The order in which this panel appears.
    className : "OMV.module.admin.service.minidlna.Settings" The class name of this panel.
});
```

## Create the workspace Shares.js

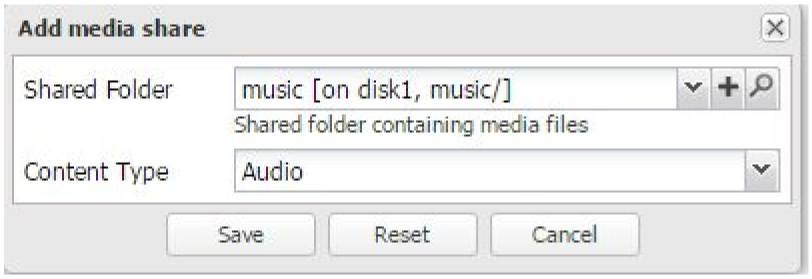
Let's walk through the 2nd panel for setting the shared folders to use with minidlna. This panel uses multiple workspace extensions. They are the grid and window extensions. The grid extension will list the shares that have been created. See figure 6 below that shows that a shared folder with the name music has been set with the content type as audio. This was created using the window extension below.



| Shared Folder | Content Type(s) |
|---|---|
| music | Audio |

Figure 6.

The window extension will create a popup window that allows us to fill out the form for setting the shared folder and the type of media when the add button is clicked. See figure 7 below.



Shared Folder: music [on disk1, music/]
Shared folder containing media files
Content Type: Audio

Figure 7.

Now we will walk through the Shares.js file. We will add comments in the code here to describe what is taking place. You can write the code on your own or use the code on Github located here: https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/var/www/openmediavault/js/omv/module/admin/service/minidlna/Shares.js

Code

```
/**
 * @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
 * @author    Volker Theile <volker.theile@openmediavault.org>
 * @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
 * @copyright Copyright (c) 2009-2013 Volker Theile
 * @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
// require("js/omv/WorkspaceManager.js")
// require("js/omv/workspace/grid/Panel.js")
// require("js/omv/workspace/window/Form.js")
// require("js/omv/workspace/window/plugin/ConfigObject.js")
// require("js/omv/Rpc.js")
// require("js/omv/data/Store.js")
// require("js/omv/data/Model.js")
// require("js/omv/data/proxy/Rpc.js")
// require("js/omv/form/field/SharedFolderComboBox.js")

Ext.define("OMV.module.admin.service.minidlna.Share", { Define the workspace structure for this extension.
  extend : "OMV.workspace.window.Form",  We will use the form extension.
  uses   : [
    "OMV.form.field.SharedFolderComboBox", Display all existing shared folders in a combobox.
    "OMV.workspace.window.plugin.ConfigObject" Automatically load the configuration data/object via the RPC
mechanism based on the given id.
  ],

  rpcService   : "MiniDlna", The RPC class name to use
  rpcGetMethod : "getShare", The get function to use in the above RPC class. This gets the data from config.xml
  rpcSetMethod : "setShare", The set function to use in the above RPC class. This sets the data to config.xml
  plugins      : [{
    ptype : "configobject" Add this plugin to automatically load the data.
  }],

  getFormItems : function () {
    var me = this;
    return [{
      xtype      : "sharedfoldercombo", The form item type to use.
      name       : "sharedfolderref", The form item name.
      fieldLabel : _("Shared Folder"), The form item text to the left of the combobox.
      readOnly   : (me.uuid !== OMV.UUID_UNDEFINED), The uuid must be present. It cannot be undefined.
      plugins    : [{
        ptype : "fieldinfo", Add a text field under the shared folder combobox.
        text  : _("Shared folder containing media files") The text to display in the fieldinfo.
      }]
    },{
      xtype        : "combo", Create a combobox as the form item type.
      name         : "mtype", The form item name.
      fieldLabel   : _("Content Type"), The form item text to the left of the combobox.
      queryMode    : "local", Query the data below.
      store : [ Store the value selected from this array.
        [ "A", _("Audio") ],
        [ "P", _("Images") ],
        [ "V", _("Video") ],
        [ "", _("All media") ]
      ],
      editable     : false,
      triggerAction : "all",
      value        : "" The default value to use which is all media.
    }];
  }
});

Ext.define("OMV.module.admin.service.minidlna.Shares", { Notice this extension name is different. It uses Shares at
the end.
  extend   : "OMV.workspace.grid.Panel", Extend the workspace using the grid extension.
  requires : [ Require the following classes
    "OMV.Rpc",
    "OMV.data.Store",
    "OMV.data.Model",
    "OMV.data.proxy.Rpc"
  ],
  uses     : [ This tells us that this extension uses the window extension.
    "OMV.module.admin.service.minidlna.Share"
  ],

  hidePagingToolbar : false, Do not hide the paging toolbar at the bottom of the grid workspace
  stateful         : true, This will save the state of the grid columns even if they are sorted.
  stateId          : "9889057b-b2c0-4c48-a4c1-8c9b4fb54d7b", The unique id for this object to use for state
management purposes.
  columns          : [{ Create the columns
    text      : _("Shared Folder"), The 1st column text to display.
    sortable  : true,
    dataIndex : "sharedfoldername", The data to index in this column.
    stateId   : "sharedfoldername" The unique identifier for this column.
  },{
    text      : _("Content Type(s)"), The text to display for the 2nd column.
    sortable  : true,
    dataIndex : "mtype", The data to index in this column.
    stateId   : "mtype",  The unique identifier for this column.
  renderer  : function (value) { Display the content type based on the case value.
    var content;
    switch (value) {
    case 'A':
        content = _("Audio");
        break;
    case 'P':
        content = _("Images");
        break;
    case 'V':
        content = _("Video");
        break;
    default:
        content = _("All Media");
        break;
    }
    return content;
  }
}],

initComponent : function () { Initialize the component.
  var me = this;
  Ext.apply(me, { When applied, do this.
    store : Ext.create("OMV.data.Store", { Store the following data.
        autoLoad : true,
        model    : OMV.data.Model.createImplicit({
            idProperty : "uuid", Create a unique id for this data.
            fields     : [ This is the data to store in config.xml.
                { name  : "uuid", type: "string" },
                { name  : "sharedfoldername", type: "string" },
                { name  : "mtype", type: "string" }
            ]
        }),
        proxy    : { Connect to the RPC service
            type    : "rpc",
            rpcData : {
                service : "MiniDlna", The RPC Class to use.
                method  : "getShareList" The RPC function in the class above to use.
            }
        }
    })
  });
  me.callParent(arguments); This calls the correct function in the main class.
},

onAddButton: function () { When the add button is pressed it will create the window.
  var me = this;
  Ext.create("OMV.module.admin.service.minidlna.Share", { Create the window with the settings below.
        title     : _("Add media share"), The text to display for the title of the window.
        uuid      : OMV.UUID_UNDEFINED, This is set to undefined as it is a new entry.
        listeners : {
            scope  : me, The items in this function.
            submit : function () { When you click submit, the data will be parsed, reload the page and show the result.
                this.doReload();
            }
        }
    }).show();
  },

  onEditButton: function () { When the edit button is pressed, create a window
    var me = this;
    var record = me.getSelected(); The item selected from the grid.
    Ext.create("OMV.module.admin.service.minidlna.Share", { Create the window.
        title     : _("Edit media share"), The text to display for the title of the window.
        uuid      : record.get("uuid"), Get the unique id associated with the item.
        listeners : {
            scope  : me, The items in this function.
            submit : function () { When you click submit, the data will be parsed, reload the page and show the result.
                this.doReload();
            }
        }
    }).show();
  },

  doDeletion: function (record) { When the delete button is pressed, create a window to confirm or cancel.
    var me = this;
    OMV.Rpc.request({ Connect to the RPC service.
        scope    : me, The items in this function.
        callback : me.onDeletion, Return after OK or Cancel is selected.
        rpcData  : {
            service : "MiniDlna", The RPC service class to use.
            method  : "deleteShare", The RPC function to use in the above class.
            params  : {
                uuid: record.get("uuid") Delete the item associated with this unique id.
            }
        }
    });
  }
});

OMV.WorkspaceManager.registerPanel({ Register this workspace panel.
  id       : "shares", The ID of this workspace panel.
  path     : "/service/minidlna", The path of this workspace panel.
  text     : _("Shares"), The text to display the name of this workspace panel.
  position  : 20, The position the workspace panel will appear. This will place it 2nd.
    className : "OMV.module.admin.service.minidlna.Shares" The name of this workspace panel class.
});
```

## Checkpoint

You should now have the following folder and files created.
var/www/openmediavault/js/omv/module/admin/service/minidlna
var/www/openmediavault/js/omv/module/admin/service/minidlna/Minidlna.js
var/www/openmediavault/js/omv/module/admin/service/minidlna/Settings.js
var/www/openmediavault/images/minidlna.png
var/www/openmediavault/images/minidlna.svg

## Datamodels

You can check your json code by using an online tool such as https://jsonformatter.curiousconcept.com/ to validate your code.

### Datamodels - Services

Create the datamodels folder located here: usr/share/openmediavault/datamodels
Create a file named conf.service.mindlna.json. If you are creating a plugin for the system instead of a service, replace it with the appropriate name here. In this example, we are creating the plugin under services. You can write the code on your own or use the code on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/datamodels/conf.service.minidlna.json

Code
```
{
        "type": "config", The datamodel type.
        "id": "conf.service.minidlna", The datamodel name. This must be unique.
        "title": "MiniDLNA",
        "queryinfo": {
                "xpath": "//services/minidlna", The xpath location.
                "iterable": false Do not iterate.
        },
        "properties": { These are the items from the Settings.js that we will check the names and values.
                "enable": { The name of the item.
                        "type": "boolean",
                        "default": 0
                },
                "name": {
                        "type": "string",
                        "default": "MiniDLNA Server on OpenMediaVault"
                },
                "port": {
```

```
            "type": "integer",
            "minimum": 1,
            "maximum": 65535,
            "default": 8200
        },
        "strict": {
            "type": "boolean",
            "default": 0
        },
        "tivo": {
            "type": "boolean",
            "default": 0
        },
        "rootcontainer": {
            "type": "string",
            "default": "."
        },
        "loglevel": {
            "type": "string",
            "default": "error"
        },
        "extraoptions": {
            "type": "string"
        },
        "shares": {
```

These are the items from the Share.js that we will check the names and values for the workspace extension window form.

```
            "type": "object",
            "properties": {
                "share": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "uuid": {
                                "type": "string",
                                "format": "uuidv4"
                            },
                            "sharedfolderref": {
                                "type": "string",
                                "format": "uuidv4"
                            },
                            "mtype": {
                                "type": "string"
                            }
                        }
                    }
                }
            }
```

```
                    }
          }
}
```

We need to create an additional datamodel for the workspace grid extension that was created in the Shares.js file. create a file named conf.service.mindlna.share.json. You can write the code on your own or use the code on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/datamodels/conf.service.minidlna.share.json

Code

```
{
          "type": "config",
          "id": "conf.service.minidlna.share", Notice this datamodel id is different.
          "title": "MiniDLNA share",
          "queryinfo": {
                    "xpath": "//services/minidlna/shares/share", The xpath location.
                    "iterable": true, Iterate the data
                    "idproperty": "uuid" Use this id to iterate.
          },
          "properties": { Remember we are looking for the items with the names below as in the
previous datamodel.
                    "uuid": {
                              "type": "string",
                              "format": "uuidv4"
                    },
                    "sharedfolderref": {
                              "type": "string",
                              "format": "uuidv4"
                    },
                    "mtype": {
                              "type": "string",
                              "default": ""
                    }
          }
}
```

## Datamodels - RPC Service

Next we need to create the RPC json which will parse the properties and pass over to the RPC service. In this example, the json checks the parameters of each item and requires a value to be set. This will tell the RPC that the enable button is set to false and the textfeild object will use the default string "Minidlna on OpenMediaVault". The RPC service will then execute the function based on the input provided. Create a file named rpc.minidlna.json. You can write the code on your own or use the code

on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/datamodels/rpc.minidlna.json

Code

```
[{
        "type": "rpc", Notice the datamodel type is set to rpc and not config.
        "id": "rpc.minidlna.setsettings", This is the rpc class function that will set the settings.
        "params": {
                "type": "object",
                "properties": { Remember we are looking for the items and their names.
                  "enable": { The name of the item.
                    "type": "boolean",
                    "required": true This item is required.
                  },
                        "name": {
                          "type": "string",
                    "required": true
                        },
                        "port": {
                          "type": "integer",
                          "minimum": 1,
                                "maximum": 65535,
                    "required": true
                        },
                        "strict": {
                          "type": "boolean",
                    "required": true
                        },
                        "tivo": {
                          "type": "boolean",
                    "required": true
                        },
                        "rootcontainer": {
                          "type": "string",
                                "enum": [ ".", "B", "M", "P", "V" ],
                    "required": true
                        },
                        "loglevel": {
                          "type": "string",
                                "enum":[ "off", "fatal", "error", "warn", "info", "debug" ],
                    "required": true
                        },
                        "extraoptions": {
                          "type": "string",
                    "required": true
                        }
                }
        }
},{
        "type": "rpc",
        "id": "rpc.minidlna.setshare", This is the rpc class function that will set the share settings.
        "params": {
                "type": "object",
                "properties": {
                        "uuid": {
                                "type": "string",
                                "format": "uuidv4",
                                "required": true
                        },
                        "sharedfolderref": {
                                "type": "string",
                                "format": "uuidv4",
                                "required": true
                        },
                        "mtype": {
                          "type": "string",
                          "enum":[ "A", "P", "V", "" ],
                                "required": true
                        }
                }
        }
}]
```

## Checkpoint

You should now have the following folders and files created.
var/www/openmediavault/js/omv/module/admin/service/minidlna
var/www/openmediavault/js/omv/module/admin/service/minidlna/Example.js
var/www/openmediavault/js/omv/module/admin/service/mindlna/Settings.js
var/www/openmediavault/images/minidlna.png
var/www/openmediavault/images/minidlna.svg
usr/share/openmediavault/datamodels
usr/share/openmediavault/datamodels/conf.service.minidlna.json
usr/share/openmediavault/datamodels/conf.service.minidlna.share.json
usr/share/openmediavault/datamodels/rpc.minidlna.json

## RPC Service

In this section, we discuss the PRC service. We register the functions that will be used and they are initialized. When a function is called, it will parse the request. The function getSettings reads the config.xml and returns what is stored. The setSettings function will parse the data passed by rpc.minidlna.json by validating it using the id defined in the json and read the data input from the items. It will then create, modify, or delete based on the data provided. The same will apply for the conf.service.minidlna.share.json and the related functions. Create the folder usr/share/openmediavault/engined/rpc and create a file named minidlna.inc. You can write the code on your own or use the code on Github located here: https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/engined/rpc/minidlna.inc

Code

```
<?php
/**
 * Copyright (C) 2009-2013 Volker Theile <volker.theile@openmediavault.org>
 * Copyright (C) 2013-2016 OpenMediaVault Plugin Developers
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
class OMVRpcServiceMiniDlna extends \OMV\Rpc\ServiceAbstract This is the class OMVRpcServiceMiniDlna.
{
    public function getName()
    {
        return "MiniDLNA"; What's my name?  That's my name! :)
    }
    public function initialize()
    { Register the following functions
        $this->registerMethod("getSettings");
        $this->registerMethod("setSettings");
        $this->registerMethod("getShareList");
        $this->registerMethod("getShare");
        $this->registerMethod("setShare");
        $this->registerMethod("deleteShare");
        $this->registerMethod("doRescan");
    }
    public function getSettings($params, $context)
    {
        // Validate the RPC caller context. You must be logged in as admin to perform this function.
        $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
        // Get the configuration object.
        $db = \OMV\Config\Database::getInstance(); Read the config.xml file
        $object = $db->get("conf.service.minidlna");  Use the json file to gather the data we want.
        // Remove useless properties from the object.
        $object->remove("shares"); We don’t need the data for shares in the json so exclude it.
        return $object->getAssoc(); Show the result in the Webgui.
    }
    public function setSettings($params, $context)
    {
        // Validate the RPC caller context.
        $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
        // Validate the parameters of the RPC service method.
        $this->validateMethodParams($params, "rpc.minidlna.setsettings"); Notice this time we use the rpc json to
validate the data.
        // Get the existing configuration object.
        $db = \OMV\Config\Database::getInstance();
        $object = $db->get("conf.service.minidlna");
        $object->setAssoc($params); This time we set the items from the Webgui
        $db->set($object); Write the items to the config.xml
        // Remove useless properties from the object.
        $object->remove("shares"); We don’t need the data for shares in the json so exclude it.
        // Return the configuration object.
        return $object->getAssoc(); Show the result in the Webgui.
    }
    public function getShareList($params, $context)
    {
        // Validate the RPC caller context.
        $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
        // Validate the parameters of the RPC service method.
        $this->validateMethodParams($params, "rpc.common.getlist"); Validate using a common rpc function getlist.
        // Get the configuration object.
        $db = \OMV\Config\Database::getInstance();
        $objects = $db->get("conf.service.minidlna.share"); Get the data to show in the grid by loading the array below.
        // Add additional share informations.
        $objectsAssoc = []; Load the data as an array.
        foreach ($objects as $objectk => &$objectv) {
            // Add the new property 'sharedfoldername'.
            $objectv->add("sharedfoldername", "string", gettext("n/a"));
            // Get the shared folder configuration object.
            $sfObject = $db->get("conf.system.sharedfolder",
                $objectv->get("sharedfolderref"));
            // Update the 'sharedfoldername' property.
            $objectv->set("sharedfoldername", $sfObject->get("name"));
            $objectsAssoc[] = $objectv->getAssoc();
        }
        // Filter the result.
        return $this->applyFilter($objectsAssoc, $params['start'], $params['limit'],
            $params['sortfield'], $params['sortdir']);
```

```
  }
  public function getShare($params, $context)
  {
    // Validate the RPC caller context.
    $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
    // Validate the parameters of the RPC service method.
    $this->validateMethodParams($params, "rpc.common.objectuuid"); Validate using a common rpc function
objectuuid.
    // Get the configuration object.
    $db = \OMV\Config\Database::getInstance();
    return $db->getAssoc("conf.service.minidlna.share", $params['uuid']); Get the data we want with the unique id.
  }
  public function setShare($params, $context)
  {
    // Validate the RPC caller context.
    $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
    // Validate the parameters of the RPC service method.
    $this->validateMethodParams($params, "rpc.minidlna.setshare"); Validate our items using our rpc json for
setshare.
    // Prepare the configuration object.
    $object = new \OMV\Config\ConfigObject("conf.service.minidlna.share");
    $object->setAssoc($params);
    // Set the configuration object.
    $isNew = $object->isNew(); Check to see if this is a new item
    $db = \OMV\Config\Database::getInstance(); Check the config.xml
    if (TRUE === $isNew) {
      // Check uniqueness - Shared folder
      $db->assertIsUnique($object, "sharedfolderref"); Make sure it has a unique id for the new entry.
    }
    $db->set($object);
    // Return the configuration object.
    return $object->getAssoc();
  }
  public function deleteShare($params, $context)
  {
    // Validate the RPC caller context.
    $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
    // Validate the parameters of the RPC service method.
    $this->validateMethodParams($params, "rpc.common.objectuuid"); Validate the unique id we want to delete
    // Delete the configuration object.
    $db = \OMV\Config\Database::getInstance();
    $object = $db->get("conf.service.minidlna.share", $params['uuid']); Get the entry based on the unique id.
    $db->delete($object); Delete!
    // Return the deleted configuration object.
    return $object->getAssoc(); Return the result in the Webgui
  }
  public function doRescan($params, $context)
  { This function is specifically for the rescan button. It will reload minidlna.
```

```
        // Validate the RPC caller context.
        $this->validateMethodContext($context, ["role" => OMV_ROLE_ADMINISTRATOR]);
        // Get the configuration object.
        $db = \OMV\Config\Database::getInstance();
        $object = $db->get("conf.service.minidlna");
        if (TRUE !== $object->get("enable")) Make sure the plugin is enabled.
            return;
        exec("systemctl force-reload minidlna", $output);
    }
}
```

## Modules

See the description in the getting started section about modules. Create the folder usr/share/openmediavault/engined/module and create a file named minidlna.inc. You can write the code on your own or use the code on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/engined/module/minidlna.inc

Code

```
<?php
/**
 * Copyright (C) 2009-2013 Volker Theile <volker.theile@openmediavault.org>
 * Copyright (C) 2013-2016 OpenMediaVault Plugin Developers
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
class OMVModuleMiniDlna extends \OMV\Engine\Module\ServiceAbstract
    implements \OMV\Engine\Notify\IListener, \OMV\Engine\Module\IServiceStatus
{
    public function getName()
    {
        return "minidlna";
    }
    public function applyConfig()
    { Execute an instance of the shell script in the mkconf folder named minidlna.
        $cmd = new \OMV\System\Process("omv-mkconf", "minidlna");
        $cmd->setRedirect2to1();
        $cmd->execute();
    }
    public function startService()
    { Start mindlna.
        $db = \OMV\Config\Database::getInstance(); Check to see if the plugin is enabled.
        $object = $db->get("conf.service.minidlna");
        if (TRUE !== $object->get("enable"))
            return;
        // Start this service and enable the unit file.
        $systemCtl = new \OMV\System\SystemCtl("minidlna");
        $systemCtl->enable(TRUE);
    }
    public function stopService()
    { Stop mindlna.
        $systemCtl = new \OMV\System\SystemCtl("minidlna");
        $systemCtl->disable(TRUE);
    }
    public function getStatus()
    { Get the status of minidlna. Is it running or not? Return the result.
        $db = \OMV\Config\Database::getInstance();
        $object = $db->get("conf.service.minidlna");
        $systemCtl = new \OMV\System\SystemCtl("minidlna");
        return array(
            "name" => $this->getName(),
            "title" => gettext("MiniDLNA"),
            "enabled" => $object->get("enable"),
            "running" => $systemCtl->isActive()
        );
    }
    final public function onSharedFolder($type, $path, $object)
    {
        $db = \OMV\Config\Database::getInstance();
        if (TRUE === $db->existsByProperty("conf.services.minidlna.share",
          "sharedfolderref", $object['uuid'])) Set a unique id to each entry.
            $this->setDirty();
    }
    public function bindListeners(\OMV\Engine\Notify\Dispatcher $dispatcher)
    { This will notify OMV of changes for each configuration type and the level to notify.
        $dispatcher->addListener(
            OMV_NOTIFY_MODIFY,
            "org.openmediavault.conf.service.minidlna",
            [ $this, "setDirty" ]
        );
        $dispatcher->addListener(
            OMV_NOTIFY_CREATE | OMV_NOTIFY_MODIFY | OMV_NOTIFY_DELETE,
            "org.openmediavault.conf.service.minidlna.share",
            [ $this, "setDirty" ]
        );
        $dispatcher->addListener(
            OMV_NOTIFY_MODIFY,
            "org.openmediavault.conf.system.sharedfolder",
            [ $this, "onSharedFolder" ]);
                        $dispatcher->addListener(
            OMV_NOTIFY_MODIFY,
            "org.openmediavault.conf.system.sharedfolder.privilege",
            [ $this, "onSharedFolder" ]);
    }
}
```

## Checkpoint

You should now have the following folders and files created.
var/www/openmediavault/js/omv/module/admin/service/mindlna
var/www/openmediavault/js/omv/module/admin/service/minidlna/Minidlna.js
var/www/openmediavault/js/omv/module/admin/service/minidlna/Settings.js
var/www/openmediavault/images/minidlna.png
var/www/openmediavault/images/mindlna.svg
usr/share/openmediavault/datamodels
usr/share/openmediavault/datamodels/conf.service.mindlna.json
usr/share/openmediavault/datamodels/conf.service.mindlna.share.json
usr/share/openmediavault/datamodels/rpc.mindlna.json
usr/share/openmediavault/engined/rpc
usr/share/openmediavault/engined/rpc/minidlna.inc
usr/share/openmediavault/engined/module
usr/share/openmediavault/engined/module/minidlna.inc

## MKCONF

See the description in the getting started section about mkconf. Create the folder usr/share/openmediavault/mkconf and create a file named mindlna. You can write the code on your own or use the code on Github located here:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/mkconf/minidlna

Code

```
#!/bin/sh
#
# @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
# @author    Volker Theile <volker.theile@openmediavault.org>
# @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
# @copyright Copyright (c) 2009-2013 Volker Theile
# @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>.

set -e Set attributes and exit immediately if a command exits with a non-zero status.

. /etc/default/openmediavault This loads the default values for OpenMediaVault.
. /usr/share/openmediavault/scripts/helper-functions A set of helper functions.
Define these values.
MINIDLNA_CONFIG="/etc/minidlna.conf"
XPATH="/config/services/minidlna"
Use the helper function omv_config_get to load the data for the value.
log_level=$(omv_config_get "${XPATH}/loglevel")
name=$(omv_config_get "${XPATH}/name")
port=$(omv_config_get "${XPATH}/port")
rootcontainer=$(omv_config_get "${XPATH}/rootcontainer")
strict=$(omv_config_get "${XPATH}/strict")
tivo=$(omv_config_get "${XPATH}/tivo")
extraoptions=$(omv_config_get "${XPATH}/extraoptions")

if [ -n "${rootcontainer}" ]; then
   rootcontainer="root_container=${rootcontainer}"
else
   rootcontainer="#root_container=."
fi
```

```
if [ "${strict}" = "1" ]; then
   strict="yes"
else
   strict="no"
fi

if [ "${tivo}" = "1" ]; then
   tivo="yes"
else
   tivo="no"
fi
```

This section replaces and then writes the config file with the new values. It is recommended that you do not use applications such as sed to replace lines in the config file.

```
# Create minidlna config file
cat <<EOF > ${MINIDLNA_CONFIG}
port=${port}
#network_interface=eth0
friendly_name=${name}
#db_dir=/var/cache/minidlna
#log_dir=/var/log
album_art_names=Cover.jpg/cover.jpg/AlbumArtSmall.jpg/albumartsmall.jpg/AlbumArt.jpg/albumart
.jpg/Album.jpg/album.jpg/Folder.jpg/folder.jpg/Thumb.jpg/thumb.jpg
inotify=yes
enable_tivo=${tivo}
strict_dlna=${strict}
notify_interval=60
serial=31446138
model_number=1
${rootcontainer}
log_level=${log_level}
${extraoptions}
EOF


# Process FTP shares
index=$(omv_config_get_count "${XPATH}/shares/share")
while [ ${index} -gt 0 ]; do

   # Get the shared folder reference and path
   sfref=$(omv_config_get "${XPATH}/shares/share[position()=${index}]/sharedfolderref")
   sfpath=$(omv_get_sharedfolder_path "${sfref}")
   mtype=$(omv_config_get "${XPATH}/shares/share[position()=${index}]/mtype")
```

```
    if [ -n "${mtype}" ]; then
        mtype="${mtype},"
    fi

    if [ -n "${sfpath}" ]; then
        echo "media_dir=${mtype}${sfpath}" >> ${MINIDLNA_CONFIG}
    fi

    index=$(( ${index} - 1 ))
done
```

### Checkpoint

You should now have the following folders and files created.

```
var/www/openmediavault/js/omv/module/admin/service/mindlna
var/www/openmediavault/js/omv/module/admin/service/minidlna/Minidlna.js
var/www/openmediavault/js/omv/module/admin/service/minidlna/Settings.js
var/www/openmediavault/images/minidlna.png
var/www/openmediavault/images/mindlna.svg
usr/share/openmediavault/datamodels
usr/share/openmediavault/datamodels/conf.service.mindlna.json
usr/share/openmediavault/datamodels/conf.service.mindlna.share.json
usr/share/openmediavault/datamodels/rpc.mindlna.json
 usr/share/openmediavault/engined/rpc
 usr/share/openmediavault/engined/rpc/minidlna.inc
 usr/share/openmediavault/engined/module
 usr/share/openmediavault/engined/module/minidlna.inc
usr/share/openmediavault/mkconf
usr/share/openmediavault/mkcconf/minidlna
```

## Packaging your plugin

You should now have all the files and folders in place for you to start packaging your plugin. It is recommended that you read through the introduction to Debian Packaging:
https://wiki.debian.org/IntroDebianPackaging
It will help explain the additional files that are needed and it describes each section. You will now add the debian folder if you have not done so already to your working directory. You will also need to install some packages in your development environment. They are build-essential, devscripts, and debhelper. Install them just like any other package from the command line. Example: apt-get -y install build-essential devscripts debhelper

Next you will start including the files needed for packaging. You can write your own or copy the debian folder on github located here:

https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/tree/master/debian

Start off by creating the changelog by running dch --create -v 3.0-1  This will invoke you to edit the changelog. Change as needed.

Create the debian/compat file. This file will only contain the number 9.

Create the debian/control file. This file describes the source and binary package, and gives some information about them, such as their names, who the package maintainer is, and so on. Note the XB-Plugin-Section: This is where the plugin will be listed in the plugins list on OpenMediaVault.

```
Source: openmediavault-minidlna
Section: net
XB-Plugin-Section: multimedia
Priority: optional
Maintainer: OpenMediaVault Plugin Developers <plugins@omv-extras.org>
Build-Depends: debhelper (>= 9.0.0 )
Standards-Version: 3.9.6
Homepage: http://omv-extras.org/

Package: openmediavault-minidlna
Architecture: all
Depends: minidlna (>= 1.1.2),
        openmediavault (>= 3.0.22),
        ${misc:Depends}
Description: OpenMediaVault miniDLNA (DLNA server) plugin
 lightweight DLNA/UPnP-AV server targeted at embedded systems
 MiniDLNA (aka ReadyDLNA) is server software with the aim of being
 fully compliant with DLNA/UPnP-AV clients.
 .
 The minidlna daemon serves media files (music, pictures, and video)
 to clients on your network.  Example clients include applications
 such as totem and xbmc, and devices such as portable media players,
 smartphones, and televisions.
 .
 MiniDLNA is a simple, lightweight alternative to mediatomb, but has
 fewer features. It does not have a web interface for administration
 and must be configured by editing a text file.
```

Create the debian/copyright file. This file should include any copyrights.

```
Format: http://dep.debian.net/deps/dep5
Upstream-Contact: OpenMediaVault Plugin Developers <plugins@omv-extras.org>
```

```
OMV 3. X  DIY Plugin Guide - Release 1.0
```

```
Copyright: 2013-2016 OpenMediaVault Plugin Developers <plugins@omv-extras.org>
License: GPL-3

Files: /var/www/openmediavault/images/minidlna.png
       /var/www/openmediavault/images/minidlna.svg
Copyright: IcoMoon <http://www.icomoon.io>
IcoMoon <http://www.icomoon.io>
```

Create the debian/rules file. Note that you can add commands to dh command in debhelper to do additional tasks such as in this case to not initialize the application after it is installed. You must also TAB the dh $@ line. Do not use spaces on that line.

```
#!/usr/bin/make -f

%:
	dh $@

override_dh_installinit:
	dh_installinit --no-start --name=minidlna
```

Create the debian/source/format file. You will need to create the source folder and then edit the format file. The file should contain “3.0 (native)” without quotes.

Create the debian/install file. It should only contain the following lines.

```
usr/share/openmediavault/*  usr/share/openmediavault
var/www/openmediavault/*    var/www/openmediavault
```

Create the debian/triggers file. It should contain the line “activate restart-engined” without quotes.

## Postinst Script

Create the debian/postinst file. This is the shell script that will execute after minidlna is installed. You can also test it by running /bin/sh postinst configure.

```
#!/bin/sh
#
# @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
# @author    Volker Theile <volker.theile@openmediavault.org>
# @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
# @copyright Copyright (c) 2009-2013 Volker Theile
# @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>.

set -e

. /etc/default/openmediavault
. /usr/share/openmediavault/scripts/helper-functions

case "$1" in
  configure)

    SERVICE_XPATH_NAME="minidlna"
    SERVICE_XPATH="/config/services/${SERVICE_XPATH_NAME}"

    if [ -z "${2}" ]; then
      deb-systemd-helper disable ${SERVICE_XPATH_NAME}.service >/dev/null || true
      deb-systemd-invoke stop ${SERVICE_XPATH_NAME} >/dev/null || true
    fi

    if ! omv_config_exists "${SERVICE_XPATH}"; then
      omv_config_add_element "/config/services" "${SERVICE_XPATH_NAME}"
      omv_config_add_element "${SERVICE_XPATH}" "enable" "0"
      omv_config_add_element "${SERVICE_XPATH}" "name" "MiniDLNA Server on OpenMediaVault"
      omv_config_add_element "${SERVICE_XPATH}" "port" "8200"
      omv_config_add_element "${SERVICE_XPATH}" "strict" "0"
      omv_config_add_element "${SERVICE_XPATH}" "tivo" "0"
      omv_config_add_element "${SERVICE_XPATH}" "rootcontainer" "."
      omv_config_add_element "${SERVICE_XPATH}" "shares"
      omv_config_add_element "${SERVICE_XPATH}" "loglevel" "error"
      omv_config_add_element "${SERVICE_XPATH}" "extraoptions" ""
    fi

    echo "Add ${SERVICE_XPATH_NAME} user to group: users"
    usermod -G users ${SERVICE_XPATH_NAME}

    if grep -q "fs.inotify.max_user_watches" /etc/sysctl.conf; then
      echo "Increasing max user watches ..."
      echo "fs.inotify.max_user_watches=100000" >> /etc/sysctl.conf
      echo 100000 > /proc/sys/fs/inotify/max_user_watches
    fi
```

```
        dpkg-trigger update-fixperms
        dpkg-trigger update-locale
    ;;

    abort-upgrade|abort-remove|abort-deconfigure)
    ;;

    *)
        echo "postinst called with unknown argument '$1'" >&2
        exit 1
    ;;
esac

#DEBHELPER#

exit 0
```

## Postrm Script

Create the debian/postrm file. This script will uninstall minidlna.

```
#!/bin/sh
#
# @license   http://www.gnu.org/licenses/gpl.html GPL Version 3
# @author    Volker Theile <volker.theile@openmediavault.org>
# @author    OpenMediaVault Plugin Developers <plugins@omv-extras.org>
# @copyright Copyright (c) 2009-2013 Volker Theile
# @copyright Copyright (c) 2013-2016 OpenMediaVault Plugin Developers
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>.

set -e

. /etc/default/openmediavault
. /usr/share/openmediavault/scripts/helper-functions
```

```
case "$1" in
    purge)
        # Remove the configuration data
        omv_config_delete "/config/services/minidlna"
    ;;

    remove|upgrade|failed-upgrade|abort-install|abort-upgrade|disappear)
    ;;

    *)
        echo "postrm called with unknown argument '$1'" >&2
        exit 1
    ;;
esac

exit 0
```

## Final Checkpoint

This is a final list of all the files and folders you should have before building your package.

```
var/www/openmediavault/js/omv/module/admin/service/mindlna
var/www/openmediavault/js/omv/module/admin/service/minidlna/Minidlna.js
var/www/openmediavault/js/omv/module/admin/service/minidlna/Settings.js
var/www/openmediavault/images/minidlna.png
var/www/openmediavault/images/mindlna.svg
usr/share/openmediavault/datamodels
usr/share/openmediavault/datamodels/conf.service.mindlna.json
usr/share/openmediavault/datamodels/conf.service.mindlna.share.json
usr/share/openmediavault/datamodels/rpc.mindlna.json
 usr/share/openmediavault/engined/rpc
 usr/share/openmediavault/engined/rpc/minidlna.inc
 usr/share/openmediavault/engined/module
 usr/share/openmediavault/engined/module/minidlna.inc
usr/share/openmediavault/mkconf
usr/share/openmediavault/mkcconf/minidlna
debian
debian/source
debian/source/format
debian/changelog
debian/compat
debian/control
debian/copyright
```

debian/install
debian/postinst
debian/postrm
debian/rules
debian/triggers

## Build your package

Now it's time for the fun part. Building your package! Use debuild -us -uc and review any error messages you see. If all went well you should see your deb package has been built. Now you can install your package by running dpkg -i minidlna_3.0-1_amd64.deb. Check to see if your plugin is now installed. If it is, congratulations! You have successfully built your 1st plugin. If it did not install, check the error messages. They should indicate where the problem occurred.

## Items not covered

Some items such as adding translations or logging are optional and were not included on this guide. Please view them on github if you are interested in adding them.
Translations:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/tree/master/usr/share/openmediavault/locale

Logging:
https://github.com/OpenMediaVault-Plugin-Developers/openmediavault-minidlna/blob/master/usr/share/openmediavault/engined/inc/90minidlna.inc