Medialon Overture User Manual

Revision 1.5.7

FINAL USER'S LICENSE FOR MEDIALON SOFTWARE PRODUCTS

IMPORTANT - READ ATTENTIVELY: By loading and installing the Software on your computer, you indicate your acceptance of the following Final User's License (the "License Agreement") for Medialon Manager Software, Medialon MXM Plug-In, Medialon Overture Software(s), Medialon Product Browser Software, Medialon Scheduler Software, Medialon Showmaster Editor Software, Medialon MAS Client Software, Medialon SDK and any current or future Medialon Software (the Software) which is either : (i) printed on a license card with the software; (ii) on-line in the software application. If you do not agree to the terms of this License Agreement, for a full refund, promptly return this product to the place you obtained it.

The License Agreement is entered between you (the final user, a legal entity or natural person), and MEDIALON, and concerns the aforementioned Software product, any attached add-in Software, the documentation in electronic format and any example or educational Software (the "Software").

You shall inform all authorized users of the Software of the terms and conditions of this Agreement.

License granting

The Software is protected by Intellectual Property Right, copyright law and by international agreements. Any partial or total reproduction or distribution of the Software, by any means whatsoever, is strictly prohibited. Any use of the demo version for commercial purposes is strictly prohibited. Any person not respecting these provisions shall be guilty of the crime of forgery and shall be liable to the penal penalties provided for by law.

The Software is not sold but transferred under license. You are authorized to install, use, run ONE copy of the Software on ONE computer. If you use a network server, you can install one copy of the Software on it, but you must acquire a user's license for each distinct customer computer using the Software. You are not authorized to use the Software for shared work time or on behalf of a third party.

All other rights not expressly granted in the present contract are reserved by MEDIALON, in particular the present contract grants you no right in relation to the service or trade mark belonging to MEDIALON is the sole holder of the right to make any and all required corrections to the Software in order to comply with the Software documentation.

Save as otherwise provided by law, you are not authorized to reconstitute, reverse engineer, to de-compile or disassemble the Software product.

If the Software product is an update, you must, in order to use it, be the holder of a user's license for the original product. An update replaces the product that gave you the right to the update.

The original holder of the license for the software product is only authorized to transfer this contract once and permanently, to another end user provided that he provides written notice to MEDIALON and the recipient agrees to be bound by and subject to the terms and conditions of this Agreement. This transfer must include all the software elements including without limitation the most recent update and all prior versions, electronic and paper documents and the certificate of authenticity.

Copyright

All ownership rights and copyrights relating to the Software, any attached add-in software, the documentation in electronic or paper format and any other example or educational applications are proprietary of MEDIALON retains all title and ownership of the Software and Software Documentation. All intellectual property rights relating to the APIs and contents to which the product can give access are proprietary of the respective owners of these APIs and contents, and can be protected by regulations and international agreements relating to copyrights and intellectual property rights.

Limited warranty

MEDIALON warrants that the magnetic media on which the Software is recorded and any Software Documentation will be free from defects in material and workmanship under normal use for a period of one (1) year from first date of purchase (date of invoice). MEDIALON also warrants that the original copy of the Software will perform substantially in accordance with the accompanying Software Documentation for one (1) year from the date of receipt. MEDIALON does not warrant that the functions contained in the Software will meet all your requirements or that the operation of the Software will be error free or uninterrupted.

Your exclusive remedy for breach of MEDIALON warranty shall be (i) in case of defects in the media, the replacement by MEDIALON of any magnetic media not meeting the warranty and (ii) in case of any defect in the Software, MEDIALON shall use reasonable efforts to provide maintenance, modifications or fixes in a timely manner, or at its option replace the Software, provided the Software is returned with a copy of your receipt. This limited warranty is void if failure of the Software has resulted from accident, abuse, alteration or misapplication. Any replacement Software will be warranted for the remainder of the original warranty period or two (2) months, whichever is longer.

MEDIALON is not responsible for problems caused by changes in, or modifications to, the operating characteristics of any computer hardware or operating system for which the Software or any upgrade or update is procured, nor is MEDIALON for problems which occurs as a result of the use of the Software in conjunction with software of third parties or with hardware which is incompatible with the operating system for which the Software is being procured.

Infringement

MEDIALON, at its own expense, will indemnify and defend you against any action brought against you to the extent that it is based on a claim that the Software or any upgrade or update of the Software used within the scope of this Agreement infringes any US patent or copyright provided that MEDIALON is promptly notified in writing of such claim. MEDIALON shall have the right to control the defense of all such claims, lawsuits, and other proceedings. In no event may you settle any such claim, lawsuit, or proceeding without MEDIALON's prior written approval. MEDIALON shall have no liability for any claim under this section if a claim for a US patent or copyright infringement is based on the use of a superseded or altered version of the Software if such infringement would have been avoided by use of the latest unaltered version of the Software made available to you, or in the event such claim is based upon any modification or enhancement to the Software made by you or on your behalf. In the event a third party infringement claim is sustained in a final judgment from which no further appeal is taken or possible, or if your use of the Software is enjoined by a court, then MEDIALON shall, in its sole election and at its expense either (i) procure for you the right to continue to use the Software pursuant to this Agreement; (ii) replace or modify the Software to make it non-infringing; or if (i) and (ii) are not reasonably feasible, (iii) terminate this Agreement and refund to you the depreciated value of the Software, based on straight line depreciation over a period of 5 years. The foregoing obligations state MEDIALON's entire liability and your exclusive remedy of and MEDIALON shall have no other liability or obligation with respect to any actual or alleged infringement of any intellectual property rights under this Agreement

Limitation of liability

MEDIALON's entire liability to you or another party for any loss or damage resulting from any claims, demands or actions arising out of this Agreement shall not exceed the license fee paid to MEDIALON for the Software ("License Fee"), net of dealer or distributor margins, notwithstanding any failure of essential purpose of any limited remedy.

No other warranties

Except for the express warranty provided under heading Limited Warranty above, the Software and its related documentation are provided "As Is" and without a warranty of any kind, whether express, implied, statutory and MEDIALON specifically disclaims the implied warranties, terms or conditions of merchantability, non-infringement and fitness for a particular purpose.

No liability for consequential damages

ACCORDING TO DANGEROUSNESS OF CERTAIN KIND OF EQUIPMENTS WHICH CAN BE CONTROLED BY THE SOFTWARE, THE LIABILITY OF MEDIALON IS SUBMITTED TO THE STRICT COMPLIANCE WITH THE MANDATORY SAFETY RULES AS DESCRIBED INTO THE SOFTWARE DOCUMENTATION.

Save as otherwise provided by law, in no event and more particularly in case of breach of the safety rules as described into the Software Documentation, shall MEDIALON be held liable for any special, indirect or accessory damage, of any nature whatsoever, including without limitation body or material injury, loss of profit, interruption of activity, loss of information or other pecuniary losses which may result from the use or the impossibility to use the Software, and this even if the company MEDIALON has been notified of the possibility of such prejudice.

Termination

This Agreement is effective until terminated. This Agreement will terminate if you fail to comply with any provision of the Agreement. Upon termination, you shall destroy all copies of the Software, including security keys and modified copies, if any.

Maintenance and support

Maintenance, including the provision of upgrades and updates to the Software, and telephone support is available from MEDIALON only through a maintenance plan. Updates and upgrades are not available separately.

General provisions

If any term, condition, or provision in this Agreement is found to be invalid, unlawful or unenforceable to any extent, such invalid term, condition or provision will be severed from the remaining terms, conditions and provisions, which will continue to be valid and enforceable to the fullest extent permitted by law.

Any controversy or claim arising out of or relating to this contract, or the breach thereof, shall be settled by arbitration administered by the American Arbitration Association in accordance with its Commercial Arbitration Rules, and judgment on the award rendered by the arbitrator(s) may be entered in any court having jurisdiction thereof.

Table of Contents

1 Overview

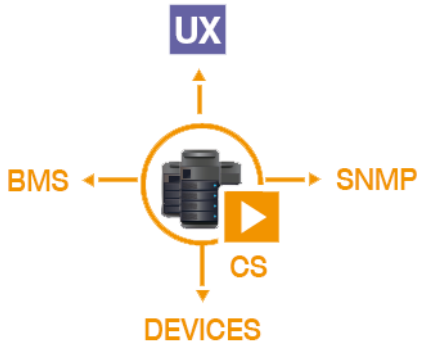
Medialon Overture is an effective enterprise-wide AV control software. It controls, monitors, and automates thousands of AV devices in multiple locations, integrates with IT services, and delivers highly interactive user interfaces.

Medialon Overture, as an Enterprise Class software, is designed to run in a virtualized fault tolerant environment and is made of two main software components:

- Control Server, which controls devices, and
- User Experience Server, which generates and serves all the user interfaces (Help desk, Smartphone, In-room GUI).

A User Experience Server can be connected to one or several Control Servers, depending on the architecture required or the size of the project. A large venue may use one User Experience Server with one Control Server to manage all devices in the same building, while a corporation may use one User Experience Server at its headquarters, connected to 20 Control Servers—one in each branch or country—to centrally manage and control all its AV devices worldwide.

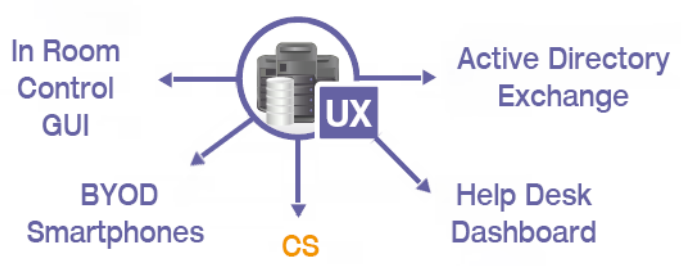
1.1 Control Server



The Control Server controls audiovisual devices via TCP/IP and other protocols using IP to serial or I/O converters and includes drivers for industry standard audiovisual equipment. It interfaces with IT applications and services such as Active Directory Services, Microsoft Exchange, SQL databases, Building Management Systems, and commonly used enterprise-class asset management software.

It includes a graphical driver editing tool, a complete SDK for developing custom plug-ins, and an API for integration with other software. The real-time graphical programming environment (Task Engine) does not require a code/compile/download process, easing automation task programming for operators.

1.2 User Experience(UX) Server



The User Experience Server categorizes all controlled A/V equipment in an infrastructure database by type, location, and an unlimited number of sub-categories and tags. It automatically generates all graphical user interfaces for the help desk, mobile devices, and in-room control interfaces using information from the infrastructure database.

Graphical user interfaces are served as HTML5 and use CSS to manage look and feel, delivering a consistent user experience throughout the system, anywhere in the world, supporting multiple language versions. The help desk dashboard displays live data about device status, such as video projector lamp usage, rack room temperatures, or device failures, while the map interface uses a familiar graphical building plan model to geographically access remote room control interfaces as well as device control panels.

The UX Server consists of two major GUIs:

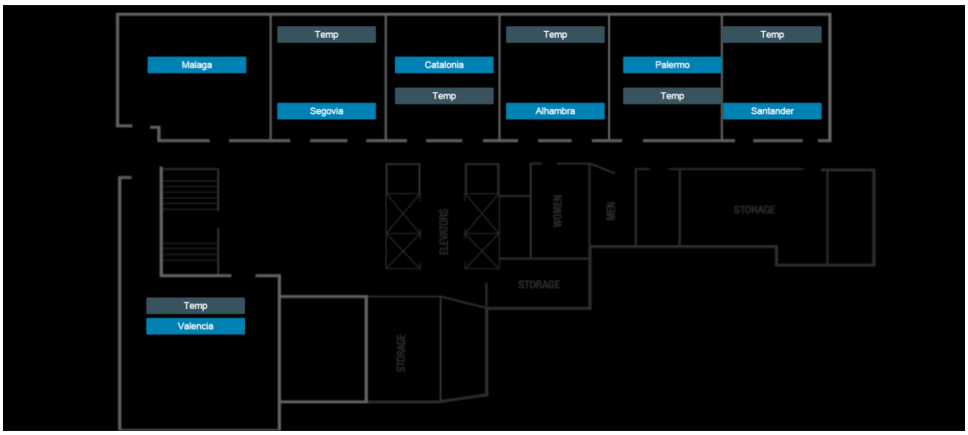
- Dashboard is the main GUI interface for users. It houses facility **maps**, room or device based control **panels**, and front-end **widgets** for viewing helpful details at a moment's notice.
- Magic Menu is the mobile device GUI. It consists of multiple **pages** used for specific room or device based controls.

1.2.1 Dashboard

Dashboard is the main GUI in the Overture system. You can access the dashboard system by opening up a web browser and typing the IP address and port of your UX HTTP server. (Default: http://localhost:80/)

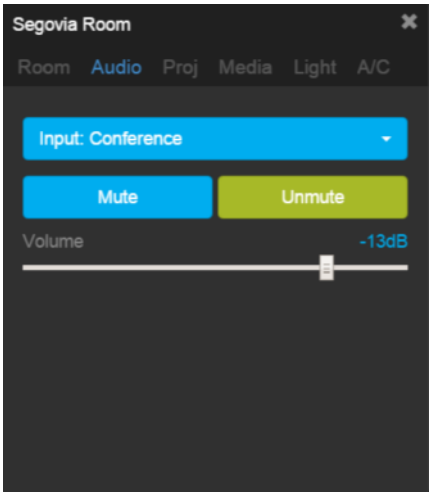
Your Dashboard will consist of four main components to help you monitor and control devices: Maps, Control Panels, Widgets, and Logs

1.2.1.1 Maps



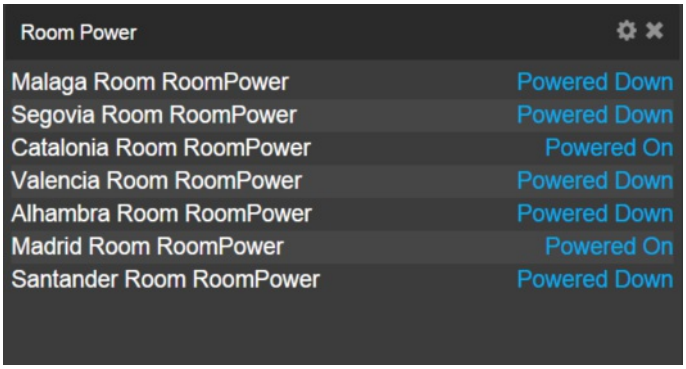
Maps are images that are displayed within Dashboard. Inside of a map, you can create links to control panels or other maps for a layered monitoring service.

1.2.1.2 Control Panels



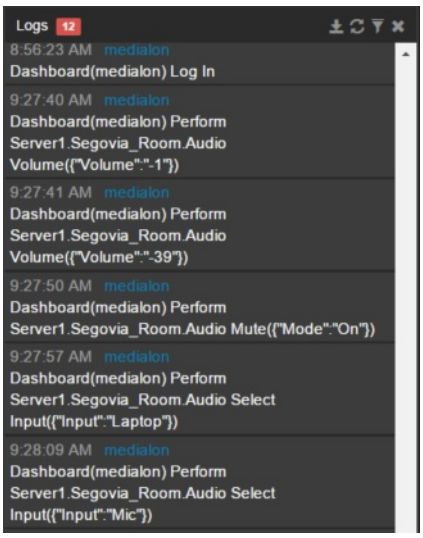
Control panels are HTML pages displayed within Dashboard. Inside of control panels is where you will interact and monitor your devices directly.

1.2.1.3 Widgets



Widgets give Dashboard a quick view of information on your system. You can add graphs to track temperature, tables to display occupancy, or add bookmarks to open layered panels in one click.

1.2.1.4 Logs



Logs inside of Dashboard give detailed accounts of how devices inside your system are being used and what each user is doing within the system.

1.2.2 Magic Menu

(This is an example of a Magic Menu page)

Magic Menu is the second GUI in the Overture system. This is used for mobile and tablet devices. Access it similar to Dashboard by opening a browser and going to the URL of your UX Server and adding '/magicmenu'. (Default: http://localhost/magicmenu)

Magic Menu **pages**, like Dashboard control **panels**, are written HTML pages that provide a way to monitor or control devices on the Overture system. They can also be used to provide links to other **pages**, giving your user-interface an organized layout.

With Overture, the HTML pages you write can function as both a Magic Menu **page** and a Dashboard **panel** if you need it to!

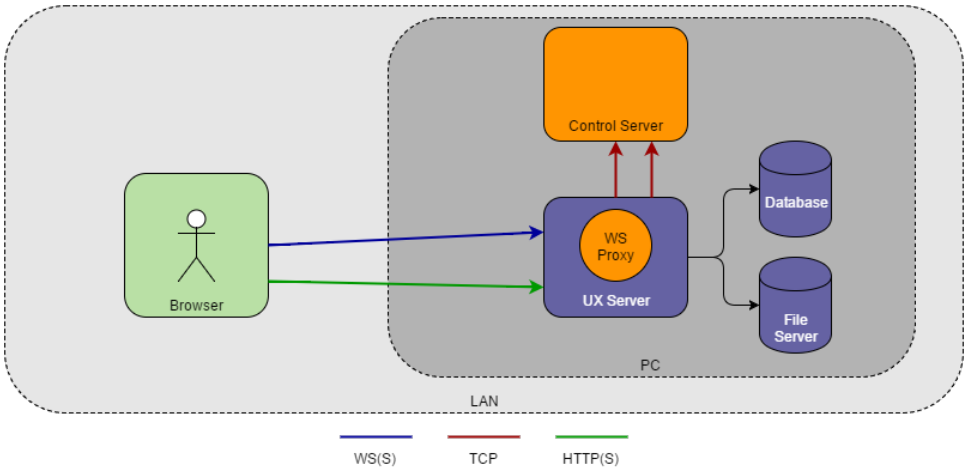
2 Topologies

As an enterprise class software, Overture can be installed following different topologies in one or several buildings, depending on the LAN / WAN configuration and IT requirements.

Each topology will use the same components just in a different configuration.

- **Browser:** This is the web browser, whether it is on a PC or mobile device, that connects to UX Server and displays dynamically created content to the user.
- **Control Server:** Control Server sends devices commands, returns important information back, and tells the browser information regarding those devices.
- **Database:** This is part of the UX Server. It holds all the information from your project including metadata about the Control Server points.
- **File Server:** Another part of the UX Server. This holds HTML documents, CSS styles, images, and other important files related to the project.
- **WS Proxy:** This component converts WebSocket or WebSocket Secure messages from the browser to the TCP for the Control Server to understand. It can be part of the UX Server or the Control Server depending on your need.

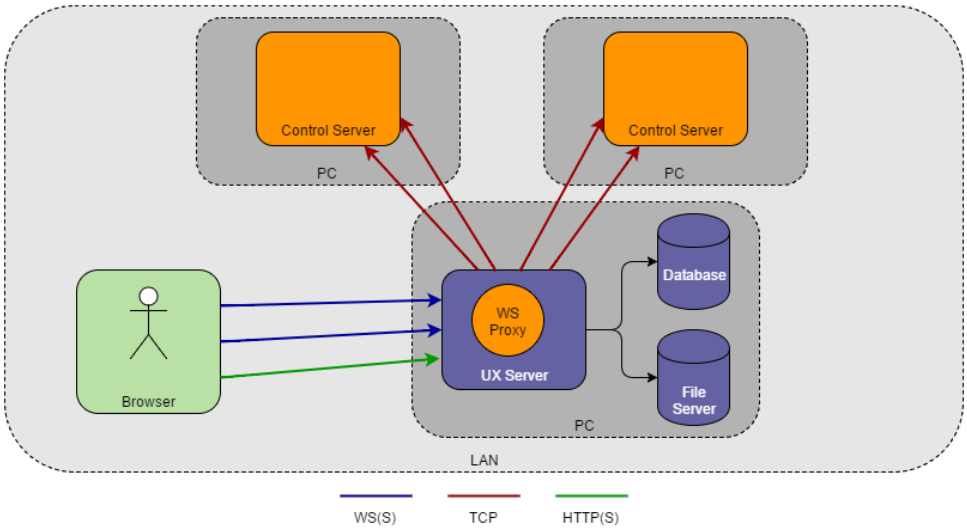
2.1 Single Controller/All-In-One



In this topology, you have one Control Server and it is installed on the same machine as the the UX Server.

The browser talks directly to the UX Server via HTTP(S) to get HTML pages served to it. The browser also talks WS(S) to the WS Proxy, which is hosted by the UX Server. The UX Server converts this to TCP and talks to the Control Server. The UX Server also establishes it's own TCP connection to the Control Server to talk directly to it.

2.2 Multi-Controller/UX Server On Same LAN

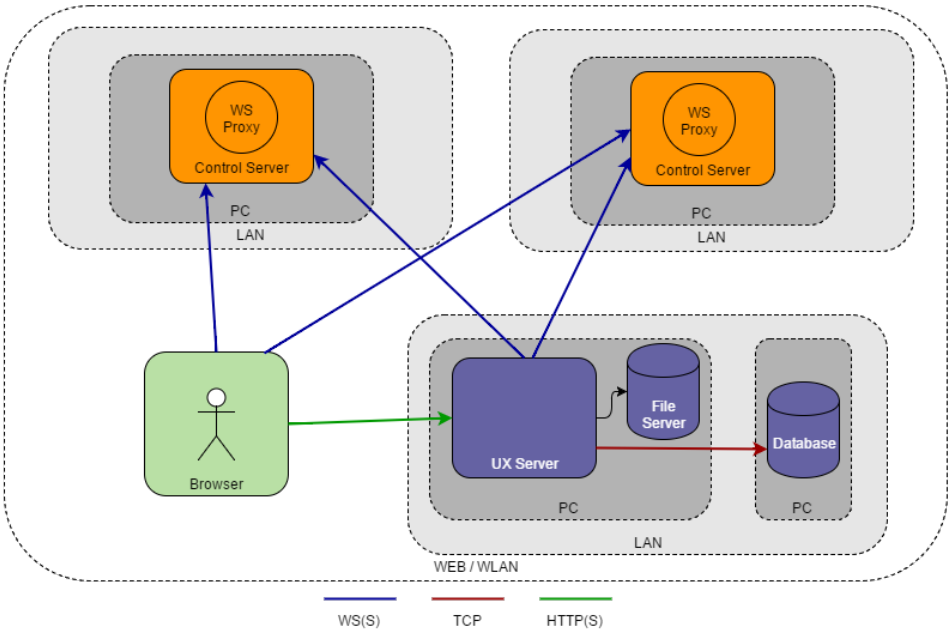


In this topology, you have multiple Control Servers. The UX Server is installed either on its own machine or on the same machine as one of the Control Servers. An example of this is when you have a control server in one building, another Control Server in a different building, and the UX Server installed in a third building all on the same LAN.

Like the previous topology, the browser talks to the WS Proxy, which then talks to the Control Server. The browser opens up a connection to the proxy for each Control Server. This means that the number of connections to the WS Proxy are equal to the number of browsers connected times the number of Control Servers present.

2.3 Multi-Controller With Proxy On Control Server And Database

Hosted Seperately



In this topology, you have multiple Control Servers and the UX Server each on seperate LANs. The database is also on it's own PC. An example of this is when you have a Control Server in New York, another Control Server in Los Angeles, and the UX Server installed in Miami. The UX is connecting to a pre-exisiting database not installed by Overture.

This example shows how the WS Proxy can be hosted by the Control Servers. In that scenario, the browser talks WS(S) directly to the Control Server and the WS Proxy to translate it to TCP. The UX Server establishes its own WS(S) connection directly to the Control Servers, which is also translated by the WS Proxies. The UX Server also communicates to the database the way it always has, but now that connection is to a seperate PC instead of the database being hosted on the same PC.

2.4 Summary

The three topologies above are just examples of ways the UX Server and Control Server can be configured to talk to each other and the browser. By pulling out individual components of the UX Server, allowing them to run as independent modules, as well as deciding where the WS Proxy will be hosted you can configure Overture based on your network needs.

3 Setup

3.1 Requirements

UX Server should be installed on a PC with the following requirements:

- Window 7 Pro (32 or 64) or higher
- Intel Core i3–2xxx or higher / AMD Athlon II X4 651 or higher
- 4GB RAM or higher
- 256GB SATA 2 SSD/HDD or higher
- 1 Gigabit Ethernet
- 1 USB 2.0 Ports

3.2 Installing UX Server

This section will guide you through the UX Server install and explain the initial options for your installation :

First open up the installer and click 'Install Medialon Overture UX'. The installer will open and welcome you to the installation.

Next, you will need to read through the License Agreement and accept the terms.

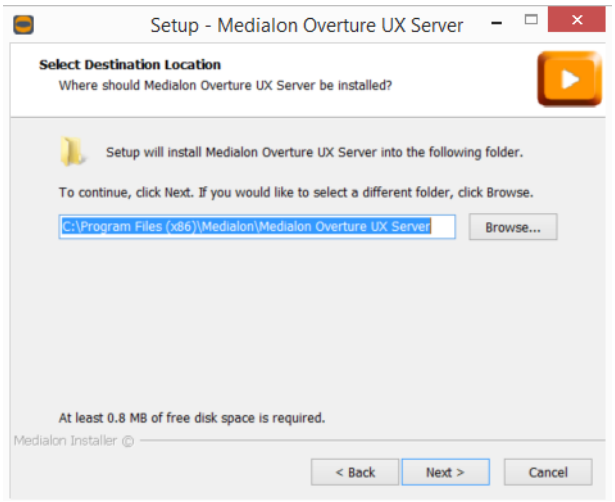
If you are just updating an existing UX Server to a newer version, select update instead of reinstall. This will preserve all of your existing data.

If you are re-installing or starting with a brand new installation you will have two options. Install automatically with default options, or advanced installation.

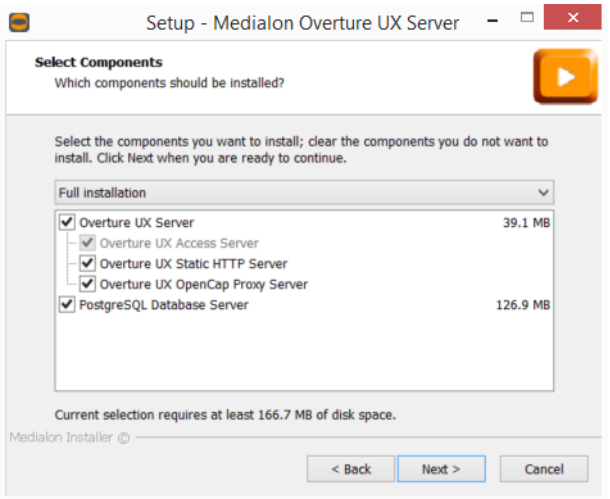
If you using the simple mode, the following options will be your defaults:

- **Installation Folder:** C:\Program Files (x86)\Medialon\Medialon Overture UX Server
- **File Server (Assets):** C:\ProgramData\Medialon\OvertureUX
- **PostgreSQL Server:** Will be installed
- **PostgreSQL Server Location:** 127.0.0.1
- **PostgreSQL Server Admin User/Password:** poadmin/medialon
- **Database Name:** ovtmap
- **Database User/Password:** ovtuser/medialon
- **HTTP/HTTPs:** HTTP
- **HTTP Port / HTTPs port:** 80/443

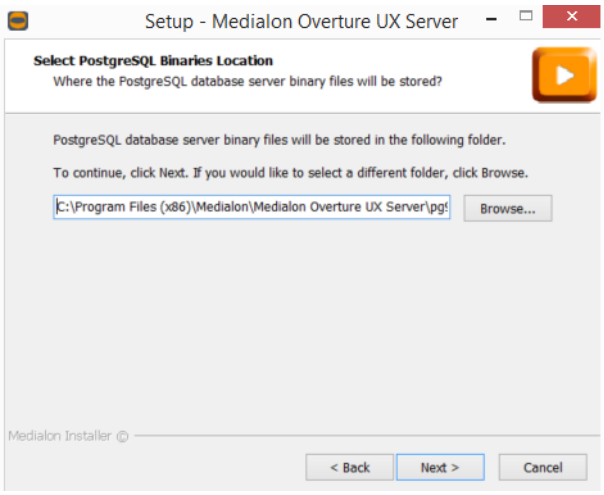
If you are using the advanced options, the following items will assist with what each option means.



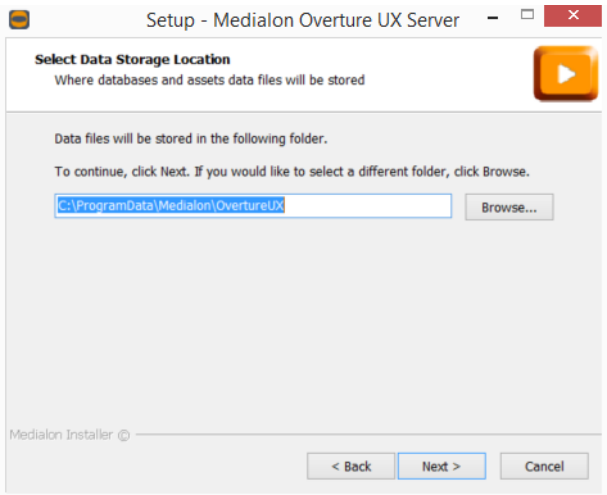
You can define the path where Overture UX Server will be installed. (32-bit Default Location: "C:\Program Files\Medialon\Medialon Overture Ux Server", 64-bit Default Location: "C:\Program Files (x86)\Medialon\Medialon Overture Ux Server")



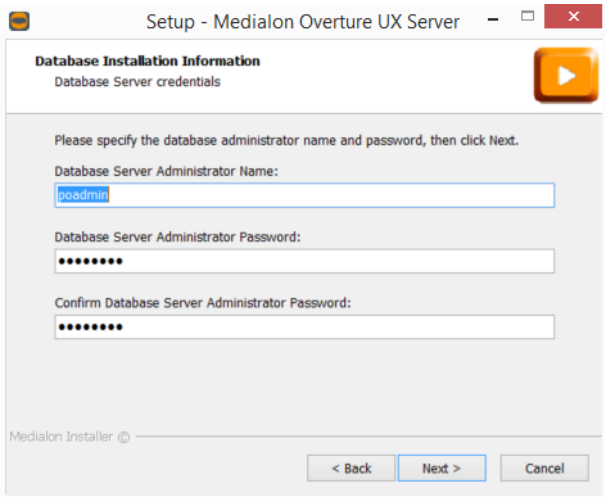
Next, you can define what parts of the UX Server are installed on this machine. Change this only if you are using a more advanced topology that separates out the HTTP Service or the database.



Next, if you are installing the PostgreSQL Database, you can define the path of where it will install the binaries.

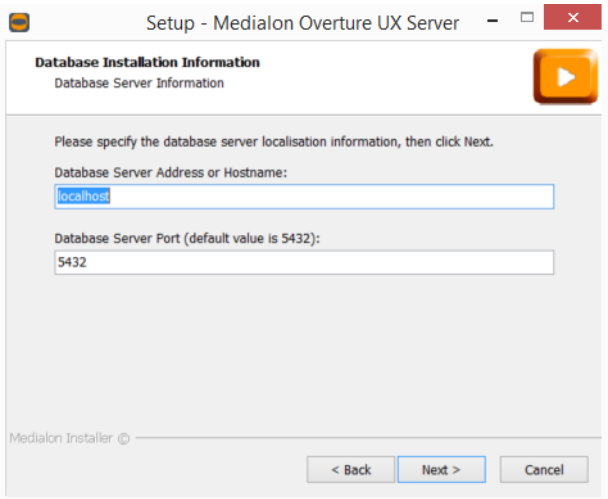


Next, you can define the path where UX Server will store its data. This will be where your assets folder is located, which is used for storing the images and HTML files for your Overture system.

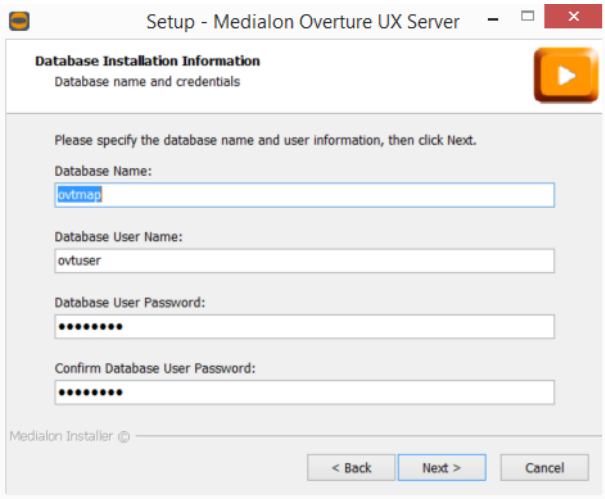


Next, if you are installing the PostgreSQL server on your machine, you will need to setup credentials for that server. This account (user: poadmin / password: medialon by default) is only used by the person in charge of performing maintenance on the database (Such as creating backups, updating the database structure, etc.). **Note:** PostgreSQL database names, usernames, and passwords must be alphanumerical and lowercase.

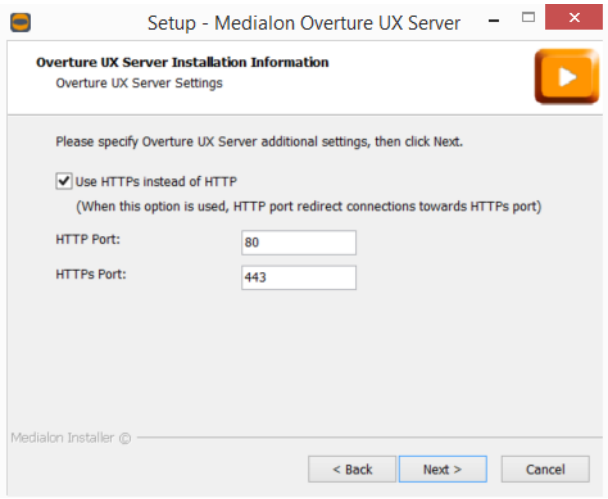
If you are not installing the database on your machine you will instead see:



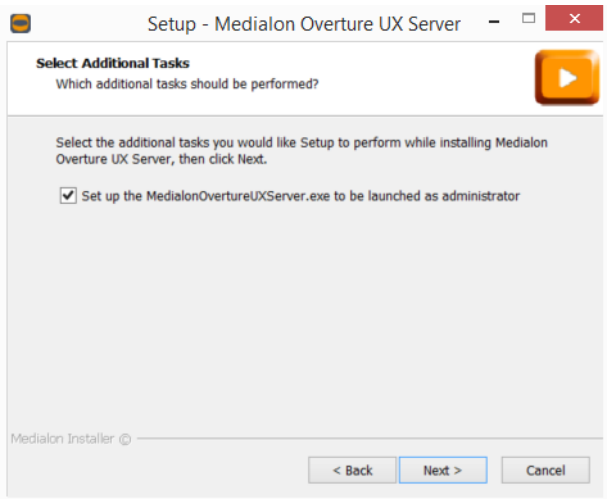
If you are not installing the PostgreSQL server on your machine, you will need to specify the IP address and port of the database server you will use.



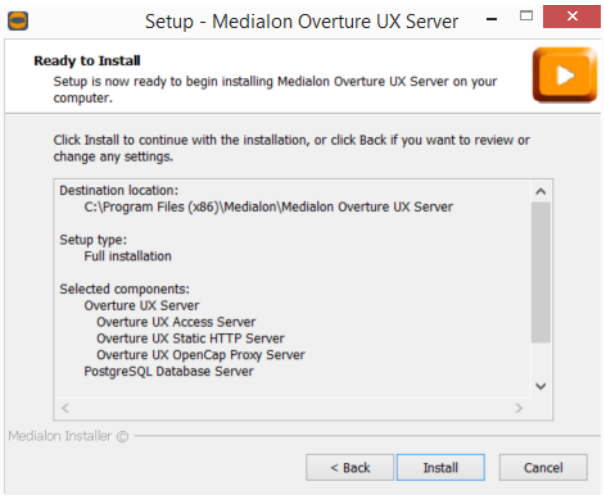
Next, you will need to set-up the credentials of the database. This name and account (username ovtuser/ password: medialon by default) are specific to your Overture installation and are used only for the maintenance of the database. **Note:** If you are installing multiple databases for different UX Projects on the same server you will need to change the database name or your installation will not complete properly.



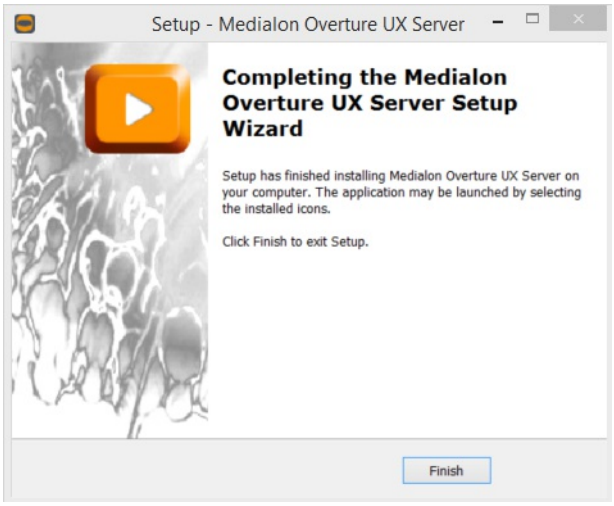
Next, you can define whether to use HTTP or HTTPS and which ports to use.



Next, you can define whether or not to run UX Server as an administrator by default. Due to Windows UAC settings, it is recommended this box is checked.



Once you have confirmed your settings, click 'Install'. If there are any settings you would like to change you can go back and change them now before the installation starts.



You have successfully installed UX Server. Click 'Finish'.

3.3 Next Steps

Overture was designed with a specific work flow in mind:

- Program your Control Server(s).
- Ingest your Control Server project(s) into your database with **UX Configurator**.
- Add or edit database items with **UX Configurator**
- Write your HTML panel/pages using **GUI Editor**.

4 Control Server

This section refers only to Control Server as it relates to UX Server and Overture. For a more detailed guide to programming your Control Server project please see the 'Control Server User Guide'.

While creating your Control Server project it is important to remember to create a link to your UX Server.

To do this you will need to create and share a user group. You can name the user group however you wish, the easiest is 'Shared'. Make sure the 'Shared' property is checked.

Inside that user group you will need to create three separate groups: one for devices, one for tasks, and one for variables you wish to share with the UX Server.

Once those groups are created, drag devices, tasks, or variables you want the UX Server to interact with into its respective folder.

UX Configurator has the ability to read the shared user group in your project and will automatically set up your initial database based on the items in that group. This will save you time on your project as you will not have to add each item into the database manually!

5 UX Server

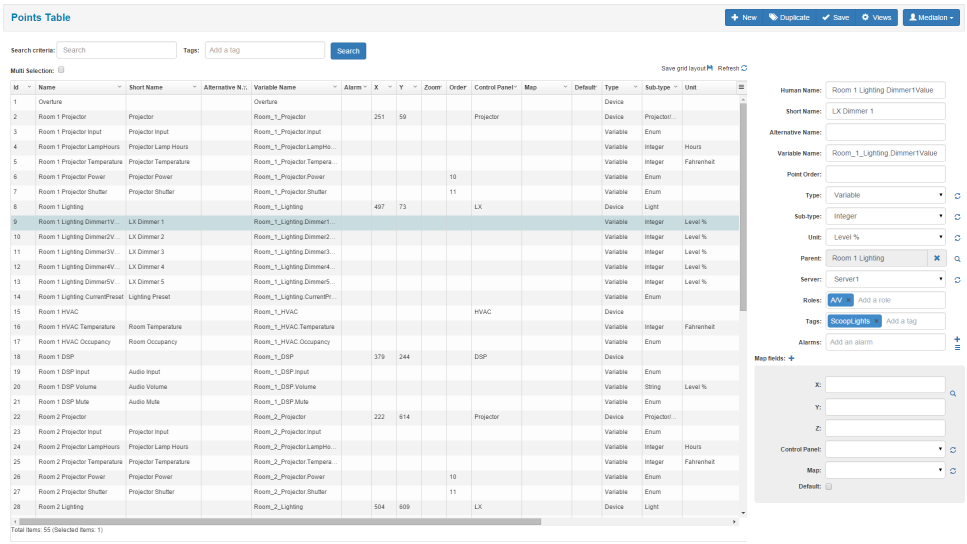
The UX Server is how we will add metadata and dynamically display information from Control Server to the GUIs of users on the system.

The administration interface of UX Server includes two main tools:

Configurator is the component that ingests a Control Server project, adds its information to a database, and sets up unique metadata to each item. It is also used to establish access rights for your project so only certain users can see or control the items you want them to.

GUI Editor is the programming interface for the UX Server. It's how you create HTML pages that will show users information from Control Server along with the metadata created inside of UX Configurator.

5.1 UX Configurator



Configurator is the tool for interfacing with five parts of the Overture system:

- **UX Settings:** The settings for your HTTP Server, database, and Control Servers on the system.
- **Points:** The various devices, tasks, variables across Overture that you can monitor or control.
- **Access Rights:** The Overture system's users, groups, and roles that control what users can see and control on the system.
- **Assets:** The panels and maps that the dashboard displays and provides links to.
- **Alarms:** Notifications set up in the system that monitor and alert users of abnormal operating events.

The Configurator can be accessed through the Overture Dashboard by clicking on the user menu (top-right portion of the screen) or by going directly to its URL inside of a browser. (Default: http://localhost/configurator)

You need to log in to the system using your user credentials. If this is your first time logging in, use the default user name and password. (Default username/password: medialon/medialon)
It is highly recommended you change the password for this user once you have set up the system.

Inside Configurator, the top of the screen is the main control bar. This is where most interaction with items occurs. Each view inside Configurator uses the control bar to create new items, save changes, or change which view of items you are looking at. Each view also has a search bar that helps to find items when there are a lot of items in the database.

The User Menu displays an overview of the currently logged in user's roles, links to user interfaces and administration tools within Overture, and tells you what version of Configurator you are using.

The Views Menu is how you navigate to the management tools for each of the various types of items in the Configurator.

5.1.1 UX Settings

UX has many settings that can be configured to fit the needs of your install and topology. This can be changing the HTTP server, adding another Control Server to the project, or just ingesting all of the points from a Control Server project.

The UX Server settings are initially set during installation of the software. By clicking 'UX Server Config' inside of 'Views' you can look at or edit any of these initial settings.

Inside of 'UX Server Config', you can add any HTTPS key or cert files needed to provide browsers with a trusted link to your server.

You will only need to change these settings for advanced topologies or changing how your browser communicates with the HTTP Server.

5.1.1.1 Control Servers

Clicking 'Control Servers' under 'Views' allows you look at or add Control Servers to your Overture system. Overture comes with 'Server1' listed as default. If you need to add another Control Server you can click 'New' at the top. Give your Control Server a name, IP address, port (9255 is default), and the name of the user group where your shared items are stored.

If you are installing the Control Server WS Proxy seperately, please specify the IP address of the proxy instead

of the Control Server IP address.

5.1.1.2 Ingesting A Project

Overture allows you to ingest your Control Server project, which automatically fills the database with points based on what you have programmed.

Overture UX's ingest engine reads the XML data of the project file, creating a point for each shared device, variable, and task. The information is not retrieved from the Control Server in real time; the ingest must be manually initiated using the CS project file.

Ingesting automatically adds as much metadata about the points as it can. For example, Devices are given Device type, Variables are given Variable type and specific sub-type of Integer, Real, Enum, etc. Device Variables are assigned the corresponding Device point as their parent. By default, points that are associated with the selected Server but do not exist in the ingested CS project file are erased during the ingest process. This helps to clean up the database when re-ingesting projects where devices, tasks, or variables have been deleted.

To ingest a project, select the Control Server that the project is running on from the drop down.

Next, click the "Upload Overture CS Project…" button. Find your project file and click open. Then click the "Update Database" button to the right. The ingest engine uploads the project file, extracts the information from the Shared user group, and updates the database, adding or deleting points as needed.

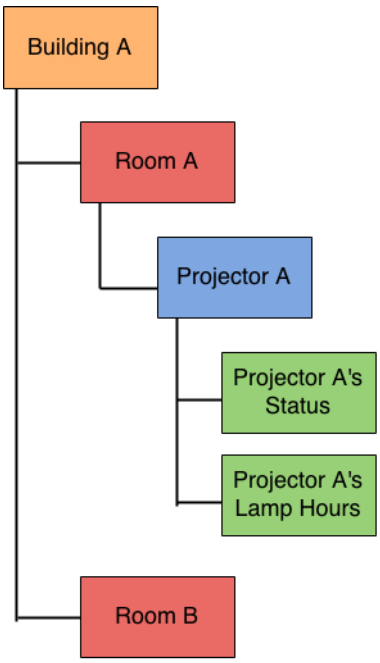
If you need to make changes to the Control Server project, such as adding a variable, you can ingest the project again and the database updates, adds, or deletes points accordingly.

Note: If you do not want the ingest engine to automatically delete points associated with variables/devices/tasks that are no longer shared in the CS project, you can modify the default behavior in the 'UX Server Config' settings.

5.1.2 Points

A point is an item that can be monitored and controlled by Overture. This includes Control Server devices, tasks, and variables. A point can also be a virtual item that isn't directly related to a Control Server, like a map or a control panel in Dashboard. Points have a number of attributes related to them and also act in a parent/child/sibling relationship between each other.

5.1.2.1 Parent-Child-Sibling Relationships



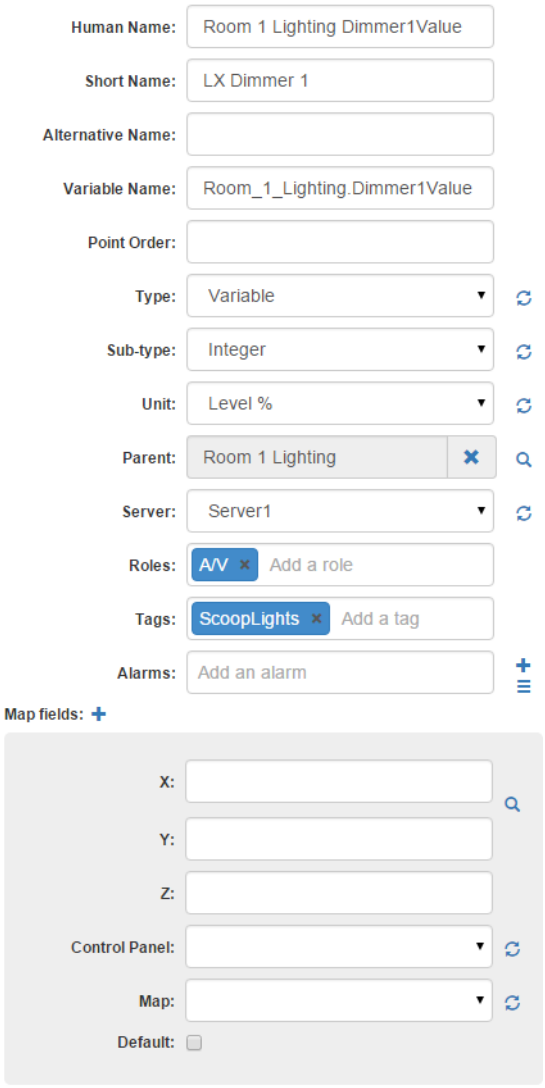
One of the main features of the Overture system is how points relate to each other. A point can be defined as parent of another point. In the above example, a projector's status and lamp hour variables are children of that device, the projector device is a child of Room A. Both Room A and Room B are children of Building A. When ingesting a Control Server project, the Configurator automatically assigns shared variables of a device as children to that device.

Another example is defining a map's point as a parent to a control panel point. In this case, the control panel will now appear on that map.

It is important to understand this relationship when building interfaces and for advanced programming techniques.

5.1.2.2 Point Manipulation

Points can be added, deleted, or edited in the 'Points' section of 'Views'. During your programming you may need to add points not related to your Control Servers. You can also create a point manually to link to a Control Server if you did not want to re-ingest a project. Click 'New' to add a point, and make sure to click 'Save' when you are done or else the point is not created.



If you need to edit an existing point, there are a few tools in this view to help aid you. You can search for points by typing anything related to the point in the search bar. For example if you wanted to find all points that were devices, type 'Device', or if you wanted all points that are children of 'Room A', type 'Room A'. This helps when you have hundreds of points on your system.

You can select multiple points at once by enabling the 'Multi Selection' check box underneath the search bar. With multiple points selected, you can edit them all at once for a speedy option when many points share some qualities.

You can sort the view of your points by clicking on the column you'd like to sort by. You can hide or show columns of points by selecting the arrow below the refresh icon. This allows you to hide attributes you don't wish to see.

You can also delete points you no longer need by clicking the trash can icon on the rightmost column.

5.1.2.3 Point Attributes

- **Id:** The unique id of a point inside the database. Can be used as a reference when programming where the exact id is needed. This is not editable.
- **Human Name:** The name your point is displayed as anywhere in Overture.
- **Short Name:** Short name is used for helping create labels. Example: 'Projector A's Status' short name might be 'Status' for an easy label inside of a panel or page.
- **Alternative Name:** Alternative name is used for the secondary language of the system.
- **Variable Name:** This is the link between the point and the control server. **It is extremely important. Changing this will de-link the point from the Control Server.**
- **Point-Order:** Used for changing view order of a point when multiple points are selected within the same display, can also be used in the dashboard navigation panel. If not specified ID is used for deterining order.
- **Type:** Defines whether the point is a variable, device, room, etc. Used to group points and help search for them.
- **Sub-type:** Further distinction of the type. Example: integer, enum, etc. Only used for grouping and searching.
- **Unit:** Used to create suffix added to the end of a point's value when it's displayed to provide context to numbers. Example: db, %, etc.
- **Parent:** The parent of the point. Used to define inheritance, or show assets on a map.
- **Server:** The link to what Control Server the point is stored on if it is tied to one.
- **Roles:** The access rights associated with the point. Example: If a point has a 'Security' role, only users who have access to 'Security' are able to see it displayed.
- **Tags:** Advanced grouping and search method for when multiple points share something in common but not a type or sub-type. Example: An 'Audio' tag might be created and assigned to both an audio volume point, and an audio source point. Use of tags allow you to program panels / pages more effectively.
- **Alarms** Triggered events that are displayed in the system when this point is behaving abnormally.

If a specific type, sub-type, unit, or tag for a point is not in the system you can create by going into 'Views' and clicking on that item.

Maps

To display a map inside of Dashboard, you must first upload the image you plan to use. To do this, go to the 'Asssets' view.

The 'Assets' view shows a list of selectable items that can be either Maps or Control Panels. To make a new item, click 'New'. If the item you want to use is in your assets folder already, use the magfyining glass to select it. If not, check the `Show Upload Section`. Specify the sub-folder in your assets to upload the item to, and then click 'Upload Asset'.

Once uploaded or selected, make sure the type is an image and save.

In 'Points' view, create a new point. This point will be the Map you wish to display.It can represent a building, floor, site, or whatever you need for your installation.
Under 'Map Fields', select the new asset in the 'Map' drop down menu. Click 'Save'.

If a map has no parent, it will be the first map displayed on Dashboard. If multiple maps have no parents, the Dashboard displays the map with the lowest ID first. If you would like multiple maps with no parents, but would like to define which map gets shown first you can select the 'Default' check box and that map will be the first displayed.

Assigning Control Panels

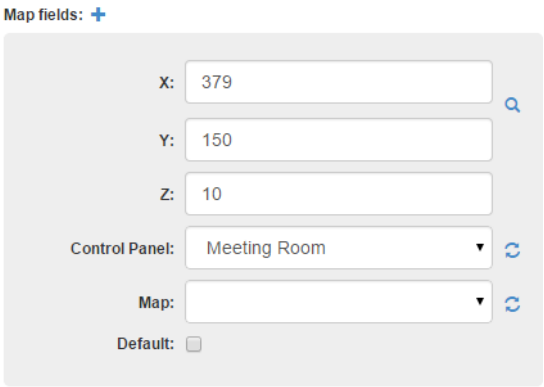
Points can have Control Panels assigned to them. These Control Panels will be how your points will be displayed in Overture. By default, a list of basic templates are provided.

To select one of these panels, choose the point you want to assign a control panel to. This point can be a device, or a virtual point like a room. Under 'Map Fields' select the Control Panel you wish to assign to the point. Click `Save`.

Exisiting Control Panels can be edited, or you can create your own depending on your needs. See: GUI Editor

5.1.2.4 Map Fields

A point is only visible on a map if the point's parent is a map or has a map in it's heirarchy. Once visible, Map Fields tell Dashboard where to display the point on the parent map. This usually applies to virtual points that are maps or control panels, but it is important to know that a point can be a device in a Control Server and a map or control panel on the dashboard.



You can either define a point's x and y value on the map or you can click the magnifying glass icon next to the 'x' field. This will allow you to see the parent map and select the x and y values by clicking anywhere on the map.

You can also define a z access for the point. This allows the point to only show up on the map as you zoom in. This is useful when you have a lot of points in a small area. The higher the number the more zoomed in you must be.

NOTE: A point can be both a Map and a Control Panel. In this case, clicking the name will display the Control Panel while clocking the icon will display the map.

5.1.3 Access Rights

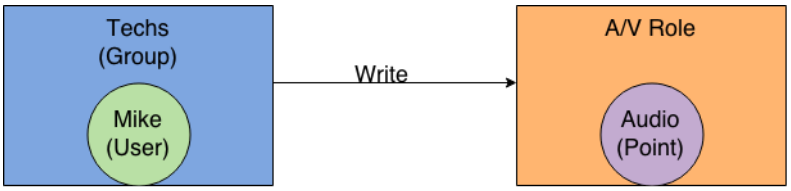
Overture's built in access rights allow various level of controls to help both security and organization of users on

the system.

Access rights have three major components:

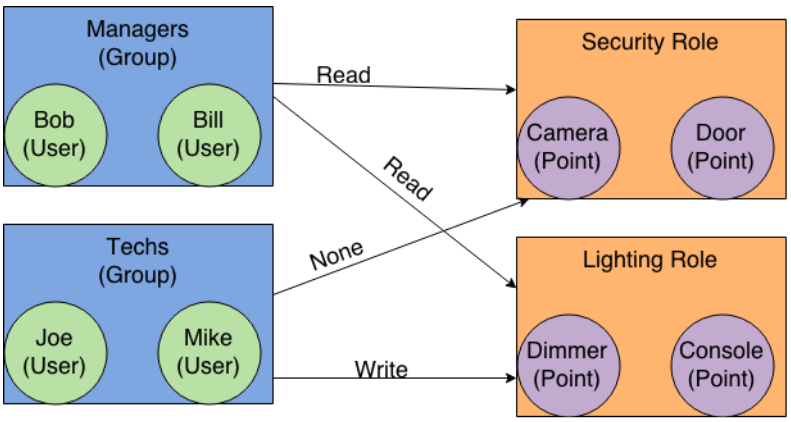
- **Users** are the people on your system. You can give each person who will access your system a seperate username and password to login.
- **Groups** are groups of users. Rather than needing to assign read/write privileges for each user, Overture assigns those privileges to each group of users.
- **Roles** are definitions, assigned to points in the system, that control whether that point is read/writeable by a group.

5.1.3.1 Basic Rights



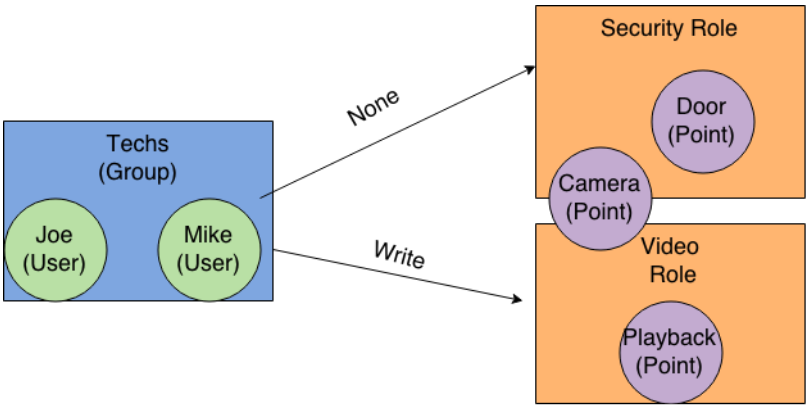
This is a basic example of access rights in the Overture system. Mike is a user who is assigned the Techs group. He will be able to control the Audio point because the Techs group has Write access to the A/V role.

5.1.3.2 Multiple Users/Groups/Roles



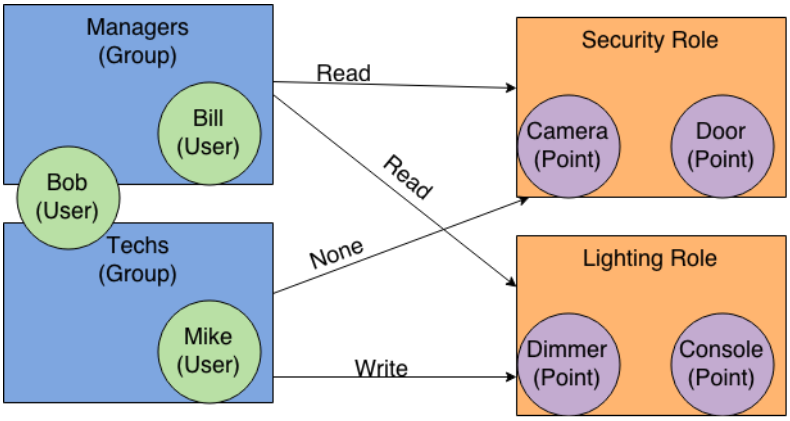
This more advanced example shows multiple groups and multiple roles on the system. Bob and Bill, who are both in the Managers group, can see points that have the Security and Lighting roles but cannot control them. Joe and Mike, both in the Techs group, can control points with the Lighting role but cannot see points with the Security role.

5.1.3.3 One Point With Multiple Roles



This is an example of a point having two roles defined to it. The Techs group needs to be able to control the Camera point to be able to fix it but doesn't need to be able to see the Door point. Groups can access points if any of the point's Roles match the Group's access rights.

5.1.3.4 One User In Multiple Groups



This example shows how a user can be assigned to two different groups that each have access to different roles. Bill is a user who can only see the Dimmer point, but not change it. Mike can change the Dimmer point, but cannot see the Cameras point. Bob is a member of both the Managers group and the Techs group, allowing him to change the Dimmer point (via the Write access of the Techs group) and also see the Camera point (via the Read access of the Managers group).

5.1.3.5 Users/Groups

Under 'Users' in 'Views', a list of users on the system is shown. You can create new users for the system here or change passwords for current users in the system.

Creating groups or adding users to existing groups is done in the 'Groups' section of 'Views'. You should use groups to organize users who will need similar access rights on the system. (IT, Maintenance, Executives, etc.)

5.1.3.6 Roles

Inside the 'Roles' section of 'Views', is where you will create new roles and define how each group can interact with it.

- **None:** Points with this role will not show up in any GUI for that group.
- **Read:** Users are able to see the point and its value but are not able to change it in any way. Example: Seeing what the audio volume is set at, but not being able to change it.
- **Write:** Users have full control of this point in any GUI that it's in.

You will also see options related to alarms:

- **See:** The group can see alarms triggered by points with this role, but cannot affect them in any way.
- **Acknowledge:** The group can acknowledge an alarm, clearing it from the current list of active alarms.

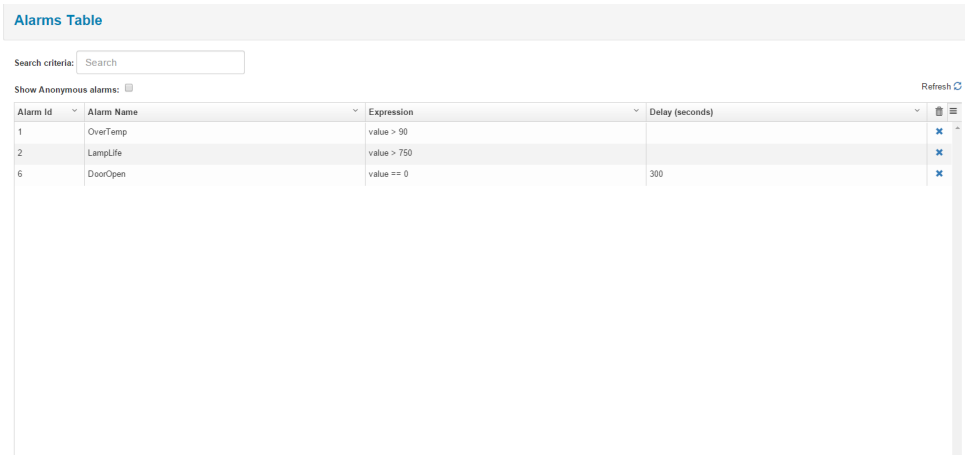
If group has neither of the alarm items selected, alarms will not show up for points associated with that role.

As well as user defined roles, there is an already created 'System' role that changes how users interact with the Overture software itself. Inside of this role you can determine if a user can see the Configurator, GUI Editor, or change points inside of the Configurator.

5.1.4 Logs

Inside 'System Logs' users can see what is happening within the Overture system itself. You can search to see just what certain users are doing, or just what points are being changed. Logs will also help you troubleshoot the system when you aren't getting the expected result you wanted from your control panels or pages. A user's access to the logs are based on their access to the 'Sytem' role.

5.1.5 Alarms

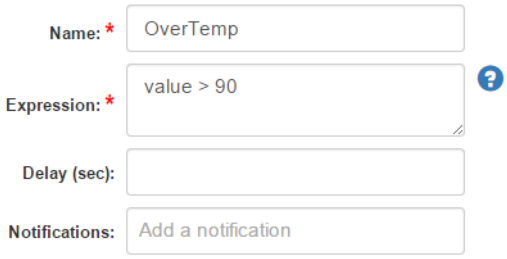


Alarms are triggered events that notify users of abnormal system operations. They can be set up for a variety of things including when a projector is at a high temperature, a PLC has gone offline, or a door has been left open for longer than 10 minutes. A user's access to alarms are based on their access to the role associated to that alarm's point.

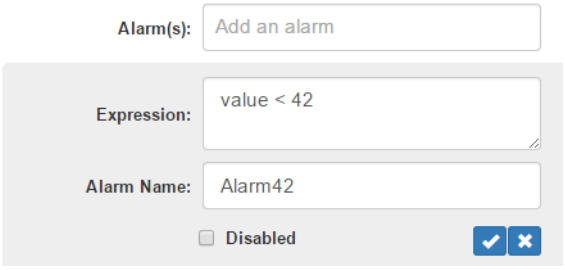
Alarms can be separated into two parts:

- Setting Alarms Up In The System
- Interacting with Alarms

5.1.5.1 Setting Up An Alarm



There are two ways to set up an alarm on the system. First you can go to the 'Alarms' view inside of Configurator and set up your alarm. After that, click on a point and specify the name of the alarm in the alarms section.
For example, you could specify a projectorOverTemp alarm in the Alarms view and then apply it to all the projectors in the point view using multi-selection.



Another way to set up an alarm is in the 'Points' view. There, when selecting a point you will see an area to specify an existing alarm. Clicking the plus icon allows you to directly set up an alarm on that point, adding a name if wish.

5.1.5.2 Alarm Expressions

No matter how you set up your alarm, you will need to define what triggers it. This is known as the alarm expression. The alarm expression is the logic the system uses to see whether an alarm should be triggered and relates directly to the point.
For example, if you are writing an alarm that triggers when a projector is above 90 degrees, your expression would be: `value > 90`. This tells the system whenever the point associated with this alarm has a value over 90, trigger an alarm.

When writing your alarms you will have the following options related to points:

- **value:** or point.value. This is the value of the point stored in the Control Server. It may be a string, or a number depending on what type of variable the point is. If the point is an enum, the value is the current index of the enum.
- **string:** or point.string. This is the string associated with the current index of an enum. For example, Projector.Power might be an enum with 'Off and 'On'. If the Projector is off, the value would be 0, but the string would be 'Off'.

Next, you will need to write the comparison. `<, >,<=, =>, ==` all work. Then you will need to write a value (For example: 42 or "On").

Expressions, can be combined with `&&` or `||` for AND/OR comparisons. For example, `value > 100 || value < 20`. The alarm would be triggered if the point's value was above 100 or below 20.

An expression can also directly reference a value of another point within the system. For example, you can attach an alarm to `Room1_Projector.Power` and for the alarm expression use `Server1.Room1_Projector.Power.string == "On" && Server1.Room2_Projector.Power.string=="On"`. In that case, the alarm will only trigger if both Projectors are on at the same time.
Note The logs only indicate an alarm to points that are tied to the alarm even if other points are referenced in those alarm expressions.

Expressions can also use some of the inheritance within the system. If the point tied with this alarm is a device variable, you can check the value or string of any sibling device variables by calling parent.variableName.value(or.string).
For example, if you need to check the shutter and power of the projector you can set an alarm to the Projector.Power point and and use an alarm expression : `string == "On" && parent.Shutter.string == "Closed"`. Now, you have an alarm that will trigger only when both the Projector Power is on, and the shutter is closed.

Alarm expressions also have an advanced function called `replace()` that can be used when you want to check other device variables in the system but also use the alarm expression as a template, so you can apply it to multiple points. For example, if you have multiple rooms that have projectors and occupancy sensors and you want have one complex alarm that monitors these attributes together you can use `replace()`.
Your alarm expression would be something like this : `value > 40 && replace('Projector.Temperature', 'HVAC.Occupancy').string == 'Occupied'`

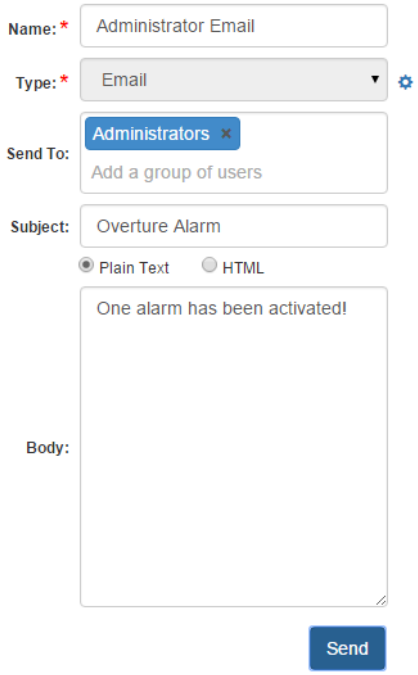
This complex alarm only works when variable names of the points have the same prefix (ex: Room1_Projector.Temperature & Room1_HVAC.Occupancy). In this example, the alarm must be tied to the Projector.Temperature point. When that point's value is above 40 and then the point with a similar name but HVAC.Occupancy string is equal to 'Occupied' the alarm will trigger. This allows you to create template alarms for devices that exist in the same room or other scenarios where more information is needed for the alarm.

5.1.5.3 Alarm Delay

Inside of the Alarms view, an alarm can have a specific amount of time associated with it before it alerts in the system. For example, the room temperature becoming 90 degrees once for 2 seconds might not matter but staying that way for 2 minutes may indicate a problem.

You can enter a Delay value in seconds to indicate how long the alarm expression must remain true before alerting the system. A floating point number can be given to increase precision.

5.1.5.4 Alarm Notifications



Alarms that are triggered can automatically send notifications out to users on the system. This is set-up in the Notifications view.

There are a few options when setting up a new notification:

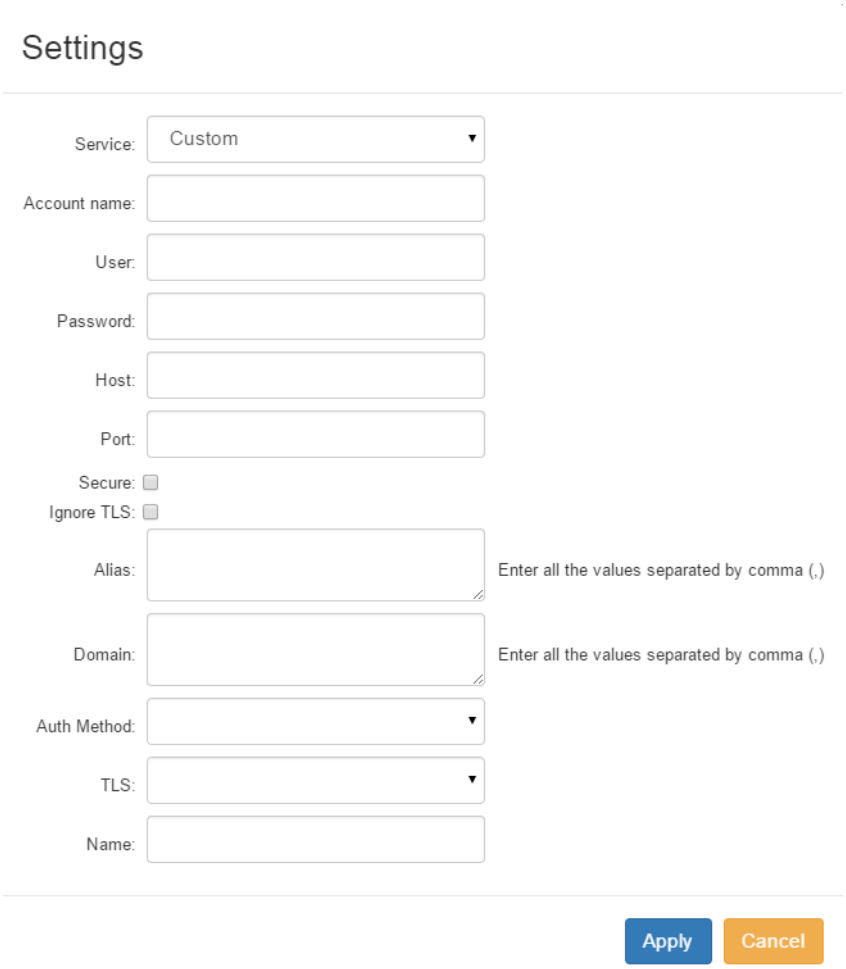
- **Name:** The name of notification. Alarms can have multiple notifications tagged to it.
- **Type:** For now, the only type is Email.
- **Send To:** The group within Overture this will send the notification to.
- **Subject:** Subject of the notification.
- **Plain Text/ HTML:** If writing an HTML notification, you will need to write all the HTML.
- **Body:** What is sent to the users.

Variables In The Subject/Body

The subject and body can have the following variables used in them for a more dynamic email:

- %pointname%: The name of the point which has triggered the alarm
- %pointvalue%: The value of the point on which the alarm triggered.
- %alarmname%: The name of the alarm
- %date%: The date the trigger condition has been detected
- %trigger%: The alarm expression

Setting Up Email

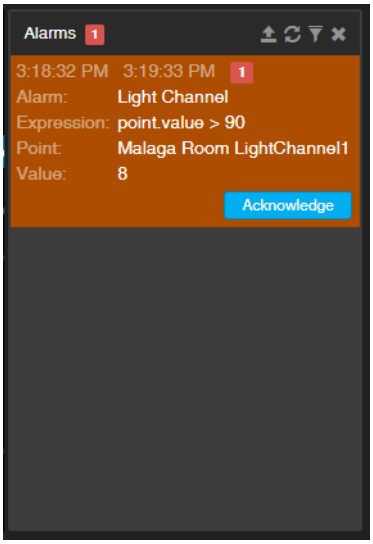


You will need to set-up the email server Overture should use for sending emails out. You can specify from an existing list of common email services, or you specify a custom email service.
The follow options are needed for a custom email service:

- **Account:** Human sent on top of the address
- **User:** Username to login to the email server.
- **Password:** Password to login to the email server.
- **Host:** Host name or IP of the SMTP server.
- **Port:** Port for the server.
- **Secure:** Whether to use SSL or not.
- **Ignore TLS:** Turns off STARTTLS if true.
- **Alias:** Aliases that maybe needed by the email server
- **Domains:** Domains that maybe needed by the email server.
- **Auth Method:** Defines the preferred authentication method.
- **TLS:** Which encryption to use if using TLS.
- **Name:** Client host name, needed by the server sometimes.

Once set up, you can send a test email by clicking the send button. This allows you to make sure the email server is working correctly.

5.1.5.5 Interacting with Alarms



Alarms appear as specialized logs in the system. In the Dashboard view, they appear in the logs view as the

number of current alarms within the system. Clicking that number will take you to the 'Alarms' view of the logs. Here you can see the alarm name, the time it first happened, the time it last happened, the number of times it has triggered, the name and expression of the alarm, the point the alarm happened on, and the current value of that point.
The same alarm number shows up next to views in Dashboard, in case you have logs hidden. In MagicMenu, the alarms show up as a red number in the title bar.

If you know an item should be shown as an alarm, but isn't, make sure your access rights for that point allow you to see alarms.

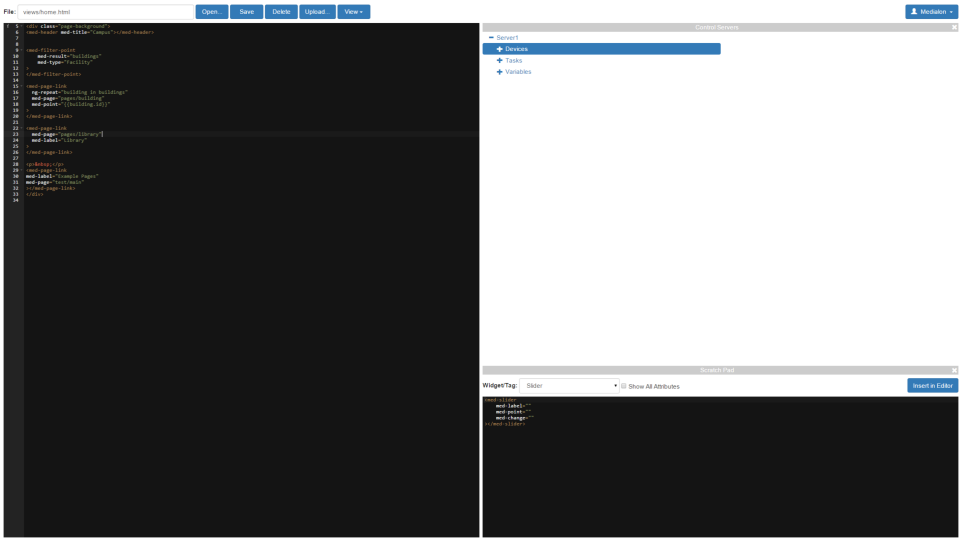
Alarms have four different states:

- Triggered and Active: This means the alarm expression has become true, is still true, and no one has acknowledged it.
- Triggered and Non-Active: This means the alarm expression has become true, no one has acknowledged it, but the expression is no longer true.
- Acknowledged and Active: This means the alarm expression has become true, a user has acknowledged it, but the expression is still true. This will clear it from the current list of alarms but the system will know it's still active.
- Acknowledged and Non-Active: This means the alarm expression has become true, is no longer true, and the user has acknowledged it. This clears the alarm from the system. If the alarm is re-triggered it's counted as a new instance of the alarm.

5.1.5.6 Disabling Alarms

There may be times when a device is in a known failure state for a long period . Inside Configurator, in the points view, you can disable an alarm on a point. This will no longer log when the alarm expression is triggered for that point.

5.2 GUI Editor



The GUI Editor is an in-browser HTML editor provided to edit Dashboard control panels, Magic Menu pages, CSS files or other text based assets you bring into your Overture project. To open the Overture GUI Editor, use the User menu at the top right corner of the Dashboard or Configurator interface and select Overture GUI Editor. You may be required to log in.

You can customize the way GUI Editor looks and feels for you by going to 'Views':

- **Scratch Pad** uses the project browser and widget selection tools to generate the HTML code for the Overture tags used in writing panels/pages. Using this will speed up programming time by allowing quick copy and pasting.
- **Control Servers** will allow you to see the devices, tasks, and variables ingested into the Overture project. Selecting one automatically fills the scratch pad with information used to display or control that item.

5.2.1 Assets

Assets are the important pieces to the Overture project and relate directly to how things are shown in the GUI. Every asset, whether it be an HTML file, a CSS document, or an image must be stored in the assets folder of your project.

Assets are stored in your assets folder of the UX Server. They can be uploaded directly through the GUI Editor by clicking 'Upload...'. The main assets for your project include:

- **Images:** Images used as maps or displayed inside of html pages. (Use the following formats: .jpeg, .png, .bmp)
- **Styles:** CSS files used to style the HTML inside the browser. You can create different themes by making new CSS files and calling them as you want in the project.
- **Views:** HTML files that are displayed either as Control Panels or Magic Menu pages. The same HTML file can be used as both a Control Panel and a Magic Menu page.

5.2.1.1 Organization

Three subfolders have been provided for organization of your HTML pages. You do not need to store them in those subfolders, but it is recommended you do.

Note: 'home.html' is the first page opened by Magic Menu and should not be moved.

5.2.2 Default Assets

Overture provides a set of starting assets available for use right after installation. The following items exist and can are used in the following ways:

- **Device Templates:** These are files have are used to control or view data from specific devices based on the subtype (projector.html, lighting.html, etc). To use these, assign them in the Configurator via 'Control Panel' selection.
- **Room Templates:** This is the `room.html` in the `views/common` folder. This file will create a room control panel based on the devices that are children of the room point. This is assigned in the Configurator via 'Control Panel' selection.
- **Home Page:** This file is called as the starting page to Magic Menu. It will display any points as links that have the special tag 'Home`. Clicking on these points, will load the Control Panel specified in the Configurator, or their default Control Panel.
- **Default Templates:** These files display their children as page-links for Magic Menu. They are used as default templates based on type, for when a point is displayed as a link in Magic Menu but does not have a 'Control Panel' associated with it.

These files can be edited for your specific project needs. Use the provided HTML, and following section as a guide when editing panels or creating your own.

5.2.3 Writing Panels and Pages

Overture panels and pages are written in HTML. More precisely, they are written in a slightly customized version of HTML that uses pieces of JavaScript to help control things. Having some knowledge of these two languages will help you, but it is not necessary for creating pages or panels within the Overture system.

The main three components you will use to write your displays are:

- **Med Tags:** Special HTML tags used to create different displays, sliders, or organizational tools specific to the Overture system.
- **Med Attributes:** The attributes used within HTML tags that will directly affect what/how that tag interacts with the database points/browser.
- **Functions:** These are the commands inside of attributes that perform an action related to that attribute. For example, the setVariable function might be called to change a variable when a user moves a slider on the page.

Note: Normal HTML tags like `<div>`, `<img>`, `<a>`, and `<iframe>` will work inside Overture control panels and pages.

5.2.3.1 Displaying Variables

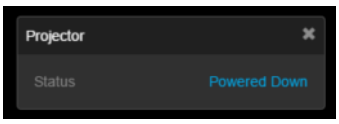
Drag a column header here and drop it to group by that column.

Id	Name	Short Name	Variable Name
20361	Projector Status	Status	Projector.Status

Let's say we have a Projector Status variable like the one above inside of your database and you want to display that inside of a control panel on the Dashboard. You will need to write a med tag.

Med tags are how the panels and pages you write interact with the points inside of your Overture project. Each tag has a number of attributes that are used to help interact with the points you want to monitor or control. Here is a basic example of a tag:

```
<med-display
    med-point="Projector.Status"
></med-display>
```



In this example we made a simple display, for your control panel. The `<med-display>` tag creates the display. The attribute `med-point` tells us what point inside of the project to look at. In this case, we are referencing the point directly by using its variable name in the project. Next, the value of the object we see 'Status' displayed. This is the point's short name. If you need to display something else you can specify the `med-label` attribute.

5.2.3.2 Changing Variables Statically

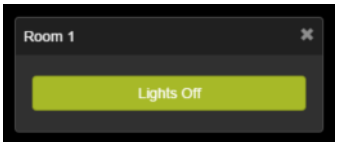
Drag a column header here and drop it to group by that column.

Id	Name	Short Name	Variable Name
20362	Room 1 Lights	Lights	Room1_Lights

Often you will want to do more than just monitor variables within Overture. There are a few attributes to help you do this: `med-click` for buttons, and `med-change` for sliders and other tags.

Inside of those attributes you will need to call a function that will take care of changing your variables. In this case we are going to call `setVariable()`.

```
<med-button
    med-label="Lights Off"
    med-click="setVariable('Room1_Lights',0)"
></med-button>
```



In this example, we have a simple button. When the user clicks the button, the `med-click` attribute is triggered. That attribute tells the `setVariable` function to run.

`setVariable` is a function that takes in two parameters. First, it needs to know what to change. We must provide it the variable name of the point we want to change. Second, it needs to know what to change it to. Here, we are turning it off so we set it to 0.

When you change a variable to a direct value this is known as changing it **statically**. This means it does not matter what the value of the point was before we changed it. It only cares what we are changing it to now.

5.2.3.3 Changing Variables Dynamically

Let's say instead of making just an off button, we want to be able to control the lights with a slider. We can't set the variable equal to just a static value or else every time a user moves the slider, the slider will only change it to that hard coded value. Instead we will now be making a **dynamic** change.

```
<med-slider
    med-point="Room1 Lights"
    med-change="setVariable('Room1_Lights',value('Room1_Lights'))"
></med-slider>
```

There are a few differences in this example. First, we need to tell the slider which point we want to change and display. We use the `med-point` attribute to determine which point we are going to affect, referring once again to the variable name of that point.

Second, the slider uses a `med-change` attribute. This will run any function tied to when the slider is moved in either direction.

Finally, `setVariable` now looks different. We still have the same first parameter because we are still acting upon 'Room1_Lights', but instead of just giving it a number, we call another function.

The `value()` function returns the value of the point we specified. In this case, 'Room1_Lights'. So instead of just giving a direct number, we now give a number that is changed as the value is changed.

Here is how it works: The user moves the slider, which is tied to point ('Room1_Lights'), which changes the value point. Now the slider's `med-change` attribute triggers and runs the function that is tied to it. In our case, `setVariable` will now run.

The `setVariable` function now tells the Control Server that it wants to change the variable to what we specified. In our case we specified the value of the point. **This is different than the current value of the variable in the Control Server. The slider changed the value of the point in Overture and that value is what is returned by the value() function.** Now, once the Control Server variable is actually changed by the `setVariable` function, the Control Server reports the variable change back to the browser. The new value and the point takes on the updated value.

Knowing how this round-trip communication works will help you troubleshoot your code, when you get unexpected results. For example, if we use a slider and `setVariable` but instead of using a dynamic value like `value('Room1_Lights')` we use a static value like 1. The slider moves wherever the user drags it , `setVariable` changes the Control Server variable to 1 and reports back that value to the browser. The slider snaps back to 1 because that's what the point's value was set to by the return information from the Control Server.

5.2.3.4 Using perform() To Start Tasks

Overture can do more than just affect variables. You can directly control tasks on your system.

```
<med-button
    med-label="Configure Floor"
    med-click="perform('ConfigureSecondFloor','starttask')"
></med-button>
```

In this example, we are making a button to start a task. That task runs on Control Server. To do this, we use a function called `perform()`.

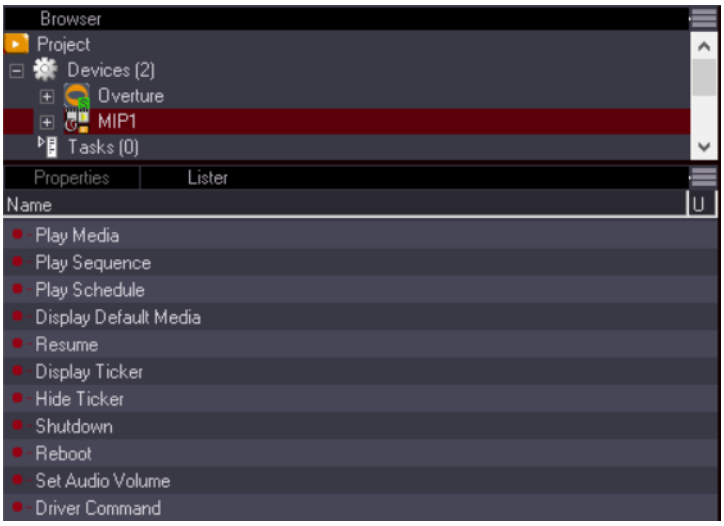
When controlling a task, the `perform` function has two parameters. First, we need to know what point we are acting upon. This point should be a task ('ConfigureSecondFloor').

Second, we need to give that point a command. With tasks, the 'starttask' command will start the task. The 'stoptask' command will stop it.

5.2.3.5 Using perform() To Control Devices Without Parameters

The `perform` function has an even more powerful ability. It can be used to control devices directly.

```
<med-button
    med-label="Reboot MIP"
    med-click="perform('MIP1','Reboot')"
></med-button>
```



In this example, we have a MIP device in our Control Server and a point in the UX Server named `MIP1`. Just like our previous example, we are using the `perform` function and specifying a point. Instead of the `starttask` command, this time we are using the `Reboot` command that is native to the MIP device inside of the control server.

Note: Each device has its own commands associated with it. You will need to check these commands inside of Control Server.

5.2.3.6 Using perform() To Control Devices With Parameters

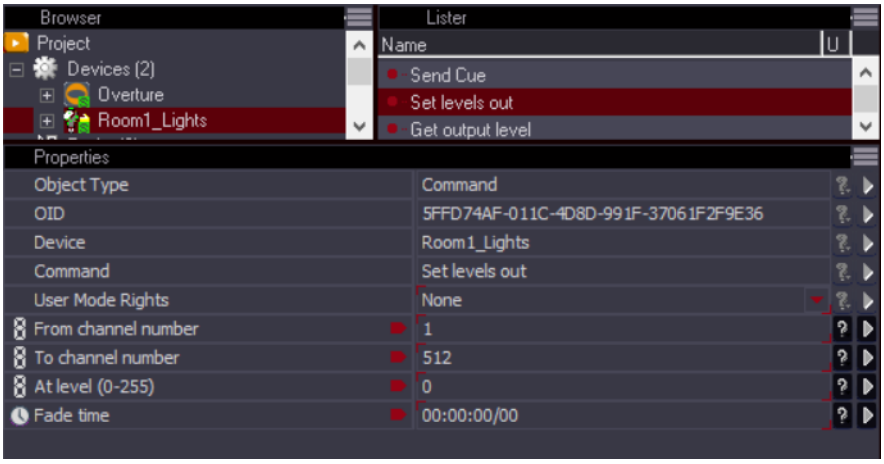
Some commands from devices need to be given values. For example, a 'Set levels out' command needs to know channels and levels to set for the outputs. The `perform` function handles parameters in a slightly more advanced way.

```
<med-slider
    med-point="Room1 Lights"
    med-change="perform('Room1 Lights','Set levels out',
                {From channel number:1, To channel number: 4, At level: value('Room1_Lights') })"
></med-slider>
```

Let's take a close-up look at the perform:

```
med-change="perform(
'Room1 Lights',
'Set levels out',
{
    From channel number:1,
    To channel number: 4,
    At level: value('Room1_Lights')
}
)"
```

You can see `perform` still uses the first two parameters. It needs a point (device), it needs the command to perform ('Set levels out'), and now it needs a third parameter which is the **command parameters**.



In this case, if you look at the `Set levels out` command inside of Control Server, you will see four unique parameters: `From channel number`, `To channel number`, `At level`, and `Fade time`. Just like Control Server programming, if you run this command, you have to change the parameters that you do not want to use the

default values. It is the same here.

The first step is we must enclose all of our device parameters with `{ }`. This lets the language know these are all command parameters for that command.

Second, you must specify the command parameter you wish to change followed by a `:`. This is a special assignment expression used in JavaScript.

Third, assign that command parameter a value. This value can be static or dynamic as seen above.

Fourth, separate each parameter you wish to change by a comma. If you do not wish to add any further command parameters, do not add a comma. **Note:** Even if a device has 10 command parameters, you do not need to specify all of them. Any command parameter not specified will use its default value.

Finally, make sure the command parameters were correctly enclosed with a `}` and close the `perform` function with the closing `)`. This syntax may seem confusing but you can use as much empty space as you need to help it make sense.

5.2.3.7 Using openUrl()

You may need to provide a link to an external site within Dashboard through the special tag objects. You can do this using the function openUrl().

```
    <med-button
        med-label="Go to Google"
        med-click="openUrl('http://www.google.com')"
    ></med-button>
```

5.2.3.8 Advanced Syntax

Overture has some helpful syntax items to avoid needing to write complex lines.
When using advanced syntax, one way is not better than another. Please use whatever you feel most comfortable with.

Advanced Point Reference

A point can be referenced multiple ways. We've already gone over the first way. That was known as **direct reference**. Direct reference is using the variable name of the point. Example: `'Room_1_Projector'`

Another way is to call the `point` object. The `point` object refers to the point the control panel is tied to. For example if the control panel was tied to a projector device in the Configurator you could write :`perform(point, 'Power On')`.

If the control panel is tied to a device, that device's variable can be referenced by their relative name. For example, if we wanted to just display that projector's power we could write:

```
    <med-display
        med-point=".Power"
    ></med-display>
```

In that case, the rest of the name will be taken from the point that the control panel is tied to.

A point can also be referenced by `$point`, if already referenced in the tag. Example:

```
    <med-button
        med-point="Room 1 Status"
        med-click="setVariable($point, 'On')"
    ></med-button>
```

Point Property Reference

Points inside of Overture are objects that have properties assigned to them. A lot of the work with these properties is done behind the scenes. For example, when you write :

```
    <med-display
        med-point="Room_1_Audio.Volume"
    ></med-display>
```

You are telling the control panel to display both the **name** or **shortname** property of that point, along with the **value** of that point.

You saw that sometimes, you need direct access to that property. For example, when using a slider to dynamically get the value of the point.
Before, we would use a function `value()` to retrieve the value of a point, but there is some other syntax that will grab these properties for us.

- **$value** : This returns the value of the point.

```
<med-slider
    med-point="Room1 Lights"
    med-change="setVariable($point, $value)"
></med-slider>
```

- **$string** : This returns the string of the current index of an enum variable.

```
<med-select
    med-point="Room1 Status"
    med-change="setVariable($point, $string)"
></med-select>
```

- **$parent** : This returns the parent (as an object) of the point.

```
    <med-button
        med-point="Room 1 Projector.Power"
        med-click="perform($parent, 'Power On')"
```

```
    ></med-button>
```

Note: These references are only for the tag they are written in. `$value` from one tag will not be carried over to another.

5.2.3.9 Default Behaviors

Overture tag attributes `med-change` and `med-click` have a default behavior associated with them if left blank or not specified. This default behaviour depends on the type of point specified with `med-point`.

If the point is a variable `med-change` has a default behavior of `setVariable($point, $value)`. So for example:

```
<med-slider
    med-point="Room1_Lights"
    med-change="setVariable($point, $value)"
></med-slider>
```

and

```
<med-slider
    med-point="Room1_Lights"
></med-slider>
```

do the same thing.

If the point is a task, and tied to `<med-button>`, the default behavior of `med-click` is to start the task. For example:

```
    <med-button
        med-point="Room 1 Turn On"
        med-click="perform($point, 'starttask')"
    ></med-button>
```

and

```
    <med-button
        med-point="Room_1_Turn_On"
    ></med-button>
```

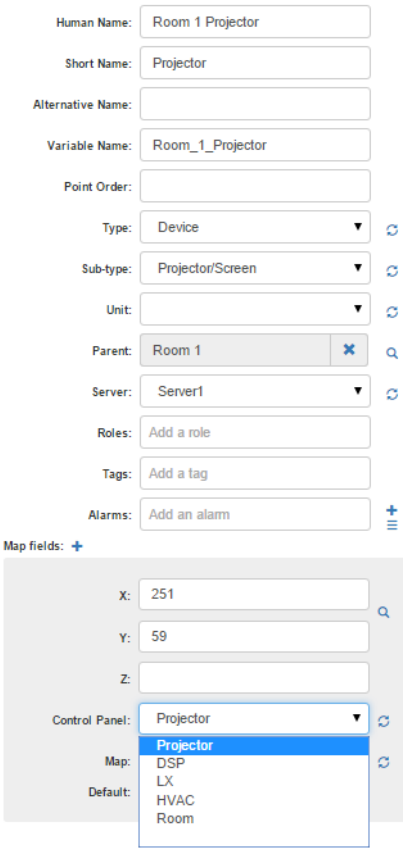
do the same thing.

5.2.3.10 Writing Templates

Overture provides several features which ease the authoring of many nearly identical panels or pages. These panels/pages are called **templates**.

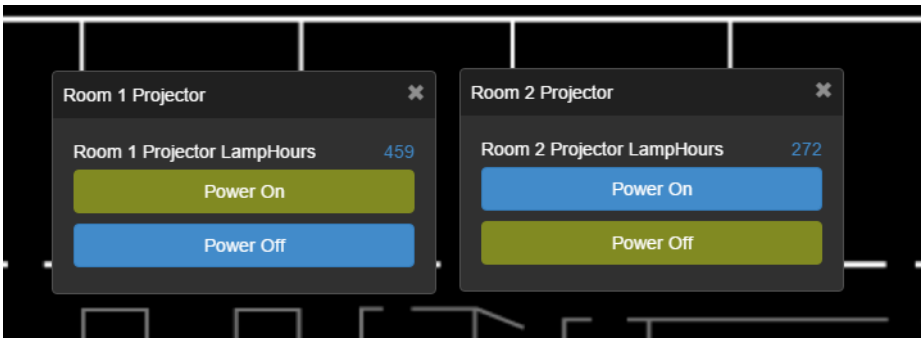
Device Control Panels

One way to achieve templates is to make device control panels. To do this, you will need to make a control panel asset and assign it to a few devices:



By doing this, you will be able to use some of the advanced syntax described earlier to create non-direct references to the points. For example:

```
<med-display
    med-point=".LampHours"
></med-display>
<med-button
    med-point=".Power"
    med-label="Power On"
    med-click="perform(point,'Set Power',{Status:'On' })"
    med-on-state="1"
></med-button>
<med-button
    med-point=".Power"
    med-label="Power Off"
    med-click="perform(point,'Set Power',{Status: 'Off' })"
    med-on-state="0"
></med-button>
```



You can see that with the same HTML you can control devices without having to write a panel for each one.

Filtering Using Inheritance

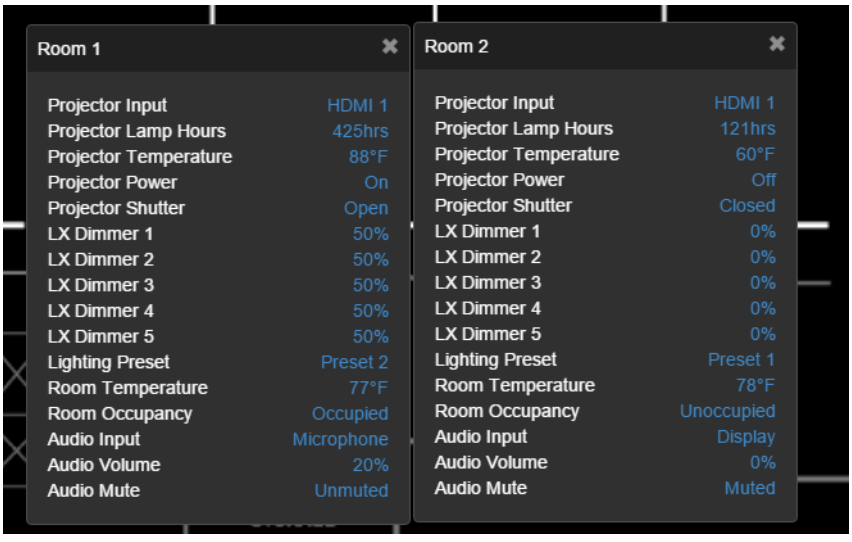
Another way templates can be achieved is by using filtering to find the points you want to display and control. This is achieved with an item called `med-repeat`.

For example, lets say you wanted to see every variable related to every device in a given room. We can do that with this code:

```
<med-point-table
    med-repeat="{parent: point, type: 'device'}"
></med-point-table>
```

In order to acheieve this, we are doing two things. First, we want to display all the device variables. We do this with the tag `<med-point-table>`. That tag takes any point and creates a display for each child variable related to that point. So if a device, has the `.Power, .Temperature, and .Shutter` variables it will display each of them individually.

Then, the `<med-point-table>` has an added special attribute `med-repeat`. This attribute first finds an array of points defined by the filter given to it. We want to find points that are devices in the room, so we proide the `{parent: point, type: 'device'}`. This means the filter returns all points that are children to the current control panel (the room), and are also devices. Once that array of points is found, the `med-repeat` attribute creates a seperate `<med-point-table>` tag for each item in the array and passes that item in as that tag's `med-point`.



With those lines of code, you now have a template that can be assigned to any room. It will find all the devices of that room, no matter how many there are and display all of their variables no matter how many exist.

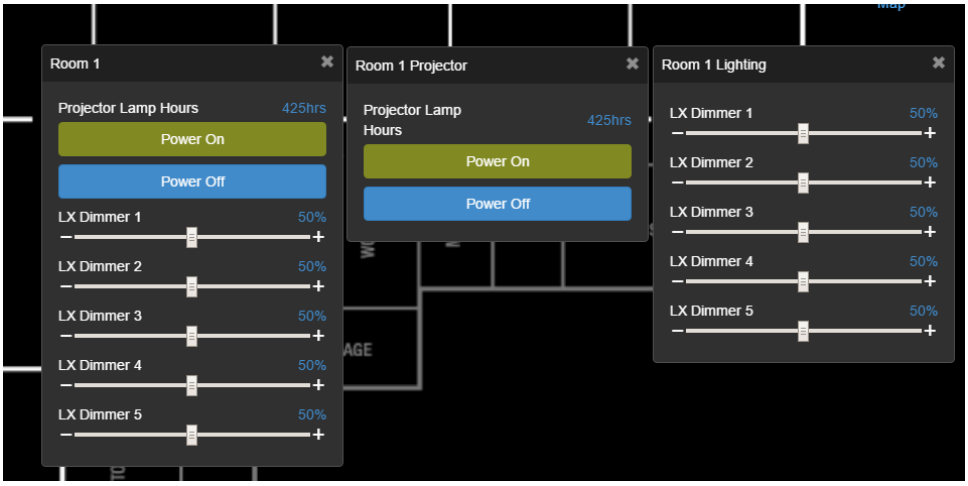
`med-repeat` can be used with many tags and in many different ways. To get more information please see: Tag Reference: Med-Repeat

Using inclusion

Another way we can make templates is using already existing code. For example, you may have already written device control panels and now we want to create our room panel. We can taken already written code add it in with the tag `med-include`.

```
<med-include
    med-repeat="{parent: point, subtype: 'projector'}"
    med-file="common/proj"
></med-include>
<med-include
```

```
    med-repeat="{parent: point, subtype: 'lighting'}"
    med-file="common/lx"
></med-include>
```



In this example, `med-include` is grabbing the already exisiting HTML files that were written for both the projector and the lighting. (`common/proj` and `common/lx`). `med-repeat` then finds the projector and lighting devices that exist in the room and passes them in as context their respective HTML files. The attribute then repeats the `med-include` equal to the number of devices it found in each filter. (1 in this case).

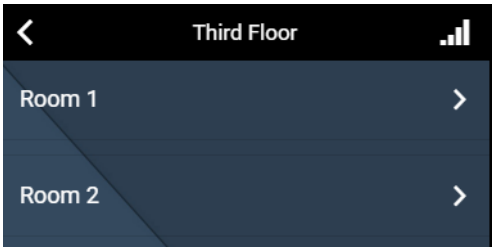
5.2.3.11 Magic Menu Pages

MagicMenu pages are written the same way as Control Panels, and the same HTML already written for Control Panels can be loaded into MagicMenu, but there are some important things to note.

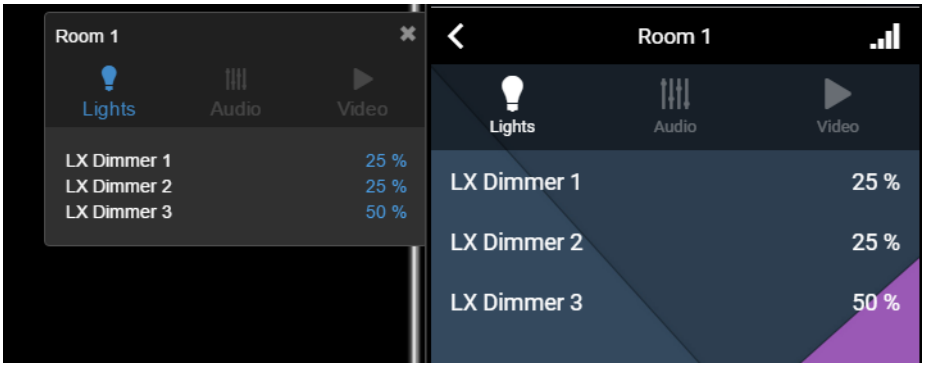
First, the assets folder has `views/home.html` which is the starting page loaded when you first load MagicMenu.

Second, because you don't assign the HTML page directly to a point like you do with Control Panels, you will need to provide some context to the pages you load.
We can do this with a MagicMenu specific tag called `<med-page-link>`.

```
<med-page-link
    med-repeat="{type: 'room'}
    med-page="common/room"
></med-page-link>
```



`<med-page-link>` requires a file parameter, similar to the way `<med-include>` needed a file. Once that file is specified, you can pass in a point (which is done with `med-repeat`). The point specified gives both give the link a name and pass its information into the HTML loaded from the specified file.



When the room is clicked, you can see the same HTML is loaded and you are able to control the same points. This allows only one HTML page to be written and for it to be used by multiple interfaces with no extra work.

5.2.4 Tag Reference

There are many tags created to help you design interfaces for users. This section explains how all of the basic tags work and provide examples to help you program. **Note:** Using the scratch pad option helps to see each attribute used in a tag.

5.2.4.1 Display

The `med-display` tag displays the value of a variable.

Example

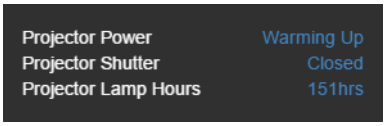
```
<med-display
    med-label="Audio Volume"
    med-point="Player1.AudioVolume"
    med-suffix="dB"
></med-display>
```

Attributes

- `med-label`: the text which is displayed at the left of the value. Defaults to the name of the point.
- `med-point`: the point to be displayed.
- `med-suffix`: the suffix to be appended to the point value. . Overrides a point's unit from the database if specified.
- `med-big`: specifies whether the value should be displayed using a larger character in Magic Menu. . Defaults to `false`.

5.2.4.2 Point Table

The `med-point-table` tag displays a list of point values. Useful when gathering all the data from a single device.



Example

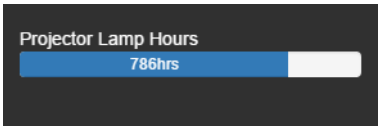
```
<med-point-table
  med-point="Projector1"
  med-subpoints=".PowerStatus,.ShutterStatus,.LampHour"
></med-point-table>
```

Attributes

- `med-point`: the root name of the points which you are displaying.
- `med-subpoints`: the name of the subpoints, separated by a comma, which, combined with the root name, lists the points you want to display. If nothing is specified it displays all children of the point.

5.2.4.3 Progress Bar

The `med-progress` tag allows monitoring of a variable inside of a gauge. This is used for comparing the current value against a minimum or maximum value.



Example

```
<med-progress
    med-point="Room_2_Projector.LampHours"
    med-min="0"
    med-max="1000"
></med-progress>
```

Attributes

- `med-label`: the text displayed with the progress bar. Defaults on the name of the point.
- `med-point`: the point being monitored.

- `med-max`: Set the maximum value of the progress bar. . Uses the maximum variable value from the Control Server project file, if not specified
- `med-min`: Set the minimum value of the progress bar. . Uses the minimum variable value from the Control Server project file, if not specified
- `med-suffix`: the suffix to be appended to the point value. . Overrides a point's unit from the database if specified.

5.2.4.4 LED

The `med-led` tag creates an LED that changes color based on the value provided to it. It's useful for monitoring enums or integers and providing a color index to their values.

Example

```
<med-led
    med-label="Power"
    med-point="Projector.Power"
    med-colors="red, #0000FF, #FFFF00, green"
></med-led>
<med-led
    med-label="Shutter"
    med-point="Projector.Shutter"
    med-colors="red, green"
></med-led>
```

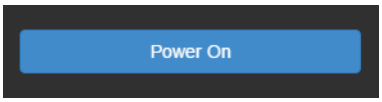
Note: In the above example, the Projector's power is an enum with a value of 3, and the shutter is an enum with a value of 0.

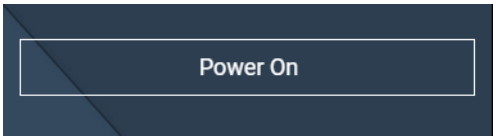
Attributes

- `med-label`: the text which is displayed at the left of the LED . Defaults to the name of the point.
- `med-point`: the point to display.
- `med-colors`: the comma seperated list of colors relating to the value of the point being monitored. If the value is not a number, the color will default to the zero index(First item in the list). If the value of the point is greater then number of colors listed, it will take the last color specified in the list. (By default colors are red(0) and green(1))

5.2.4.5 Button

The `med-button` is a button that triggers an action when clicked (or tapped). It can also change its appearance depending on the value of a specified point.





Example

```
<med-button
    med-label="Power On"
    med-click="perform('PowerOn', 'starttask')"
    med-point="Projector.PowerStatus"
    med-on-state="1"
></med-button>
```

Attributes

- `med-label`: the text displayed in the button. .
- `med-click`: the action triggered when the button is clicked.
- `med-point`: the point being monitored which changes the appearance of the button. .
- `med-on-state`: the value of the point that causes the button to appear highlighted.

5.2.4.6 On/Off Button

The `med-on-off` tag creates two buttons, inline with each other, monitoring the same point.

Example

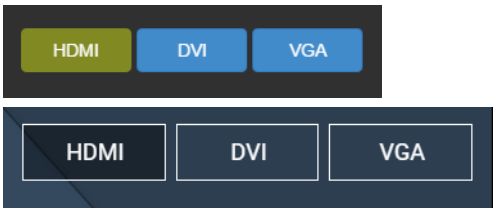
```
<med-on-off
  med-point="Projector.PowerStatus"
  med-label-on="Power On"
  med-state-on="1"
  med-click-on="perform('Projector','Power On')"
  med-label-off="Power Off"
  med-state-off="0"
  med-click-off="perform('Projector','Power Off')"
></med-on-off>
```

Attributes

- `med-point`: the point to be monitored.
- `med-label-on`: the text displayed on the left button
- `med-state-on`: the value of the point that causes the left button to appear highlighted.
- `med-click-on`: the action triggered when the left button is clicked.
- `med-label-off`: the text which is displayed on the right button
- `med-state-off`: the value of the point that causes the right button to appear highlighted.
- `med-click-off`: the action triggered when the right button is clicked.

5.2.4.7 Button Bar

The `med-button-bar` tag is a container that holds `med-button` tags. It displays the contained buttons inline instead of vertically, one after the other.



Example

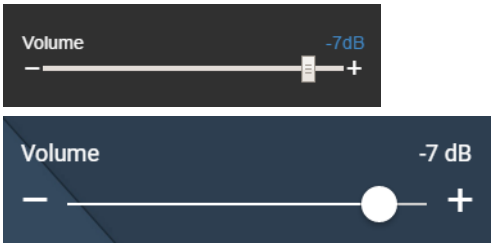
```
<med-button-bar>
    <med-button
        med-label="HDMI"
        med-point="Room 2 Projector.Input"
        med-click="perform('Room_2_Projector','Set Input',{ Input:'HDMI 1' })"
        med-on-state="0"
    ></med-button>
        <med-button
        med-label="DVI"
        med-point="Room 2 Projector.Input"
        med-click="perform('Room_2_Projector','Set Input',{ Input:'DVI' })"
        med-on-state="1"
    ></med-button>
        <med-button
        med-label="VGA"
        med-point="Room 2 Projector.Input"
        med-click="perform('Room_2_Projector','Set Input',{ Input:'VGA' })"
        med-on-state="2"
    ></med-button>
</med-button-bar>
```

Attributes

There are no specific attributes for the button bar.

5.2.4.8 Slider

The `med-slider` tag sends commands when the user moves a slider. The position of the slider marker is determined by the value of a point.



Example

```
<med-slider
    med-label="Volume"
    med-point="Player1.AudioVolume"
    med-change="perform('Player1', 'Set Volume', {Volume: value('Player1.AudioVolume')})"
    med-max="0"
    med-min="-80"
    step="1"
```

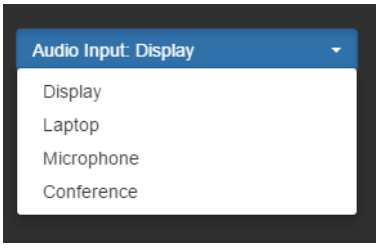
```
></med-slider>
```

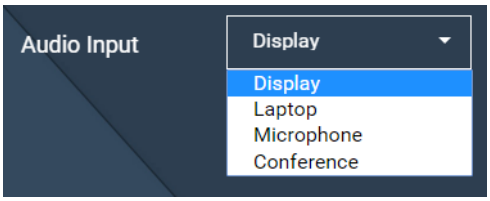
Attributes

- `med-label`: the text displayed with the slider. Defaults to the name of the point.
- `med-point`: the point being monitored.
- `med-change`: the action triggered when the slider is moved.
- `med-max`: Set the maximum value of the progress bar. . Uses the maximum variable value from the Control Server project file, if not specified
- `med-min`: Set the minimum value of the progress bar. . Uses the minimum variable value from the Control Server project file, if not specified
- `med-step`: Amount moved by cursor or 'up' and 'down' buttons in Magic Menu. Defaults to 1.
- `med-suffix`: the suffix to be appended to the point value. . Overrides a point's unit from the database if specified.
- `med-big`: specifies whether the value should be displayed using a larger character in Magic Menu. Defaults to `false`.
- `med-icon-up`: icon displayed on right of the slider. Default to "ion-chevron-right". See http://ionicons.com/ for the icons names. Only used in Magic Menu
- `med-icon-down`: icon displayed on left of the slider. Default to "ion-chevron-left". See http://ionicons.com/ for the icons names. Only used in Magic Menu

5.2.4.9 Select

The `med-select` tag creates a muliple-choice selection box, which allows the user to select an item from a list. It is intended to be used with `enum` type variables.





Example

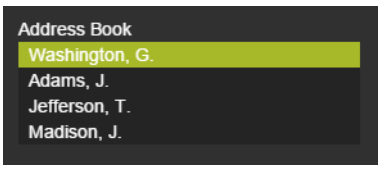
```
<med-select
    med-label="Audio Input"
    med-point="Room 2 DSP.Input"
    med-change="perform('Room_2_DSP','Set Input',{Input: $string})"
></med-select>
```

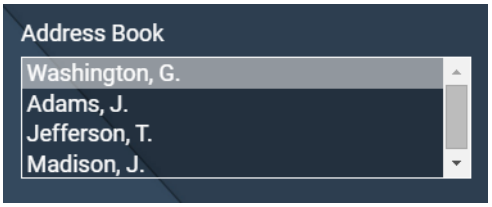
Attributes

- `med-label`: the text displayed with the select. Defaults to the name of the point.
- `med-point`: the point associated with the list, this point is intended to be an enum variable type.
- `med-change`: the action triggered when an item is chosen in the list.

5.2.4.10 List

The `med-list` tag displays a list of items, and highlights the selected item in the list. It is intended to be used with 'enum' type variables.





Example 1

```
<med-list
    med-label="Address Book"
    med-point="Room_1_VC.AddressBook"
    med-change="perform('Room_1_VC','Add To Call',{Entry: $string})"
></med-list>
```

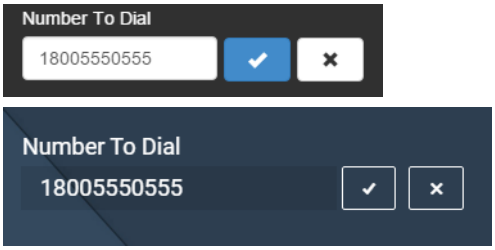
Attributes

- `med-label`: the text displayed above the list. Defaults to the name of the point.
- `med-point`: the point associated with the list, this point is intended to be an enum variable type.
- `med-change`: the action triggered when an item is chosen in the list.
- `med-count`: the number of list items displayed without needing to scroll. Defaults to 10.

5.2.4.11 Text Input

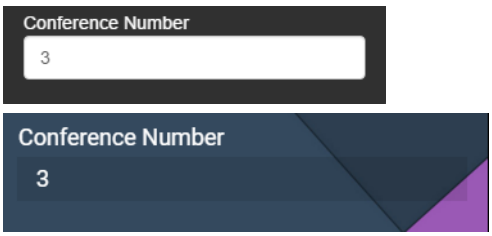
The `med-text-input` tag creates an input box that changes a variable when you type into it. It is used when you need the exact precision to a variable. There are two modes for this tag.

The default mode has a confirm button and clear button and will only perform the `med-change` when the confirm button is pressed. The second mode, eliminates the confirm and clear boxes and performs the `med-change` whenever the input box is changed(with each character entered).



Example 1

```
<med-text-input
    med-label="Number To Dial"
    med-point="Room 1 VC.NumberToDial"
    med-change="perform('Room_1_VC','Dial',{Number: $value})"
></med-text-input>
```



Example 2

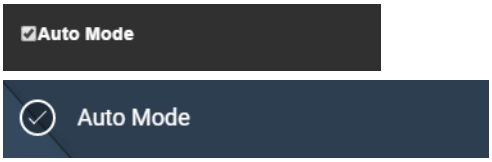
```
<med-text-input
    med-label="Conference Number"
    med-point="Room 1 VC.Key"
    med-change="perform('Room_1_VC','Key Press',{Number: $value})"
    med-confirm="0"
></med-text-input>
```

Attributes

- `med-point`: the point to monitor.
- `med-label`: the name of the text-input to display. Defaults to the name of the point.
- `med-change`: the action triggered when the input is modified.
- `med-confirm` : the mode in which the tag is operating. Can be 1 or 0. Defaults to 1(with Confirm/Clear buttons).

5.2.4.12 Checkbox

The `med-checkbox` tag allows you to toggle between values, and perform an action when checked, unchecked, or either.



Example

```
<med-checkbox
    med-label="Auto Mode"
    med-point="AutoMode"
    med-change-on="perform('SetAutoModeOn','starttask')"
    med-change-off="perform('SetAutoModeOff','starttask')"
></med-checkbox>
```

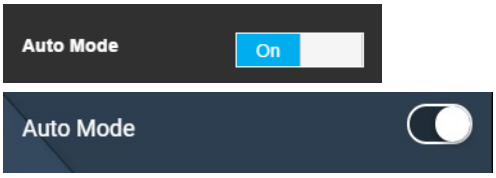
Attributes

- `med-label`: the text displayed with the checkbox. Defaults to the name of the point.
- `med-point`: the point to be monitored to change the status of the checkbox.
- `med-change`: the action triggered when the checkbox is checked or unchecked.
- `med-change-on`: the action triggered when the checkbox is checked.

- `med-change-off`: the action triggered when the checkbox is unchecked.

5.2.4.13 Toggle

The `med-toggle` tag toggles values like checkbox , but looks like a switch instead.



Example

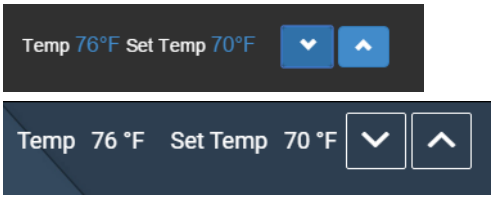
```
<med-toggle
    med-label="Auto Mode"
    med-label-on="On"
    med-label-off="Off"
    med-point="AutoMode"
    med-change-on="perform('SetAutoModeOn','starttask')"
    med-change-off="perform('SetAutoModeOff','starttask')"
></med-toggle>
```

Attributes

- `med-label`: the text displayed with the toggle. Defaults to the name of the point.
- `med-label-on`: the text displayed in the switch, while the toggle is on. Only seen in dashboard.
- `med-label-off`: the text displayed in the switch, while the toggle is off. Only seen in dashboard.
- `med-point`: the point to be monitored to change the status of the toggle.
- `med-change`: the action triggered when the toggle is switched on or switched off.
- `med-change-on`: the action triggered when a toggle is switched on.
- `med-change-off`: the action triggered when the toggle is switched off.

5.2.4.14 Set Up/Down

The `med-up-down` tag is composed of two variables to display and two arrows (up and down). It is useful for adjusting a target value while also monitoring the actual value of a variable.



Example

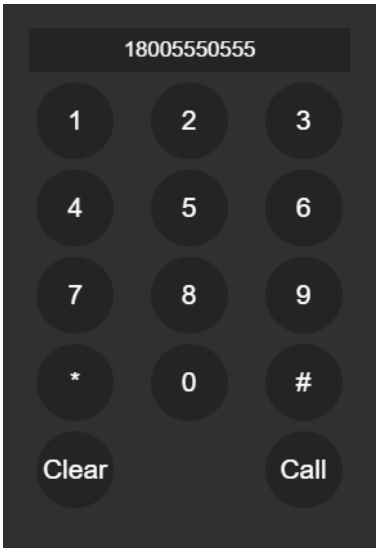
```
<med-up-down
  med-point="HVAC.CurrentTemp"
  med-label="Temp"
  med-target-point="HVAC.Target"
  med-target-label="Set Temp"
  med-change-up="perform('HVAC, 'HVAC Set Temp', { Mode: 'Up'})"
  med-change-down="perform('HVAC, 'HVAC Set Temp', { Mode: 'Down'})"
></med-up-down>
```

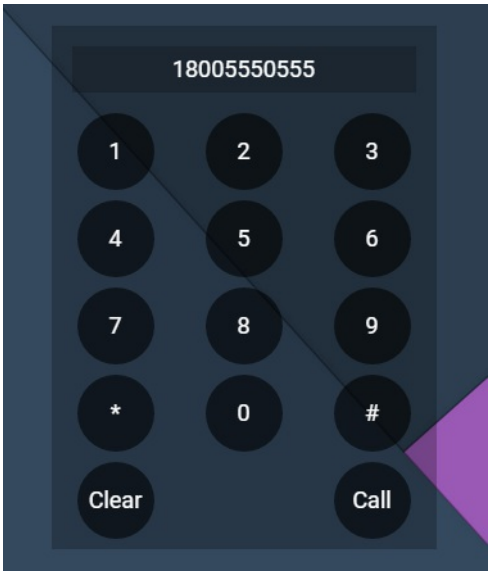
Attributes

- `med-point`: the point to monitor in the left display.
- `med-label`: the text displayed next to the left display. Defaults to the name of the point.
- `med-target-point`: the point to monitor in the right display.
- `med-target-label`: the text displayed next to the right display. Defaults to the name of the point.
- `med-change-up`: the action triggered when the up arrow is clicked.
- `med-change-down`: the action triggered when the down arrow is clicked.

5.2.4.15 Keypad

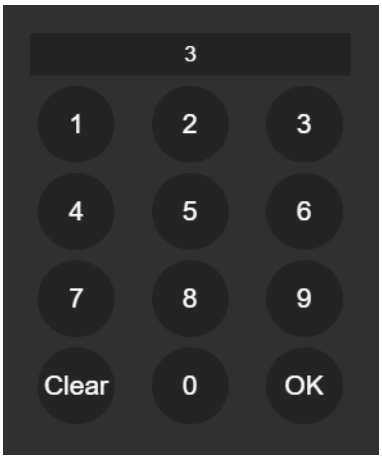
The `med-keypad` tag displays a set of buttons for making a call or for entering number based information. It's useful for interfacing with a video conference system.





Example 1

```
<med-keypad
    med-point="Room_1_VC.NumberToDial"
    med-change="perform('Room_1_VC','Dial',{Number: $value})"
    med-type="dialer"
></med-keypad>
```



Example 2

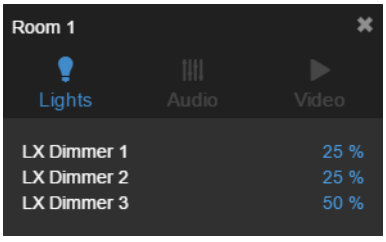
```
<med-keypad
    med-point="Room_1_VC.Key"
    med-change="perform('Room_1_VC','Key Press',{Number: $value})"
    med-type="keypad"
></med-keypad>
```

Attributes

- `med-point`: the point to monitor in the keypad display
- `med-change`: the action triggered when the 'Call' or 'Ok' buttons are pressed.
- `med-type` : the configuration of the buttons. The 'keypad' type displays only buttons 0–9, with a `Clear` and `Ok` button. The `dialer` type displays buttons 0–9, along with '.', '+', '#', '*', 'Clear', and 'Call' buttons. 'dialer' is the default type.

5.2.4.16 Tabs

Tabs are special containers that show or hide content based on which container is selected. It takes two tags to make them work, `med-tab-control` and `med-tab-content`.
`med-tab-control` creates the tab bar that has links to the containers. `med-tab-content` are the containers themselves.



Example

```
<med-tab-control></med-tab-control>

<med-tab-content med-tab-name="Lights" med-icon="ion-lightbulb">
    <med-point-table med-point="Room_2_Lighting"></med-point-table>
</med-tab-content>

<med-tab-content med-tab-name="Audio" med-icon="ion-levels">
    <med-point-table med-point="Room_2_Audio"></med-point-table>
</med-tab-content>

<med-tab-content med-tab-name="Video" med-icon="ion-play">
    <med-point-table med-point="Room_2_Player"></med-point-table>
</med-tab-content>
```

Attributes

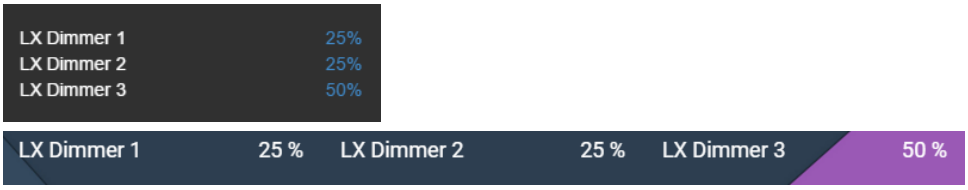
- `med-app` (in `med-tab-control`): This allows the tabs to only show up in a specific app. Options are

'magicmenu' or 'dashboard'. Shows in both, if not spcified.

- `med-tab-name` (in `med-tab-content`): The name of the tab. This is used to populate the tab control bar.
- `med-point` (in `med-tab-content`): Associates a tab to a point, giving it the same access rights as that point.
- `med-icon` (in `med-tab-content`): The icon to be displayed in the tab control bar.
- `med-auto-include` (in `med-tab-content`): If set to '1', the tab content will automatically be filled with the Control Panel link with the `med-point`. This is helpful when using `med-repeat` and you want to use `med-include` in the tab content.

5.2.4.17 Container

The `med-container` tag allows grouping of items that can be hidden/shown or organized based on the attributes supplied to it.



Example

This example shows a group of displays that will organize the displays horizontally based on the size of the screen.

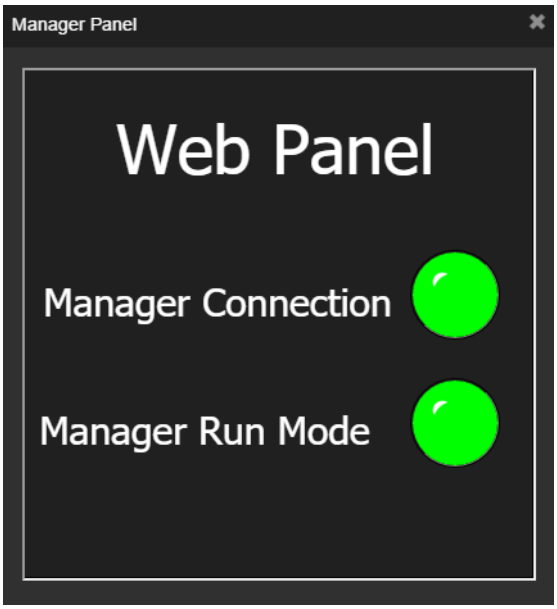
```
<med-container med-flex="1">
    <med-display med-point="Room_2_Lighting.Dimmer1Value"></med-display>
    <med-display med-point="Room_2_Lighting.Dimmer2Value"></med-display>
    <med-display med-point="Room_2_Lighting.Dimmer3Value"></med-display>
</med-container>
```

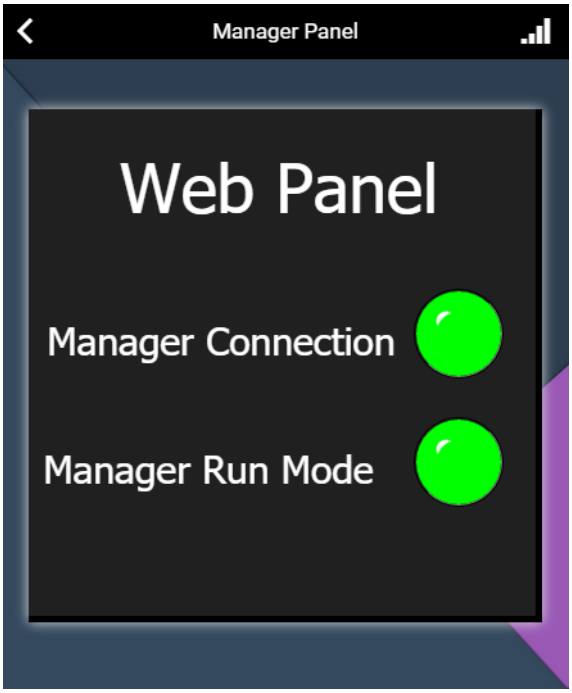
Attributes

- `med-point` : This assigns the container to the same roles as the point specified. If a user does not have read rights to this point, the contents of the container will not appear on their screen.
- `med-roles` : The specific roles that can see the contents of this container. Example: `med-roles="A/V,Building,IT"'. The name or the IDs of the roles may be specified.
- `med-app` : This allows the contents of the container to only show up in a specific app. Options are 'magicmenu' or 'dashboard'. Shows in both, if not spcified.
- `med-flex` : If true, groups items inside of it for automatic column display on tablets or mobile devices. Things specified within the tag are equally sized, and create a number of columns equal to how many of them fit on the page orientation. For example, a tablet in landscape might hold three columns, but when you go to portrait the items rearrange themselves into two columns.

5.2.4.18 Frame

The `med-frame` is a container that allows the user show another website within the container. This is similar to HTML `iFrame` tag but allows dynamic changing of the source. It is useful when displaying Manager Web Panels, or camera feeds into your control panels.





Example

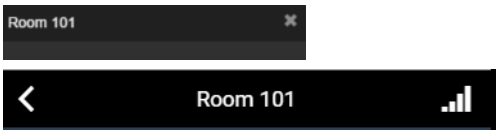
```
<med-frame
    med-src="http://localhost?panel=WebPanel_1"
    med-height="400"
    med-width="400"
></med-frame>
```

Attributes

- `med-src`: The web page to display in the container. Example "http://192.168.1.10/?panel=WebPanel_1".
- `med-height`: The height of the container in pixels. Defaults to 300px.
- `med-width`: The width of the container in pixels. Defaults to the width of the control panel or magic menu page.
- `med-point`: This assigns the container to the same roles as the point specified. If a user does not have read rights to this point, the contents of the container will not appear on their screen.

5.2.4.19 Header

This tag is used for creating a name at the top of the Magic Menu page or Control Panel. If no header is specified, the panel defaults to the name of the point.



Example

```
<med-header
    med-title="Room 101"
></med-header>
```

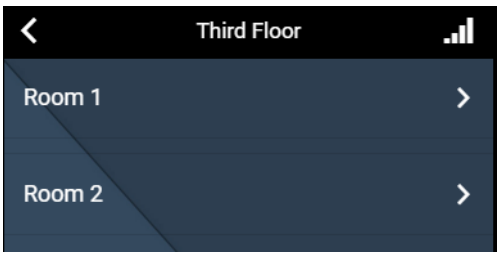
Attributes

- `med-title`: The name of the page displayed.

5.2.4.20 Page Link

Magic Menu Only

The `med-page-link` is a button which display a new page when clicked. This page must be an html document that exists in the views section of the assets folder.



Example

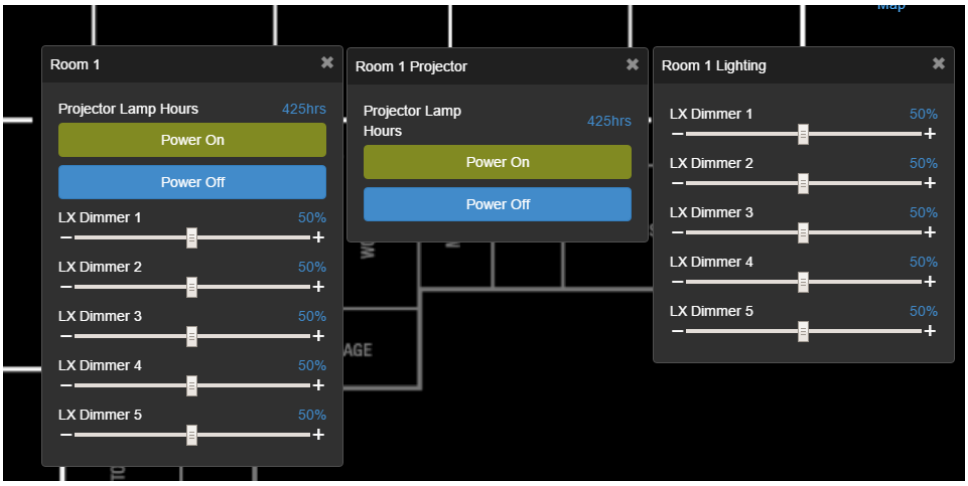
```
<med-page-link
    med-page="common/room"
    med-point="Room 1"
></med-page-link>
<med-page-link
    med-page="common/room"
    med-point="Room 2"
></med-page-link>
```

Attributes

- `med-label`: The text displayed in the button. Defaults to the name of the point.
- `med-page`: The HTML document you wish to load when the link is clicked. Must be format of `FOLDER/NAME` and no file type after. If not specified, the page loads the `med-point`'s HTML file associated to the 'Control Panel' section of the Configurator.
- `med-point`: The point associated with the page loaded. For example, if loading a projector html page, you can specify the projector you want to control in the link.
- `med-context`: Information you wish to pass into the new html page. Must be in form of an object `{item: value}`. This is an advanced feature used with creating templates.
- `med-roles`: The roles that will be able to see the page-link. These roles are in addition to the access rights provided if a `med-point` is specified. If no `med-roles` are specified, no extra access rights are added to the link.

5.2.4.21 Include

The `med-include` tag loads another HTML file in the control panel. It is useful for creating templates by allowing small portions of code to be reused in many panels.



Example

```
<med-include
    med-point="Room_1_Projector"
    med-file="common/proj"
></med-include>
<med-include
     med-repeat="{parent: point, subtype: 'lighting'}"
    med-file="common/lx"
></med-include>
```

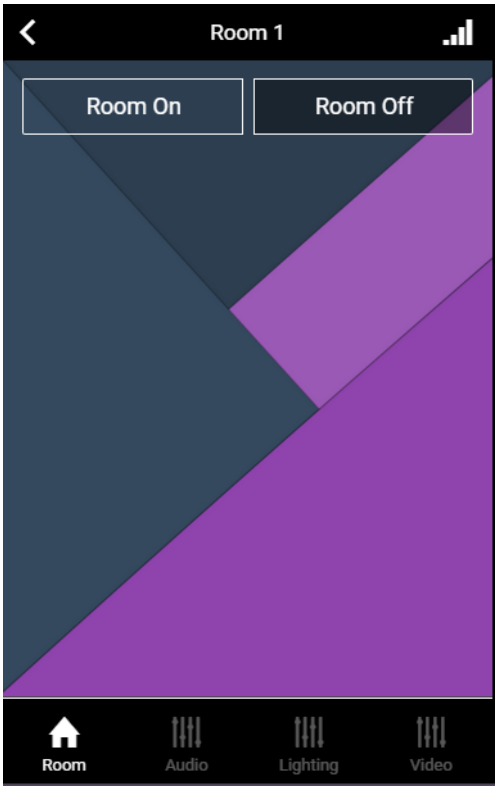
Attributes

- `med-file`: The HTML document you wish to include. Must be format of `FOLDER/NAME` and no file type after. If not specified, the page lincludes the `med-point`'s HTML file spcified in the 'Control Panel' section of the Configurator.
- `med-point`: The point associated with the page included. This is used to give context to the HTML. For example, if including projector html page, you can specify the projector you want to control in the link.
- `med-context`: Information you wish to pass into the new html page. Must be in form of an object `{item: value}`.

5.2.4.22 Footer

Magic Menu Only

The `med-footer` is a specialized include that will display it's content at the bottom of a magic menu page. This is used to create a bottom navigation bar, or always available style content like a slider.



Example

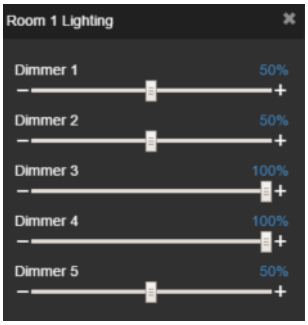
```
<!-----room.html------>
 <med-tab-content med-tab-name="Room" med-icon="ion-home">
    <med-on-off
        med-point="roomStatus"
        med-label-on="Room On"
        med-click-on="perform('Room_1_On', 'starttask')"
        med-state-on="1"
        med-label-off="Room Off"
        med-click-off="perform('Room_1_Off', 'starttask')"
        med-state-off="0"
    ></med-on-off>
 </med-tab-content>

 <med-tab-content
    med-repeat="{parent: point, type: 'device'}"
    med-auto-include="1"
    med-icon="ion-levels"
></med-tab-content>

 <med-footer med-file="pages/footer"></med-footer>

<!-----footer.html------>
    <med-tab-control ></med-tab-control>
<!--------------------->
```

Attributes

- `med-file`: the HTML document you wish to include. Must be format of `FOLDER/NAME` and no file type after.
- `med-height`: The height of the footer in pixels. If not specified, the footer has the default height of 44px.

5.2.4.23 Repeat

`med-repeat` is an attribute available for most tags. It is used to dynamically find points related to a filter, and repeat a tag a number of times equal to the number of points found.



Room 1 Lighting Dimmer1V...	Dimmer 1		Room_1_Lighting.Dimmer1...
Room 1 Lighting Dimmer2V...	Dimmer 2		Room_1_Lighting.Dimmer2...
Room 1 Lighting Dimmer3V...	Dimmer 3		Room_1_Lighting.Dimmer3...
Room 1 Lighting Dimmer4V...	Dimmer 4		Room_1_Lighting.Dimmer4...
Room 1 Lighting Dimmer5V...	Dimmer 5		Room_1_Lighting.Dimmer5...

Example

```
<med-slider
    med-repeat="{parent: point, variablename: '.Dimmer'}"
    med-min="0"
    med-max="100"
></med-slider>
```

Attributes

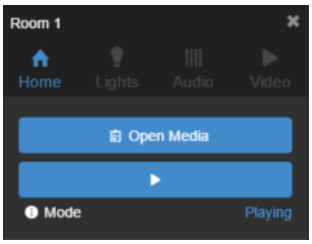
- `med-repeat`: the filter specified as an object. Atleast one filter item must be specified. Extra filter items should be comma seperated and further reduce the filter results (Uses AND logic instead of OR logic).

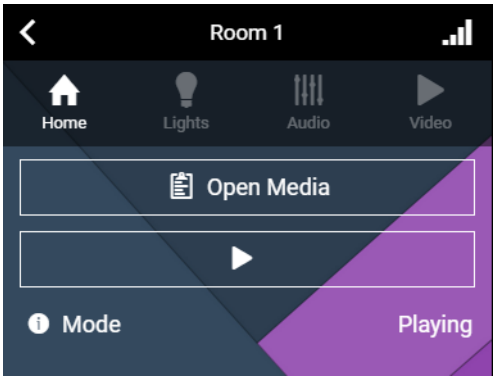
The following filter items are avilable

- `name`: select the points whose name contains the string spcified.
- `roles`: select the points which have all of the specified roles. Must be listed as an array. ex: `roles: ['Building','Energy']`
- `tags`: select the points which have all of the specified tags . Must be listed as an array. ex: `tags: ['Lights','Audio']`
- `variablename`: select the points whose variable name contains the string spcified.
- `type`: select the points whose type is specified.
- `subtype`: select the points whose sub-type is specified.
- `parent`: select points whose parent is specified.

5.2.4.24 Icons

Some tags have an `med-icon` attribute. This allows you to add icons next to information to help make your panels stand out. To add the icon, in the attribute specify the full name of the icon: `med-icon="ion-home"`. For a full list of useable icons, please use : http://ionicons.com/





Example

```
<med-tab-content med-tab-name="Home" med-icon="ion-home">
    <med-button
        med-label="Open Media"
        med-icon="ion-clipboard"
        med-click="perform(point, 'Open Media')"
    ></med-button>
    <med-button
        med-icon="ion-play"
        med-click="perform(point, 'Play')"
    ></med-button>
    <med-display
        med-point=".Mode"
        med-label="Mode"
        med-icon="ion-information-circled"
    ></med-display>
</med-tab-content>
```

6 Deployment

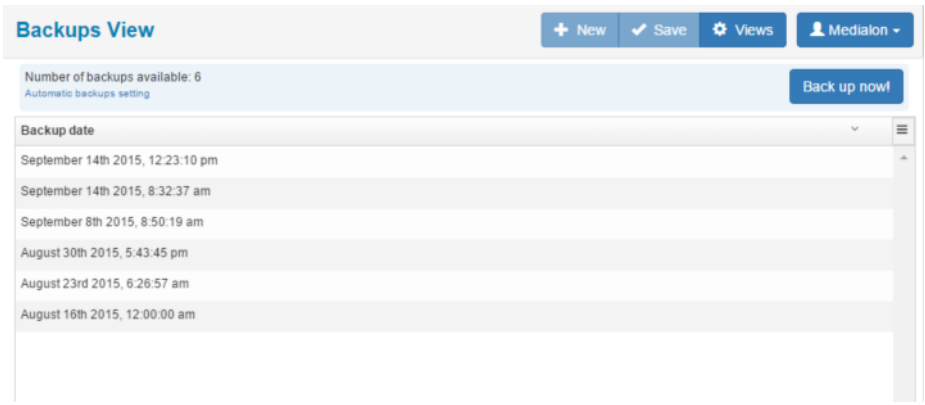
After the system has been programmed, Overture has some nice extra features to help build a better user experience within the system.

6.1 Backups/Restoring

UX Server has the ability to backup all of your data and allow restoration in case of data corruption or error.

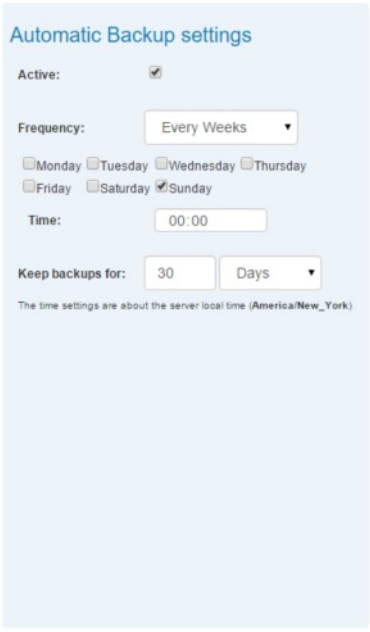
6.1.1 Backups

There are two ways to backup your data in the UX Server. The first way to backup your data is manually through the Configurator's 'Backups' view.



In this view, you can see the current list of backups in the file server as well as adding one manually by clicking 'Back Up Now'.

The other way is through the 'Automatic Backup Settings'. This is found in the 'UX Server Config' view in the Configurator.



Here you can specify, whether or not to backup, how frequent, and on what date and time to do the backup. You can also specify how long the backups should stay before the system deletes them.

The backups are stored in the `C:\ProgramData\Medialon\OvertureUX\backups` folder on the UX Server PC. They are zip folders that contain a complete copy of your assets folder, along with two restoration SQL files for your database.

6.1.2 Restoring

If you wish to restore to a previous UX Server, you will need two items that come in the backup zip folder you wish to restore from. First, take the assets folder from the backup and replace your current one. This is located where you designated during installation (`C:\ProgramData\Medialon\OvertureUX\assets` by default).

Next, place the two SQL files from the database folder of your backup into the `C:\ProgramData\Medialon\OvertureUX\dbutils` of the UX Server PC. This will replace the two exisiting files with the same name located there. Next open up a command prompt on your PC. First, find and type the location of the dbutility.exe from the installation (`C:\Program Files (x86)\Medialon\Medialon Overture UX Server\pg93\dbutils\dbutility.exe` by default). Next, make a space, and then find and type of the location of your database configuration file (`C:\ProgramData\Medialon\OvertureUX\dbutils\config.cfg`). Finally, type a space, and then the word 'restore'. Before hitting enter, your command prompt should look similar to the one below:

```
C:\>"C:\Program Files (x86)\Medialon\Medialon Overture UX Server\pg93\dbutils\dbutility.exe"
 "C:\ProgramData\Medialon\OvertureUX\dbutils\config.cfg" restore
```

Hit enter. The database will go through a series of commands that restore it back to the restore point provided by the SQL files you placed in the folder.

```
Getting configuration from C:\ProgramData\Medialon\OvertureUX\dbutils\config.cfg
Starting database restore...
Connecting to PostgreSQL using postgres://ovtuser:medialon@127.0.0.1/ovtmap
Database 'ovtmap' connected
Cleaning up ovtmap' database...
Database 'ovtmap' cleaned up.
Closing connection
Connecting to PostgreSQL using postgres://poadmin:medialon@127.0.0.1/ovtmap
Database 'ovtmap' connected
Updating database 'ovtmap'
Updating database 'ovtmap' succeed.
Closing connection
Connecting to PostgreSQL using postgres://ovtuser:medialon@127.0.0.1/ovtmap
Database 'ovtmap' connected
Populating 'ovtmap' database...
Adding Data to 'ovtmap' database...
Database 'ovtmap' populated.
Closing connection
Database restore completed.
```

6.2 Linking To Specific Maps Or Pages

Overture links can be sent to users that only allow them a certain layer of access.

With Magic Menu, this is a specific HTML page created. With Dashboard, this will be a specific map layer.

6.2.1 Magic Menu

Besides allowing only a certain level of access, Magic Menu links can be used to create different interfaces.
You may want mobile users to have the normal Magic Menu, but also want to create very specific in-room interfaces meant for tablets.

In that scenario you can write another HTML page (inRoom.html for example) and use the linking system provided below to only launch that page with the browser.

6.2.1.1 Page

You can access directly to a specific page using `home` parameter referencing the file after `views` folder.

For example, to go to a `room` page in `pages` folder, you can write:

```
http://localhost/magicmenu/?home=pages+room
```

6.2.1.2 Point context

You can give the context to a page using the `point` parameter. You could do this to gain direct access to a room even if that room is just a template.

For example to go to a `room` page in `pages` folder, using `Malaga_Room` as context, you can write:

```
http://localhost/magicmenu/?point=Malaga_Room&home=pages+room
```

6.2.2 Dashboard

Dashboard links reference map layers in the system. By providing a link, you will be bypassing a certain amount of map layers.

6.2.2.1 Access Control Links

You can change the default map for Dashboard by changing the URL you go to. If the map you specify is a child of another map, the user will not be able to access the parent map.

For example, if Fred doesn't care about any buildings in Campus A, but does care about the buildings in Campus B. We can direct him to the map of BuildingB1 for example.

```
http://localhost/?map=500
```

In that example, 500 is the id of the point that is tied to BuildingB1's map. Fred will now be able to access BuildingB1's control panels or switch to another building's map as long as that map is a sibling of BuildingB1. It is important to know that whatever map you link to, the user will have access to all sibling maps of that type as well. If you do not wish this behavior, you will need to specify a map one layer down.

6.2.2.2 Allowing Users To Travel Back

If you just wish to provide a direct path to a map layer, but want to allow the user access backwards through the chain you may specify a parameter known as 'showrootmap'.

Using the example above, if you would like Fred to have faster access to his Campus B buildings but not disallow him to get into Campus A you can give him the following link:

```
http://localhost/?map=500&showrootmap=1
```

6.3 Dashboard Extras

6.3.1 Widgets

Dashboard widgets are used to display information at a quick glance, or display data in a graph for easy comparisons. Each user is able to set up their own Dashboard widgets. Changing the definition of widgets in one user's login does not affect the widgets for other users.

By clicking "Add Widget", you can specify what type of widget you would like to see on the screen. Once chosen, you must decide what points to display with the widget. Each user is only able to specify points that he has 'read' access to. No other points are shown.

Table widgets are used to display information. If the point specified is also connected to a control panel , clicking the name of the point in the Table widget acts like a bookbark. The map jumps to where that control panel is and opens it up.

Graph widgets show information of points relative to each point. This is used when you need to track information against other information, like Projector Temperatures. Clicking on a point's name in a graph type widget hides that point's information.

6.3.2 Branding

Company branding can be added to the dashboard to make the project more unique. To do so, you must add an HTML file in the `assets` named `dashboard.branding.html`. This HTML is only meant to have images and basic styling.(No med-tags). The following is an example.

```
<img src="assets/images/Banner.jpg">
```

6.4 Overture Styles

During your programming, you may want to change the look and feel of the interfaces you are designing. This is achieved through CSS(Cascasding Style Sheets) programming.

6.4.1 Default Options

UX Server installs with two basic themes: magicmenu.css and dashboard.css. These CSS fles are located in the 'assets/styles' folder of your installation.

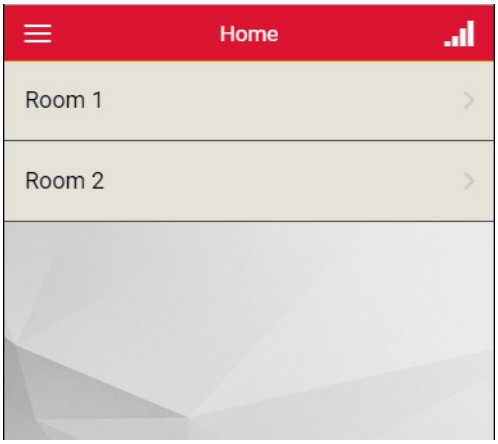
Some of the Dashboard and MagicMenu CSS is contained within the server programming. Those files cannot be overwritten. You may need to edit your HTML to fully customize the CSS to your liking.

You can edit these style sheets in the GUI Editor. You need to know which classes you will want to change in order to achieve the look you want. **NOTE**: It is recommended you use browser developer tools to assist you in finding classes, and previewing colors.

For example, to achieve the look above you could change `dashboard.css` with the following:

```
.btn-primary {
    background-color:#ED9206;
    border-color: #BD8C35;
}
.btn-primary.active, .btn-primary:active, .btn-primary:focus, .btn-primary:hover,
.open>.dropdown-toggle.btn-primary {
    background-color: #FF9B00;
    border-color: #ED9206;
}
.btn-primary.med-on {
    background-color: #228A25;
    border-color: #308A22;
}
.point-value {
    color: #ED9206;
}
.map-select-panel .map-selected {
    color: #ED9206;
}

#zoom container .landmarks .mark .text {
    background-color: #ED9206;
}
```

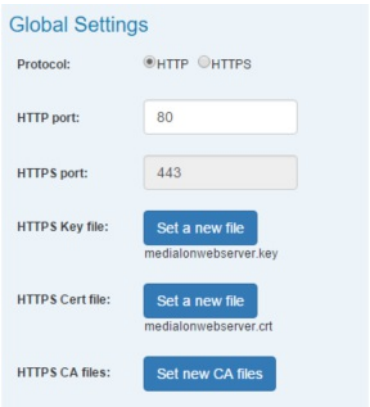
6.4.2 Themes



If you do not want to change the default options for your interfaces, but want to allow multiple looks for your interfaces you can create different themes on the system. To do this, simply save any CSS file in the styles folder with a relevant name. Then when you want to use that CSS, add '?theme=' and the name of your theme minus the .css in the URL for Dashboard or MagicMenu.

For example, if I want a more red theme to MagicMenu, you could write a red.css file and store it in the 'assets/styles' folder. Now when you want a user to see that theme, you can give them the link: `http://yourserver/magicmenu/?theme=red`

6.5 HTTPS



If you wish to switch communication over to HTTPS, you can do so from the Configurator.

In the UX Server Configuration view, you can switch the Key/Cert file as well as add your HTTPs CA files if needed.

7 Advanced

This section is for advanced programmers who are looking for things to help them specialize their programming. We recommend to read this section when you are comfortable with Overture core concepts.

Overture's advanced programming documentation will help you better use some of the frameworks that the User Experience Server was built on. These advanced features can be used when specialized panels are needed or to reduce some of the long-form writing.

7.1 Scope Items

Each HTML panel you write has access items available in the scope. In normal panel writing, the scope is automatically accessed by the tags you write. For example, `<med-display>` accesses the specified point and uses its 'name' and 'value' properties.

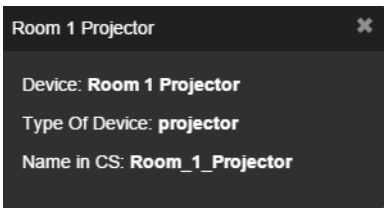
Items in the scope can be accessed directly in the HTML page by typing `{{}}`. For example, in a device control panel, the name of the device could be displayed with:

```
<p> {{point.name}}</p>
```

This is useful when you want to only display parts of the point, or when troubleshooting filtering.

7.1.1 Point Scope

As explained above, the points have their own variables that can be accessed directly.



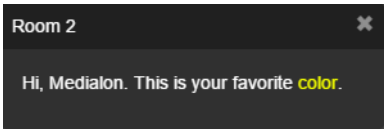
For example, if you wanted a more info based control panel, the following code could be written:

```
<p>Device: <b>{{point.name}}</b> </p>
<p>Type Of Device: <b>{{point.subtype}}</b></p>
<p>Name in CS: <b>{{point.variable_name}}</b></p>
```

The metadata items stored in the configuator are all accessible. Just remember that tags and roles return as an array.
Enum type variables also have a special property called `enumStrings` that returns an array filled with the names of the items in the enum.

7.1.2 User Scope

Inside of the scope, some fundamental items about the user are available as well.




```
<style>
    .favColor {
        color: {{user.preferences.favColor}};
    }
</style>
<p>Hi, {{user.name}}. This is your favorite <span class="favColor">color</span>.</p>
```

This allows panels/pages to be written with a more user centric approach. The four properties that can be reached are username, name, roles, and preferences.

Roles return an array with objects associated with the roles in the database. Preferences returns a JSON object from the database that's mostly used for display of the dashboard widgets. By altering the database object directly, you can add in some special properties (like the favColor above) and then use them in your code in unique ways.

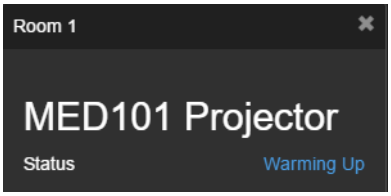
7.1.3 Using Context

Sometimes you you will wish to pass special information into a control panel. This "context" can be passed in two ways:

First, `med-include` and `med-page` link allow context to be passed in as a parameter.

```
<med-include
    med-point="Room_1_Projector"
    med-file="common/projector"
    med-context="{Model : 'MED101'}"
></med-include>
```

Second, in Magic Menu, you can pass it in via the URL: `html http://localhost/magicmenu/?Model=MED101`



Once you have passed the context to a control panel, you can retreieve and use it by finding using the `context` object. For example:

```
 <h2>{{context.Model}} Projector</h2>
<med-display
    med-point=".Status"
></med-display>
```

This would display the context, passed in so the same template could be differnt based on the context passed in via inclusion, page-link, or url.

7.2 AngularJS Items

The framework for providing HTML panels is AngularJS. As a result, some built in AngularJS attributes for tags are available for your use in creating panels.

7.2.1 ng-init

The `ng-init` is used to add items to the scope. This can be finding points from the database, or adding virtual variables that you may reference.

7.2.1.1 Example

```
<div ng-init="
    vol = findPoint({parent: point, variableName: '_Audio_Volume'});
"></div>
<med-display
    med-point="vol">
></med-display>
```

7.2.1.2 Direct Assignment

`ng-init` can be used to assign values directly or create virtual objects.


```
<div style ng-init="
    myPoint = {name : 'Created Point',  value: 100};
"></div>
<med-display
    med-point="myPoint"
></med-display>
```

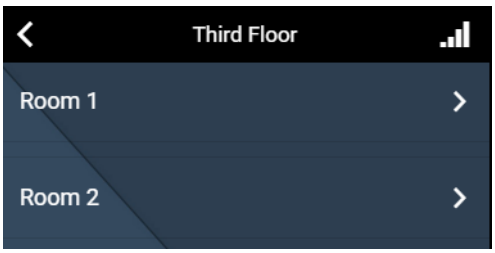
This allows you to store some information that isn't tied to a control server variable and use it while the panel is open. The value resets once the panel is closed, as this type of point is only used while the control panel is open. This behaves like a normal point, which means its value can be dynamically affected by `med-change` or `med-click` attributes.

7.2.1.3 Functions

`ng-init` also has a set of defined functions that can be used.

setPoint()

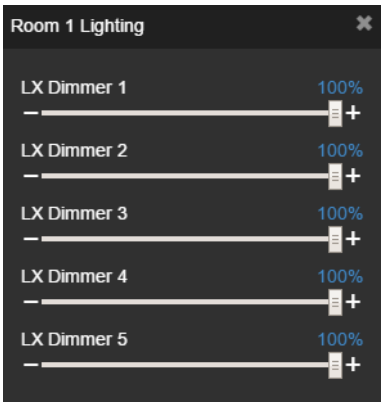
`setPoint()` is used in Magic Menu and sets the current page to refer to whatever point you specify. This is useful in giving `home.html` some type of beginning context.




```
<div ng-init="setPoint('Room 1');   "></div>
<med-page-link
    med-repeat="{parent: point, type: 'device'}"
></med-page-link>
```

findPoint / findPoints

The `findPoint()`/`findPoints()` functions are used to find a group or groups of points and store them into a variable used by the page. The `findPoint()` function only returns the first point object that matches the filter applied. The `findPoints()` function returns an array of points, that is used with `ng-repeat` to access each point in the array. These functions should be set up with the following format:
`ng-init="result= findPoint(filter);"` or `ng-init="result = findPoints(filter);"`
`result` is the name of the item added to the scope. For example, `myPoints = findPoints(filter)`. `filter` should be passed as an object with any filter parameters in it. For example `{parent: point, variablename: '_Audio'}`



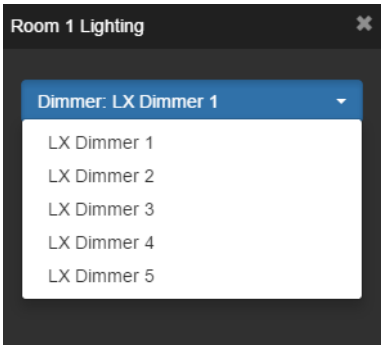

```
<div
    ng-init="dimmers = findPoints({parent: point, variablename: 'Dimmer'});
"></div>
<med-slider
    ng-repeat="dim in dimmers"
    med-point="dim"
></med-slider>
```

The following items can be used as filtering objects for both `findPoint` and `findPoints`.

- `parent` : the parent of the point(s) you want to find.
- `name` : the string contained in the human name of the the point(s) you want to find.
- `variablename` : the string contained in the variable name of the point(s) you want to find.
- `type` : the type of the point(s) you want to find. Refer to the types view in Configurator for available names.
- `subtype` : the sub type of the point(s) you want to find. Refer to the types view in Configurator for available names.
- `tags` : the tags associated to the point(s) you wish to find. Must be in an array Example: [tag1, tag2].
- `roles`: the roles associated to the point(s) you wish to find. Must be in an array Example: [role1, role2].

createEnum()

`createEnum` can be used to take an array of items, and then create a virtual enum with it. This allows that array to be put into a `med-select` without needing to manipulate the enumStrings property.




```
<div
    ng-init="dimmers = findPoints({parent: point, variablename: 'Dimmer'});
    dimmerEnum = createEnum(dimmers, 'short_name');
"></div>
<med-select
    med-label="Dimmer"
    med-point="dimmerEnum"
></med-select>
```

`createEnum()` has two items that it can take in. First is the array you wish to create the enum from. The second item is optional. It is the property from the points you wish to create the enumStrings from. If nothing is specified, the name property is used.

7.2.2 ng-repeat

The `ng-repeat` attribute can be used to iterate through an array. It will create elements equal to the length of the array. It allows for each element to have an item from the array passed into it.

7.2.2.1 Example

```
<div style ng-init="
           myColors = ['Red', 'Green', 'Blue']
"></div>

<div ng-repeat ="color in myColors"
    style="color: {{color}};">
    {{color}}
</div>
```

Attributes

- `ng-repeat`: must be specified in 'item in arrayName' format. The 'item' object represents the object in the array. The 'arrayName' object is the array you wish to itterate through.