



Customizing the Marketplace Portal

Contents

Introduction	2
Portal-wide Customization	2
Themes	2
Out-of-the-box Themes	2
Custom Themes	5
Portal Styling	6
Corporate or Organization Logo	7
Dashboard Customization	8
style property	9
Banner	9
Sections	12
Dynamic Section	15
Tiles	16
Tile Styling	17
Hot Keys	19
Widgets	19
Default widgets	19
Custom widgets	19
Using the IFrame Technique	28
Appendix A: Example Configuration File Format	31
Appendix B: Glyph Icons	32

Document Release Date: December 2014

Software Release Date: December 2014

© Copyright 2014 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Restricted rights legend: Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation. AMD is a trademark of Advanced Micro Devices, Inc. Intel and Xeon are trademarks of Intel Corporation in the U.S. and other countries. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Introduction

Your administrator can customize the display of the Propel Marketplace Portal to meet your organization's needs. We'll begin by looking at information that applies broadly to portal customization. Subsequent topics will address customization of specific areas of the Marketplace Portal.

Dashboard Configuration File

One of the primary configuration files involved in Marketplace Portal customization is the Dashboard configuration file: `/opt/hp/propel/mpp/conf/dashboard.json`. See **Appendix A: Example Dashboard Configuration File Format**.

Note: The **HP Marketplace Portal** service must be restarted after changes are made to the `dashboard.json` file. To do this, enter `service mpp restart` from the command prompt. Portal users must log out, clear their browser cache, and log in again to see the changes.

Glyph Icons

The Marketplace Portal styling framework provides a collection of glyph icons that can be used throughout the application. To reference these, use the appropriate HTML class names. The available icons are listed in **Appendix B: Glyph Icons**.

Custom images

Unless specified otherwise, store your custom Marketplace Portal images in `/opt/hp/propel/jboss-as/standalone/deployments/consumption.war/images/library/<imagefile>`.

To reference an image file named `pic.png` placed in this directory, the URL would be `https://<hostname>:<port>/consumption/images/library/pic.png`, and the html would be ``.

Portal-wide Customization

Themes

Themes define colors, fonts and the general look-and-feel of the Marketplace Portal. Select from out-of-the-box themes or create a custom theme.

Out-of-the-box Themes

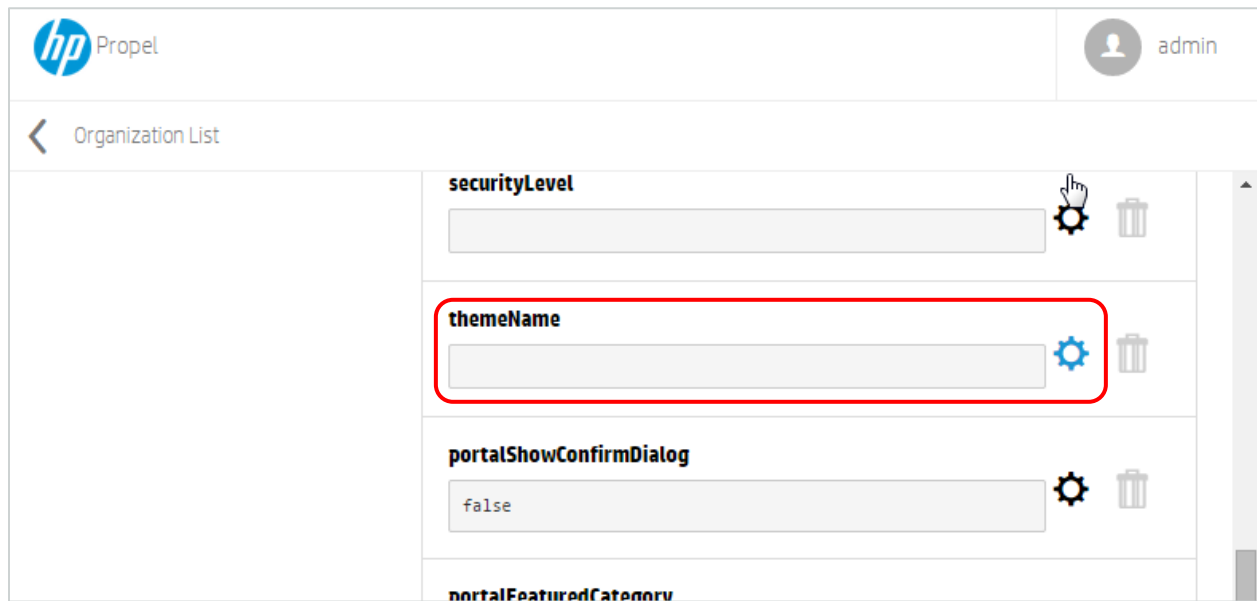
The following out-of-the-box, or pre-packaged, themes are available:

- HP Simplified
- HP Playful
- HP Enterprise

Change the selected theme as follows:

- Access the Organization Administration User Interface. (Select **Organizations** from the HP Propel Management Console.)
- Create a new organization or select an existing organization to modify.
- Select **Customization**.
- Enter the desired theme's name in the **themeName** box.
- Click on **Save**.

Figure 1: Selecting a theme



Following are example Dashboard screenshots of the three out-of-the-box themes.

Figure 2: HP Simplified

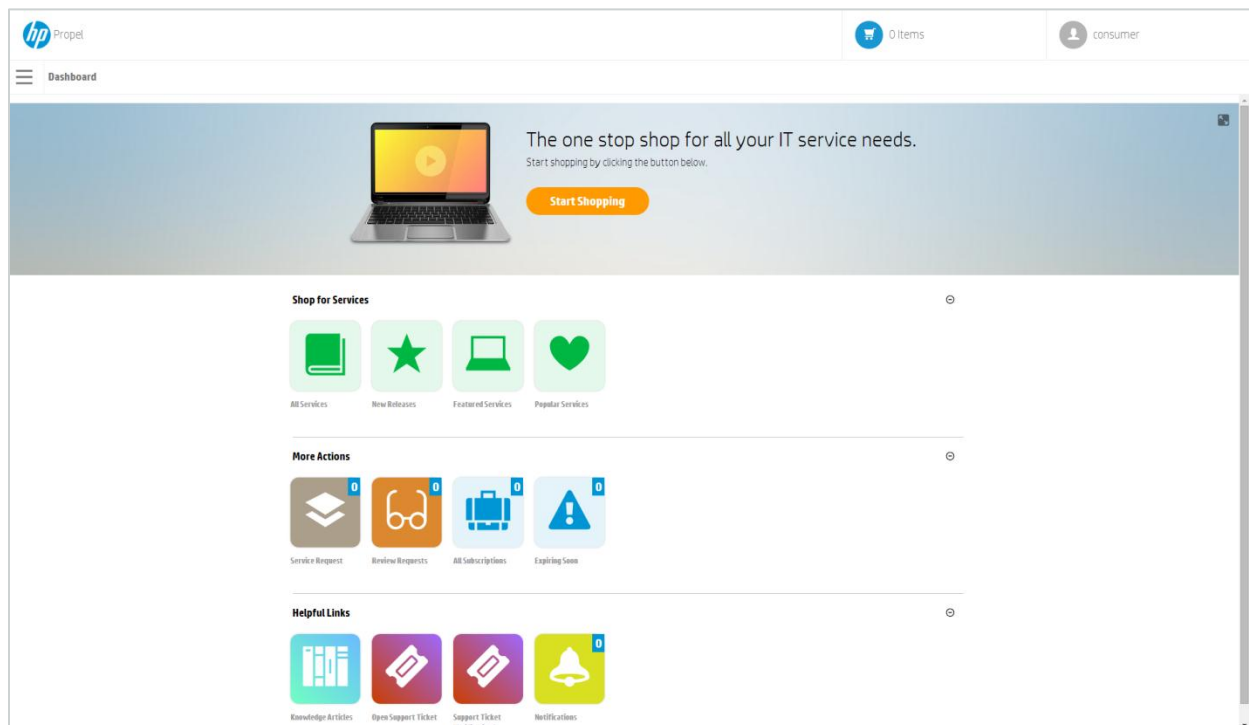


Figure 3: HP Playful

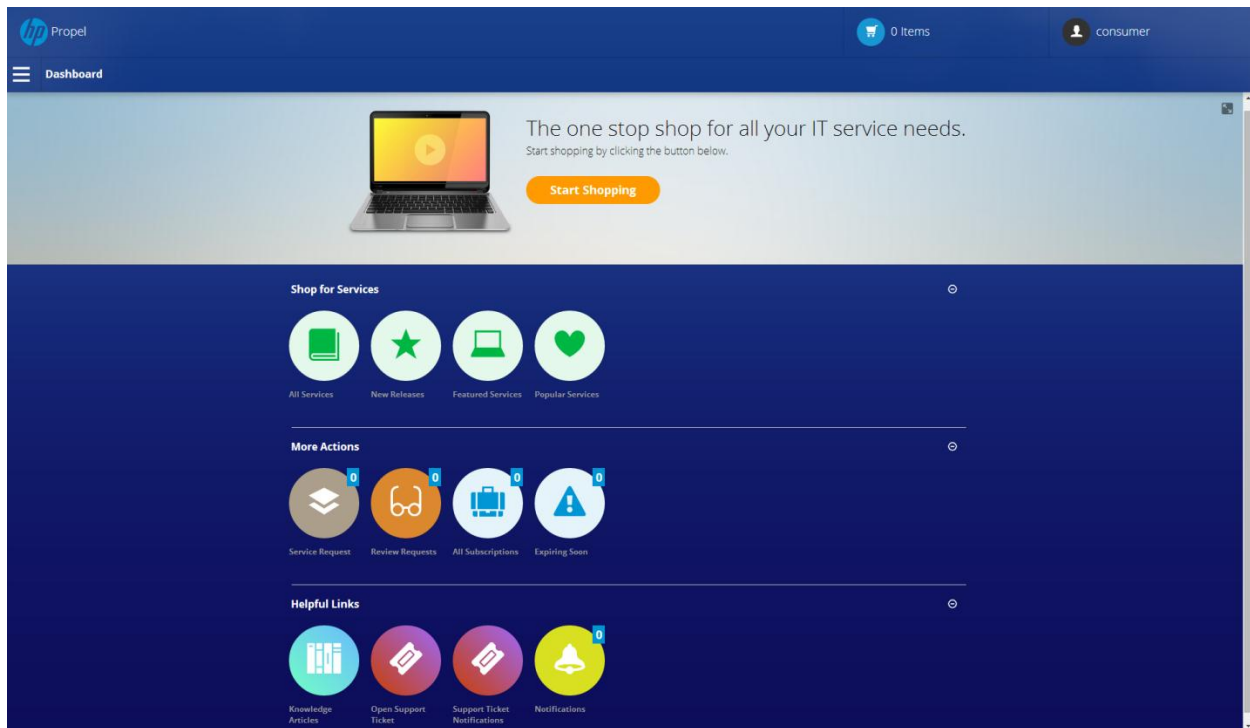
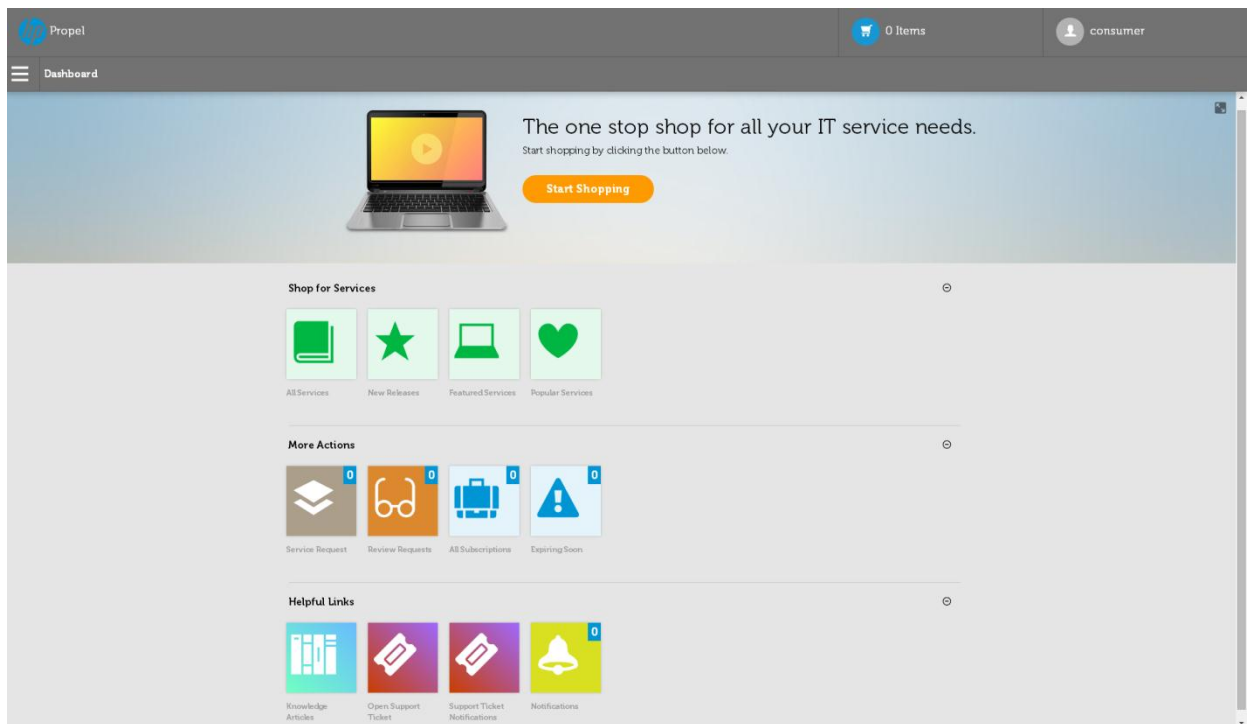


Figure 4: HP Enterprise



Custom Themes

In addition to out-of-the-box themes, HP Propel allows custom themes to be created. The administrator specifies the custom theme name (see Figure 1: Selecting a theme) and provides the content needed as described in the following.

The Marketplace Portal will look for the custom theme content in the following directory:

```
/opt/hp/propel/mpp/node_modules/mpp-ui/dist/bower_components/<THEME
NAME>/dist/
```

The login page (IDM Service) has its own theme content which can be found in the following directory:

```
/opt/hp/propel/jboss-as/standalone/deployments/idm-
service.war/bower_components/<THEME NAME>/dist/
```

Example: In the Propel Management Console, if **Custom** is entered for the custom theme name, the Marketplace Portal will look for the following files:

```
hp/propel/jboss-as/standalone/deployments/idm-
service.war/themes/Custom/styles/main.css
```

The Login page (IDM service) will also look for the following location:

```
hp/propel/jboss-as/standalone/deployments/idm-
service.war/themes/Custom/styles/main.css
```

Store your custom theme images in `/opt/hp/propel/mpp/node_modules/mpp-ui/dist/bower_components/<THEME NAME>/`. To reference an image file named `pic.png` placed in this directory, the html would be ``.

Best Practice: When creating a new theme directory, copy an existing theme directory, and then modify the new theme directory name and CSS files. To support customization, the CSS files must be named **main.css** and **main-rtl.css**.

Notes:

- Custom theme directories must model other theme directories. The directory **styles** contain all styling components, such as .CSS, images, and fonts.
- The CSS files **must be named main.css and main-rtl.css**. **main-rtl.css** is used for locales that read right-to-left.
- Out-of-the-box themes might be minified to reduce or exclude whitespace. When you duplicate these files you will want an IDE/tool to help unminify the CSS content to make it more readable. An example of this tool can be found at <http://mrcoles.com/blog/css-unminify>.

Theming Framework

The Marketplace Portal theming framework models the open source project Bootstrap at <http://getbootstrap.com>. Review the Bootstrap documentation to learn how the Marketplace Portal's components are structured with class labels.

All CSS rules used in Bootstrap are included in the main.css file for each out-of-the-box theme.

A CSS extension language used to create the framework is SASS (Syntactically Awesome Style Sheets), found at <http://sass-lang.com>. This extension language includes variables, nested rules, reusable mixins, and functions.

Portal Styling

The background and text color of the body, navigation bar, and sidebar of Marketplace Portal views can be customized through changes to the Portal CSS files. The Marketplace Portal will look for this content in the following directory:

```
hp/propel/mpp/node_modules/mpp-ui/dist/bower_components/<theme  
name>/dist/main.css
```

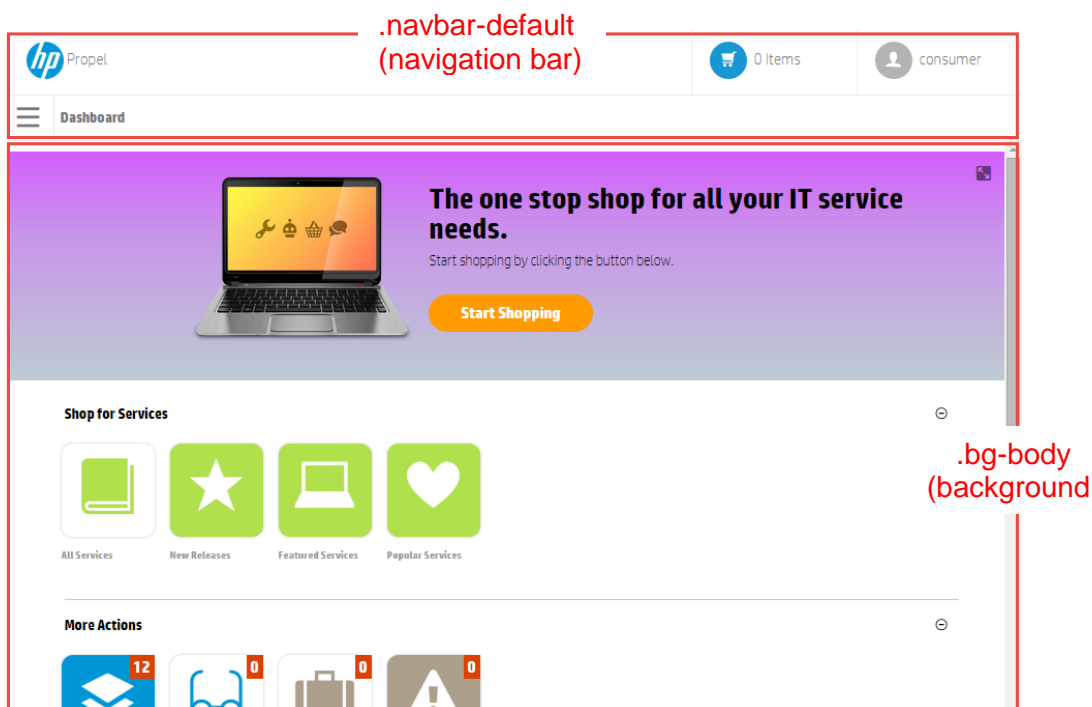
The Login page (IDM Service) has its own content which can be found in the following directory:

```
hp/propel/jboss-as/standalone/deployments/idm-service.war/themes/<theme  
name>/styles/main.css
```

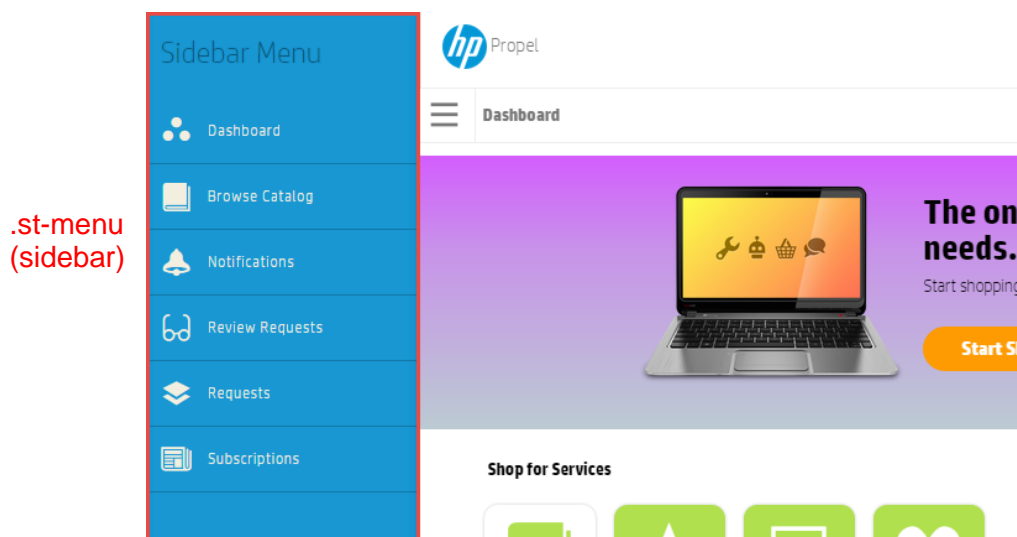
See the **Custom Themes** section for more information on these files.

The portal styling properties include:

- View body: **.bg-body**
- Navigation bar: **.navbar-default**



- Sidebar: **.st-menu**



In the CSS content, specify **background** for the background color/image and **color** for the text color.

Example CSS content:

```
// Specify a background color for the body ...
.bg-body {
    background: #000000 //black background
}

// ... or specify an image for the background.
.bg-body {
    background: url('/mpp-theme/images/skyscrapers.png') no-repeat cover;
}

// Specify text color for the sidebar menu.
.st-menu {
    color: #000000 //black text
}
```

Corporate or Organization Logo

A logo or picture for your corporation or organization can be used in the Marketplace Portal. It will be displayed on the portal login page, in the upper left-hand corner of Marketplace Portal views, and in Management Console views where appropriate.

Add a logo or picture as follows:

- Access the Organization Administration UI. (Select **Organizations** from the HP Propel Management Console.)
- Create a new organization or select an existing organization to modify.
- Select **General Information**.
- Enter the URL for a picture that represents the organization in the **Organization Picture URL** box.
- Click on **Save**.

Store your custom HP Propel organization logo image in `/opt/hp/propel/jboss-as/standalone/deployments/consumption.war/images/library/<imagefile>`. Reference

this image in the **Organization Picture URL** box as

`https://<hostname>:<port>/consumption/images/library/<imagefile>.`

The screenshot shows the 'Organization List' page in the HP Marketplace Portal. On the left, a sidebar lists 'Orange Organization' with a sub-menu containing 'General Information' (highlighted with a red box), 'Authentication', 'Customization', 'Groups', and 'Business Roles'. The main content area is titled 'General Information' and contains a description: 'Fill out the rest of your organizations profile. Provide information about your organization so that it is easily recognizable when referring back to it.' Below this are several input fields: 'Organization ID' (pre-filled with 'ORANGE_ORGANIZATION'), 'Organization Display Name' (pre-filled with 'Orange Organization'), and 'Organization Description' (empty). A red box highlights the 'Organization Picture URL' field, which contains the URL: `https://reeses.hpswlab.adapps.hp.com:8444/consumption/images/library/monitoring.png`. At the bottom, there are three buttons: 'Save' (blue), 'Discard Changes' (grey), and 'Delete Organization' (red).

Notes:

- The HP Corporate logo is not part of the out-of-the-box content. It is a custom image.

Dashboard Customization

Let's look at Marketplace Portal Dashboard banner, sections, tiles and widgets customization.

Dashboard banner, section and tile customization is achieved by modifying the Dashboard configuration file typically located at `/opt/hp/propel/mpp/conf/dashboard.json`. The file includes content for:

1. **header** (more commonly referred to as the "banner")
2. **sections**
3. **tiles**

A version of the `dashboard.json` file can optionally be customized for each organization. The filename should follow naming convention `dashboard.<organization name>.json`. For example, an organization named ACME would have a customized file named `dashboard.ACME.json`. The Marketplace Portal will automatically look for this file when a user in organization ACME logs in. If this file does not exist, the contents of `dashboard.json` will be used.

Roles The Dashboard can be configured to show selected tiles and sections only to users with specified roles. For example, the following could be included in the configuration file content for a tile or section:
`"role": ["CONSUMER_ORGANIZATION_ADMINISTRATOR"]`. See **Appendix A: Example Configuration File Format** for more details.

Note: The **HP Marketplace Portal** service must be restarted after changes are made to the `dashboard.json` file.

style property

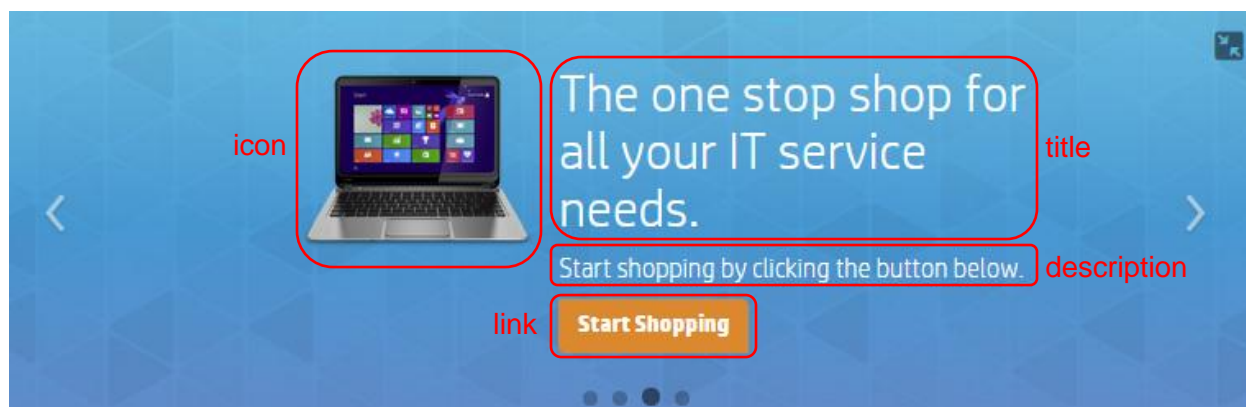
A **style** property is specified as part of each Dashboard element's configuration. Elements include header (banner), sections, and tiles, each of which is addressed in this document. A style property contains a space delimited string, for example, `className: "default my-selected-style"`. Style values must be predefined within the Marketplace Portal, and for visual appeal should be defined to work well with the overall styling of the portal.

Note: HP Propel supports only predefined styles. These will be described in detail in each topic as appropriate. Specific CSS styling is not supported.

Banner

Administrators can customize the Marketplace Portal Dashboard banner (also known as the “header”). Multiple banners can be defined. Each will be displayed for the specified time interval on a rotating basis. Customization is done by modifying the **header** content in the `dashboard.json` configuration file typically located at `/opt/hp/propel/mpp/conf/dashboard.json`. See **Appendix A: Example Configuration File Format** for more details.

Following is an example banner with properties labeled for easy identification.



Banner **header** configurations file properties:

Note: Configuration properties are typically optional. At least one entry in **items** must be defined for the banner to be useful. Values will be empty if **default** properties are not defined and individual **items** properties are also not defined.

Value Items	Description
default	Specifies the default value for the banner properties. Any items properties not specified will use these default values. Default: not specified.
Interval	If multiple banner items are defined, this value is the time in milliseconds to display one banner before switching to the next. Default: 5000 milliseconds.
items	Array or list of banners. Default: banner not displayed.
icon	Sub-property url will take precedence over className
className	Styling for the icon, for instance the icon color, or a predefined icon/glyph available in the Marketplace Portal. See Appendix B: Glyph Icons for more information.
url	URL reference to an image.
title	Title to be displayed on the banner. Default: “No data to show”.
description	Description to be displayed under the title. Default: no description

	displayed.
template	Template to use for the banner, either default or halfColumn . default uses all information defined for the banner. halfColumn ignores the icon property, and displays remaining content in the left half of the banner space. halfColumn display allows more of the background content to be viewed.
link	URL link, or a link from current page to another.
url	URL address of next page. Can be an internal or external link.
target	Optional. Web page or browser target for the URL link. null or undefined default to using the same page (“_self”). Possible values are: <ul style="list-style-type: none"> • _blank: Opens the link in a new browser window or tab. • _self: Opens the link in the current browser frame. • _parent: Opens the link in the browser parent frame. • _top: Opens the link in the full browser window. • <frameName>: Opens the link in the named browser frame.
label	Text label displayed on the link. Can be a static text string or a localized string predefined in the application resource bundle.
background	Specify one or more className values and optional image URL. Default: gray background.
className	<See list of valid values following this table.>
url	URL reference to an image. Can be an internal or external link.

Banner className values

The following predefined **className** values are available:

- Text colors:
 - **text-white**
 - **text-black**
 - **text-green**
 - **text-dark-blue**
 - **text-light-blue**
 - **text-lemon**
 - **text-rock**
 - **text-orange**
 - **text-red**
 - **text-apple-green**
 - **text-pink**
 - **text-light-red**
- Background colors:
 - **green**
 - **dark-blue**
 - **light-blue**
 - **lemon**
 - **orange**
 - **rock**
 - **white**
 - **black**
- Background images:
 - **bg-blur**
 - **bg-red-sky**
 - **bg-dots**
 - **bg-success**

- **bg-warning**
- **bg-critical**
- **bg-1** through **bg-21** – collection of background images
- Other background styling:
 - **background-stretch** – Stretch background image as needed to fill banner background. Requires that a background image be specified.
 - **background-repeat** – Repeat the background image horizontally and vertically to fill the area. Requires that a background image be specified.
 - **background-cover** – Scale the background image to cover the background. Note that some of the image might not be visible, depending on the image dimensions.

Example configuration file content format:

```
header: {
  default: Object,
  interval: Number, //in milliseconds
  items: [
    {
      "icon": {
        "url": String
        "className": String
      },
      "title": String,
      "description": String,
      "template": String,
      "link": {
        "label": String,
        "url": String"
        "target": String"
      },
      "background": {
        "className": String
        "url": String
      }
    },
    ...
  ]
}
```

Example configuration file content with three banners that each display for 5 seconds before switching to the next banner.

```
{
  "header": {
    "interval": 5000,
    "default": {
      },
    "items": [
      { //The first banner uses assets available in the portal, and uses the
        default template.
        "icon": {
          "url": "mpp-theme/dist/mpp-theme/images/laptop.png"
        },

```

```

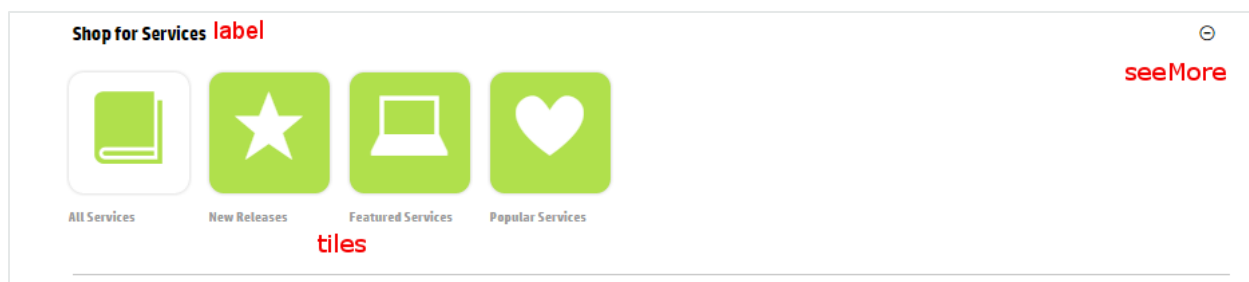
        "title": "shop.banner.description",
        "description": "shop.banner.subDescription",
        "link": {
            "label": "shop.banner.link",
            "url": "#/shop"
        },
        "background": {
            "className": "text-black bg-red-sky background-stretch"
        }
    },
    { //The second banner uses mostly external assets, and custom template
      "halfColumn".
        "icon": {
            "url": "http://sprout.hp.com/images/desktop/desktop-m2-product.png"
        },
        "title": "meet sprout",
        "description": "Blending the physical and digital worlds that you
live in, Sprout unleashes your creativity like never before. Assets come from
http://sprout.hp.com. This shows the page can use publicly available
assets.",
        "template": "halfColumn",
        "link": {
            "label": "See DEMO",
            "target": "_blank",
            "url": "http://sprout.hp.com/#"
        },
        "background": {
            "url": "http://sprout.hp.com/images/desktop/desktop-m2-bg.jpg",
            "className": "text-black"
        }
    },
    { //The third banner is empty. Uses default setting with title "No data
to show".
    }
  ]
}
...
}

```

Sections

Sections are horizontal containers for tiles on the Dashboard. A section can contain an infinite number of tiles. If all the tiles cannot be shown in the given view area of the section, a horizontal scrollbar will appear. Sections can be collapsed or expanded by clicking the section name.

Following is an example section with several properties labeled for easy identification.



The label, background color, tiles and seeMore label/link are all customizable.

Section configurations file properties:

Note: Configuration properties are typically optional. At least one entry must be defined under **tiles** for the section to be displayed. Values will be empty if **default** properties are not defined and individual **tiles** properties are also not defined.

Value Items	Description
type	Section type. Used only when the section is predefined. Can be: <ul style="list-style-type: none"> Default – type defined as empty or not declared at all. Specify other property values. Pre-defined – typically used to show service offerings in the section. No other properties need to be specified as this category is predefined. See category description that follows. Widget – section contains custom widgets. Note that these will appear as large tiles on the Dashboard. See example section types code following this table.
name	Specify CATALOG_CATEGORY or WIDGET .
category	Used only when name is CATALOG_CATEGORY . Select from: FEATURED_CATEGORY , NEW_RELEASES , MOST_REQUESTED , or CATEGORY_NAME .
label	Section label displayed on Dashboard. Can be a static text string or a localized string predefined in the application resource bundle. Default: no label.
seeMore	Clickable location on the right side of the section. By default this will be a plus or minus sign allowing the user to expand (show) or collapse (hide) the section content. Further customization is possibly by specifying sub-property: String representation , Object representation , or url value .
String representation	A simplified format for the link which contains the seeMore link URL Default attributes will be used for target and label .
Object representation	Allows more customization of the link property.
url	URL the seeMore link references. Can be an internal or external link.
target	Optional. Web page or browser target for the url link. null or undefined defaults to using the same frame (“_self”). Possible values are: <ul style="list-style-type: none"> _blank: Opens the link in a new browser window or tab. _self: Opens the link in the current browser frame. _parent: Opens the link in the browser parent frame. _top: Opens the link in the full browser window. <framename>: Opens the link in the named browser frame.
label	Optional. Label displayed on the section seeMore button.
url value	URL. Can be to an internal or external location.
internal	Navigates to a subpage of the portal (staying on same page/tab in the

	<p>browser). Can link to a specified subpage or to a pre-defined section as shown here:</p> <ul style="list-style-type: none"> Subpage examples: (Will be prepended with HP Propel instance location, typically <code>https://<propelhostname>:8089.</code>) <ul style="list-style-type: none"> <code>/notification</code> <code>/myservice</code> Pre-defined section – navigates to the specified pre-defined content: <ul style="list-style-type: none"> <code>FEATURED_SERVICES</code> <code>NEW_RELEASES_SERVICES</code> <code>SUBSCRIPTION_EXPIRING</code>
external	Navigates to an external page (staying on same page/tab in the browser). Link should start with <code>http://</code> or should define the protocol of the link so browser handles the behavior automatically.
tiles	List of tiles that will appear in the Dashboard for this section. See the Tiles section in this document for details on tiles properties.

Example section types:

- Default:**

```
"sections": [
  {
    ...
    "type": {}
    ...
  }
]
```
- Pre-defined:**

```
"sections": [
  {
    "type": {
      "name": "CATALOG_CATEGORY",
      "category": "FEATURED_CATEGORY | NEW_RELEASES | MOST_REQUESTED |
[CATEGORY_NAME]"
    }
  }
]
```
- Widget:**

```
"sections": [
  {
    "type": {
      "name": "WIDGET",
      "category": "WIDGET"
    }
  }
]
```

Example section configuration file content format:

```
sections:[
  {
    label: "Example section"
    tiles:{
```

```

    default:{
      style: "green border-square"
    },
    items:[
      {
        label: "All Services"
        icon: {
          className:
        },
        url:
      },
      {...}
    ]
  }
}
]

```

Dynamic Section

You can configure a Dynamic Section on the Dashboard by specifying the section type attribute as one of following:

- **FEATURED_CATEGORY** - Loads the services as tiles under featured category.
- **NEW_RELEASES** - Loads the new release services as tiles.
- **MOST_REQUESTED** - Loads the most requested services as tiles.
- Existing Category Name - Loads the services in the specified category as tiles.

The relevant tiles will be added to the section dynamically, updating this content each time the Dashboard is refreshed. Node.js will detect that this is a dynamic section and make the relevant call to load the services (depending on the **type** specified), create the appropriate tiles, and append them to the dynamic section.

Note: You can include a specific, fixed set of tiles in a dynamic section. The dynamically added tiles will be appended to that fixed set of tiles.

Example Dynamic section configuration file content format:

```

sections: [
  { // Tiles will be loaded dynamically on the Node.js depending on the
    section type ;
    type: FEATURED_CATEGORY | NEW_RELEASES | MOST_REQUESTED | [CATEGORY_NAME]
    // Pre-defined section
    label:
    seeMore: {
      url:
      [label]:
      [target]:
    }
    tiles: [ // Optional ; For a Dynamic section the appropriate tiles will
    be appended it to this list
      {
        label:
        icon: {
          url:
        },
        url:
      },
      {

```

```

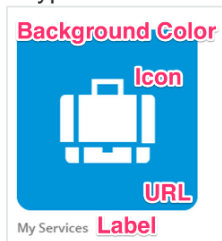
    label:
    icon: {
      url:
    },
    url:
  }
  ...
]
},
...
]

```

Tiles

Tiles are clickable icons on the Dashboard. They can contain basic information on a feature, a shortcut to another page, or something user defined (mashup widget). The background color, icon, label, and associated URL are all configurable. The URL can be an internal or external link.

A typical tile – with properties labeled for clarification – looks as follows:



Count tile

A count tile includes a small counter in the upper right-hand corner of the tile. The count represents the number of items included in the group this tile represents. For example, if this tile linked to a collection of favorite service offerings, of which there were 12, the count on the tile would look as follows:



Tile configuration items:

Note: Configuration properties are typically optional. **label** and **link** should be defined for the tile to be useful. Values will be empty if **default** properties are not defined and individual **items** properties are also not defined.

Value Items	Description
default	Specifies the default values for items properties styling. This should follow the object structure of items . When a tile is defined, any items property not specified will use the default value. Default: not specified.
items	Collection of tile configurations
label	Label displayed on Dashboard for this item. Can be a static text string or a localized string predefined in the application resource bundle. Default: no label.
icon	Specify either sub-property className or url .
className	Predefined icon/glyph available in the application. See Appendix B :

	Glyph Icons for more information. Note that predefined font styles can also be applied, e.g. <code>text-white</code> , <code>text-black</code> . Default: the <code>icon-catalog</code> icon will be displayed.
url	URL reference to an image.
link	Specify either sub-property String representation or Object representation .
String representation	A simplified format for the link which contains the tile link URL. Default attributes will be used for target and label . Default: tile will not have any link capability.
Object representation	Allows more customization of the link property.
url	URL the link references. Can be an internal or external link.
target	Web page or browser target for the url link. null or undefined defaults to using the same frame ("_self"). Possible values are: <ul style="list-style-type: none"> • <code>_blank</code>: Opens the link in a new browser window or tab. • <code>_self</code>: Opens the link in the current browser frame. • <code>_parent</code>: Opens the link in the browser parent frame. • <code>_top</code>: Opens the link in the full browser window. • <code><iframe></code>: Opens the link in the named browser frame.
label	Text label displayed for the link. Can be a static text string or a localized string predefined in the application resource bundle. Default: no label.
count	Counter to be displayed in upper right corner of tile. Defaults to red background with white text.
url	API endpoint of where to get the count value. Is called each time the user refreshes/visits the page.
className	Space delimited predefined CSS style class name.
hotkey	Hotkey or keyboard shortcut for activating the tile. Results in the same behavior as clicking on the tile. Default: no hotkey.
role	Limits users who can view this tile. See Roles for more information.

Tile Styling

Default `className` properties for tiles are:

- **white** : white background
- **text-dark-blue**: dark blue text/icon
- **corner-rounded**: rounded corners
- **shadow-light**: border shadow
- **icon-catalog**: catalog icon

Available `className` values

The following predefined tile styles are available.

Note: Multiple `className` values can be specified, and are separated by one or more spaces.

- Tile background colors:
 - **white** (default background color)
 - **black**
 - **green** (example follows with **text-white** icon)



New Releases

- dark-blue
 - light-blue
 - lemon
 - rock
 - orange
 - sky
 - bloody
- Tile icon (“text”) colors:
 - **text-green** (example follows with **white** background)



All Services

- text-white
 - text-black
 - text-green
 - text-dark-blue (default text color)
 - text-light-blue
 - text-lemon
 - text-rock
 - text-orange
 - text-red
 - text-apple-green
 - text-pink
 - text-light-red
- Tile shadows:
 - shadow-default
 - shadow-light
 - shadow-dark
 - shadow-none
 - Tile corners:
 - corner-rounded
 - corner-square
 - corner-circle

Example tile configuration file content format:

```
//dashboard.json tile configuration
{
  label: "My Services",
  link: "http://someurl.com",
```

```
icon: {
  className: "icon-briefcase"
},

style: "text-dark-blue" // Specify dark-blue icon for tile.
style: "border-rounded light-blue" // Specify rounded corner border and
light-blue background for tile.
}

//html effect of configuration for styling
<tile ng-style="border-rounded blue" class="border-rounded blue"> ... </tile>
```

Hot Keys

A hot key can be defined to activate a tile without using a mouse click. Assign a string key value to the **hotkey** property in the tile content of the dashboard configuration file as shown in the following:

```
{
  label:
  link:
  role:
  hotkey: "ctrl+<selected string key>" // "ctrl+j", for example
  image:
  icon:
  style:
  backgroundImage:
}
```

Widgets

Let's look now at customizing widgets.

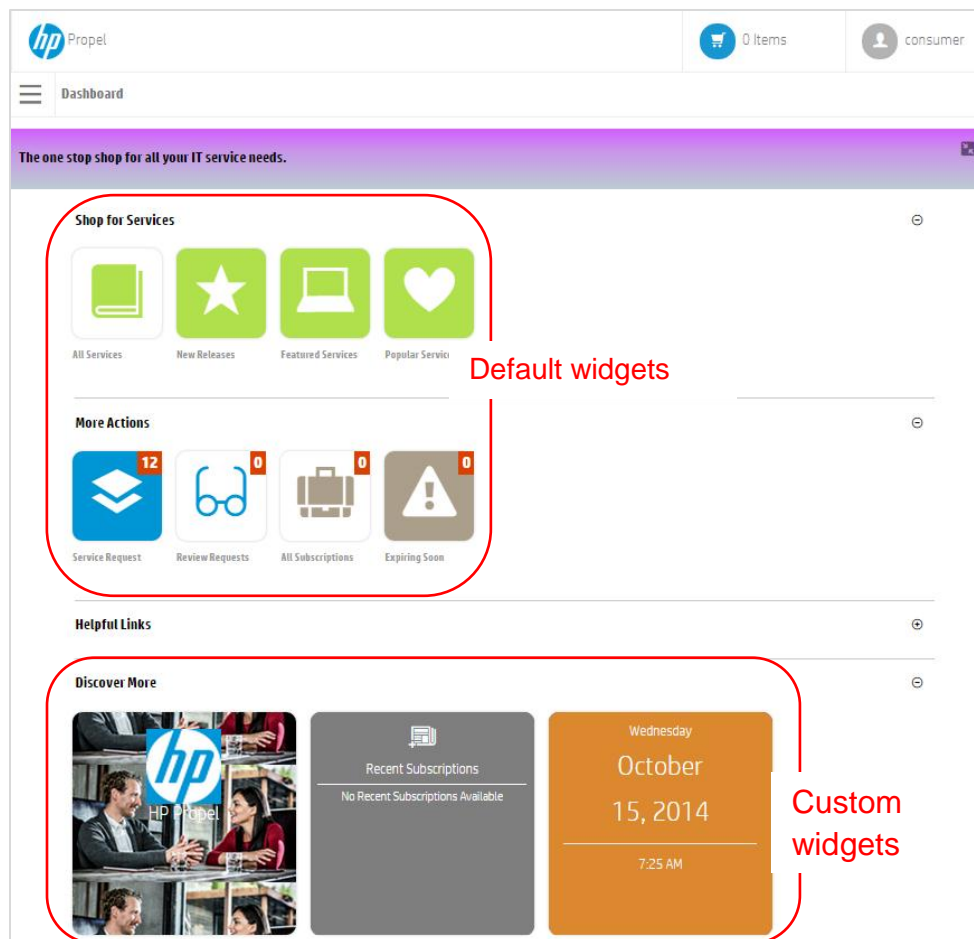
Default widgets

Default widgets are standard in HP Propel and are displayed in the first two sections of the Dashboard. They can be modified through the dashboard.json file, and are configured per installation, not per organization.

Note: The **HP Marketplace Portal** service should be restarted after changes are made to the dashboard.json file.

Custom widgets

Your administrator can add custom widgets to the Dashboard, such as the calendar, recent subscriptions, and HP link widgets shown in the following image. Widgets cannot be added to other pages or components in the Marketplace Portal, such as in the Sidebar Menu.



The Marketplace Portal supports the following types of custom widgets:

- Link Widget
- Featured Service Widget
- Mashup Widget

The size of a widget in the Dashboard is fixed. Note that it is recommended that you not change the size of even complex mashup widgets as unexpected behavior can result.

Custom widgets are defined per organization under **Customization** in the **Organization** area of the HP Propel Management Console as shown in the following, and will be available to all users in that organization. Click on the **Add KeyPair** button at the bottom of the Customization screen to create new widgets. See the *HP Propel Organizations Help* for more information.

hp Propel

Organization List

hp Consumer

General Information
Authentication
Customization

Customization

Add and label KeyPair values to customize various aspects of this organization. Please refer to the reference guide to see the different aspects of the organization you can control.

Edit KeyPair values by selecting them

portalTitle

Propel

widget:PropelLink

```
{
  "type": "LINK",
  "url": "http://www.hp.com/go/propel",
  "displayName": "HP Propel",
  "iconUrl": "http://www.www8-hp.com/us/en/images/i/header-footer/caas-hf-v3/hp-logo.gif",
  "backgroundImageUrl": "http://www.www8-hp.com/us/en/images/Community"
}
```

widget:RecentSubscriptions

```
{
  "type": "MASHUP",
  "content": "<div id=\"recentSubs\" class=\"full-size\"><b><h3><i class=\"icon-subscription-add\"></i></h3><h5>Recent Subscriptions</h5><b><ul id=\"list_recent_subscriptions\"></ul></div><script type=\"text/javascript\">var $list = $('#list_recent_subscriptions');function getSubscriptions(options) {$$.ajax({type: \"GET\",url: \"api/subscription?count\"
}
```

widget:Clock

```
{
  "type": "MASHUP",
  "content": "<div style=\"background-color:#DC882F;background-image:url(''); color: #FFFFFF;font-family: HPSimplified; font-size: 18px;background-repeat: no-repeat; padding-left: 20px;padding-top: 20px;padding-right: 20px;padding-bottom: 20px;line-height: .7;height:100%;\"><p id=\"weekday\" style=\"font-size: 18px;\"></p><h1 id=\"month\" style=\"font-size: 18px;\"></h1></div>"
}
```

Add KeyPair

Note that there are entries for three custom widgets corresponding to the three custom widgets in the previous Dashboard image.

Creating Widgets

All widget types are created in HP Propel as key:value pairs in the Customization section of the Organization Administration tab.

The key, or name, provided must be prefixed with **widget:** in order to identify this key pair as a widget. The value of the key pair will always be a JSON object and specific values required will depend on the widget type as detailed in following sections.

Figure 5: Create KeyPair Example

Create KeyPair

Provide an easily readable display name for the KeyPair.

Name

Value

```
{
  "type": "MASHUP"
  "content": "<span>Hello World</span>"
}
```

9,934 characters left

Save Cancel

Link Widget

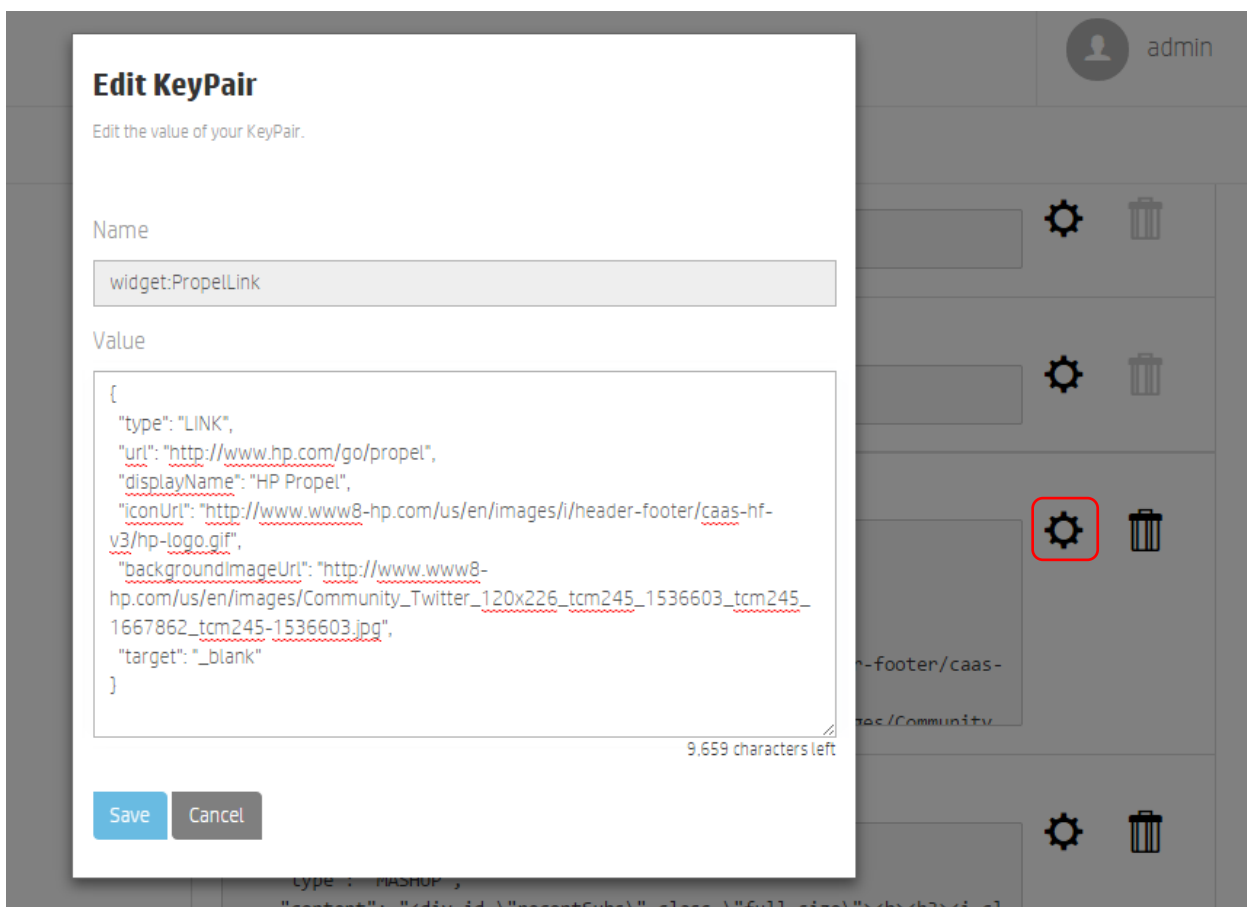
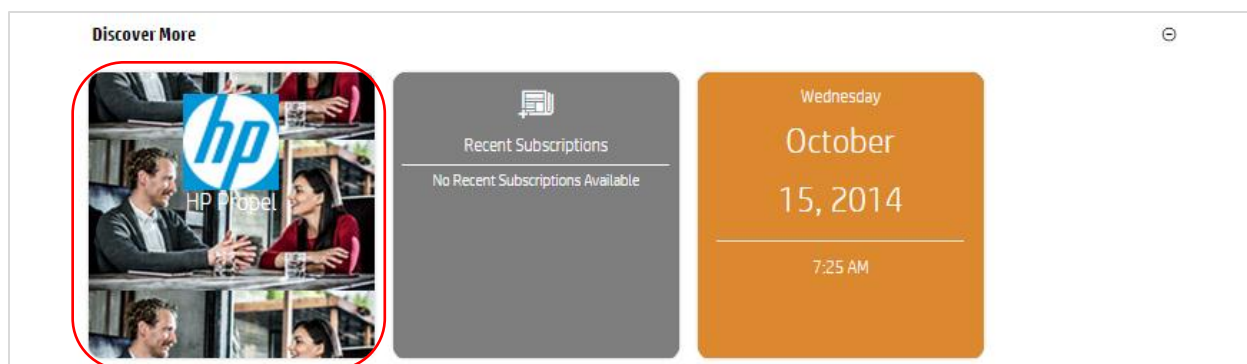
A link widget is a quick-and-easy shortcut to access a URL from the Marketplace Portal Dashboard. As with all custom widgets, link widgets display in the bottom row of the Dashboard.

Use the **Create KeyPair** dialog to create a custom link widget. The **type** value must be set to **LINK**. With link widgets you customize the link URL, display name, icon image, background image, and the HTML target.

Value Items	Description
type	Must be set to LINK
url	URL that the link references in the Marketplace Portal.
displayName	Optional name to be displayed on the widget.
iconUrl	Optional URL of an icon that displays near the center of the widget.
backgroundImageUrl	Optional URL of an image that fills the background of the widget.
target	Optional target attribute of the <link> element that appears in the Marketplace Portal and that controls the browser window in which the link will open. Valid values for the target attribute are defined in the HTML specification.

Tip: If new to creating widgets, review existing custom widgets on the Dashboard and associated details in the **Edit KeyPair** dialog to help in creating new widgets.

Following is the content for the custom link widget on our Dashboard.



Featured Service Widget

During organization creation and configuration, any category of services can be set as a featured category. Service offerings published under that category are treated as featured service offerings and can be viewed by selecting the default **Featured Services** widget in the **Shop for Services** area of the Dashboard.

You can further highlight these service offerings by creating a featured service custom widget. The featured service custom widget selects a random service from the featured services list and provides a link for easily viewing that service's details. The service selected will randomly change each time the Dashboard is refreshed.

Note: The Marketplace Portal does not support pinning a specific featured service to this custom widget.

Use the **Create KeyPair** dialog to create a custom feature service widget. The **type** value should be set to **FEATURED**. There is no other JSON content for Featured Service widgets.

Value Items	Description
type	Must be set to FEATURED .

Create KeyPair

Provide an easily readable display name for the KeyPair.

Name

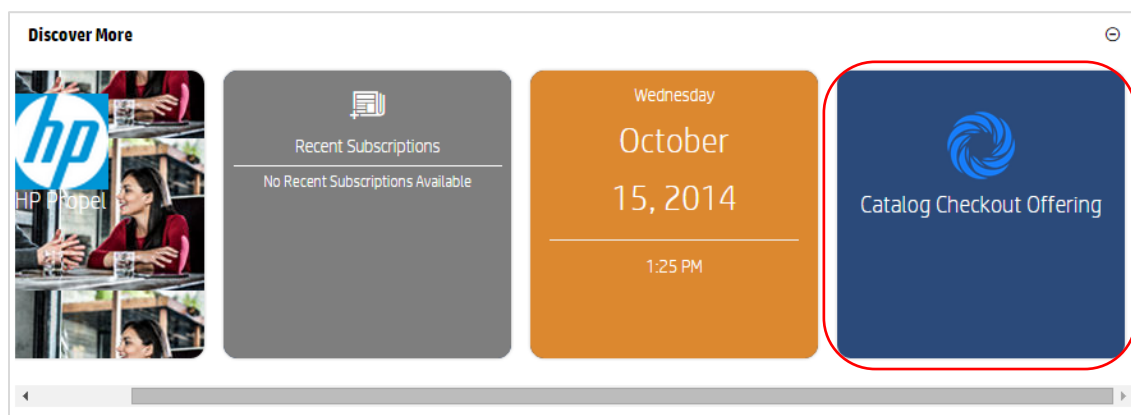
Value

```
{
  "type": "FEATURED"
}
```

9,974 characters left

Save Cancel

We now see a fourth custom widget in the Dashboard:



Mashup Widget

Mashup widgets give administrators the ability to define more complex widgets to meet an organization's needs. These might be generally useful widgets such a calendar or clock, or perhaps widgets for HP Propel-specific information such as recent subscriptions as shown in several examples in this document.

Mashups widgets require **type** be set to **MASHUP**. Mashup widgets contain one other key:value pair, **content**, which contains the source for your mashup.

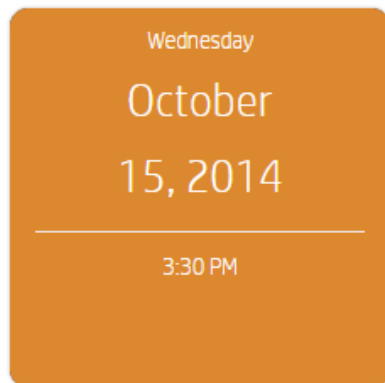
Note: All double quotation marks (") must be escaped (\") in the mashup widget content.

Value Items	Description
type	Must be set to MASHUP .

content	<p>The HTML and JavaScript code for the mashup.</p> <p>When using iFRAME in a mashup widget, note the following:</p> <ul style="list-style-type: none"> iFrames that serve HTML pages that have the same URL structure as the Marketplace Portal will work properly. The same URL structure means that the pages are placed in the following Marketplace Portal directory: <code>/opt/hp/propel/mpp/node_modules/mpp-ui/dist</code> <p>For example, to correlate to the following URL structure: <code>https://server:8089/widgets/sample/index.html</code></p> <p>You would place your pages in the following location: <code>/opt/hp/propel/mpp/node_modules/mpp-ui/dist/widgets/sample/index.html</code></p> <ul style="list-style-type: none"> iFrames that serve external NON-HTTPS content will be blocked by the browser. The specific error will vary based on client browser security. iFrames that serve external HTTPS content that contains mixed HTTP and NONHTTPS content will be blocked by the browser. The specific error will vary based on client browser security. iFrames that serve external HTTPS content will work only if the following are true: <ul style="list-style-type: none"> The remote site must not specify x-frame-options DENY in the response header. If the content is not of the same origin domain, and the remote site has not specified x-frame-options SAMEORIGIN, the content will display properly.
---------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example custom widget: Clock

```
{
  "type": "MASHUP",
  "content": "<div style='background-color:#DC822F;background-image: url(''); color:
#FFFFFF;font-family: HPSimplified; font-size: 18px;background-repeat: no-repeat; padding-left:
20px;padding-top: 20px;padding-right: 20px;padding-bottom: 20px;line-height: .7;height:100%;'><p
id='weekday' style='font-size: 18px;'></p><h1 id='month' style='font-size: 36px;'><
</h1><h1 style='font-size: 36px;'><span id='day'></span>, <span id='year'></span></h1><hr
style='color: #FFF;'><p style='font-size: 18px;'><span id='hmm'></span> <span
id='ampm'></span></p></div><script type='text/javascript'>function startClock() { var months
= [ \"January\", \"February\", \"March\", \"April\", \"May\", \"June\", \"July\", \"August\",
\"September\", \"October\", \"November\", \"December\" ]; var weekdays = [\"Sunday\", \"Monday\",
\"Tuesday\", \"Wednesday\", \"Thursday\", \"Friday\", \"Saturday\"]; function updateClock() { var
now = new Date(); var hours = now.getHours(); var minutes = now.getMinutes(); var ampm = \"AM\";
if (Math.floor(hours/12)==1) { ampm = \"PM\"; hours = hours - 12; } if (hours == 0) { hours = 12;
} var ms = \"\" + minutes; if (ms.length == 1) { ms = \"0\" + ms; } try {
document.getElementById(\"weekday\").textContent = weekdays[now.getDay()];
document.getElementById(\"month\").textContent = months[now.getMonth()];
document.getElementById(\"day\").textContent = now.getDate();
document.getElementById(\"year\").textContent = now.getFullYear();
document.getElementById(\"hmm\").textContent = hours + \":\" + ms;
document.getElementById(\"ampm\").textContent = ampm; } catch (err) {
clearInterval(clockInterval); } } updateClock(); var clockInterval = setInterval(updateClock,
1000);startClock();</script>\"}
```

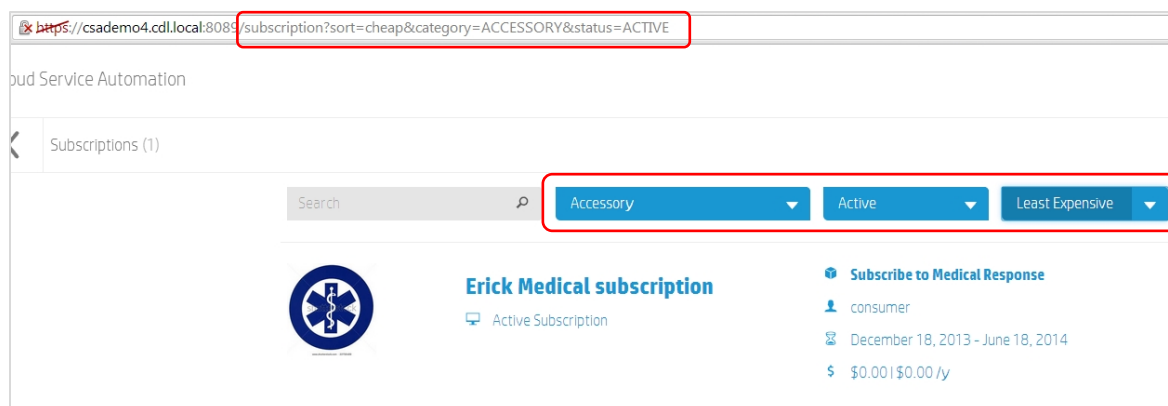


Example HP Propel-specific custom widget: 5 Most Recent Subscription

```
{
  "type": "MASHUP",
  "content": "<div id=\"recentSubs\" class=\"full-size\"><b><h3><i class=\"icon-subscription-
add\"></i></h3><h5>Recent Subscriptions</h5></b><ul
id=\"list_recent_subscriptions\"></ul></div><script type=\"text/javascript\">var $list =
$('#list_recent_subscriptions');function getSubscriptions(options) {$.ajax({type: \"GET\",url:
\"/api/subscription?count=5&offset=0&sort=newest\",success: function(data, textStatus, jqXHR) {if
(options.success) {options.success(data);}},error: function(jqXHR, textStatus, errorThrown)
{console.log(\"Widget Error: Failed to retrieve 5 recent subscriptions\");}}});}function
htmlEncode(value){return $('<div/>').text(value).html();} function htmlDecode(value){return
$('<div/>').html(value).text();}function render() { $list.empty();getSubscriptions({success:
function(data) {if (data && data.length > 0) {for (var i = 0; i < data.length; i++) {var name =
data[i].name;var id = data[i].id;var url = \"'/subscription/'\" + id;if(i < data.length-
1){ $list.append('<li><a class='ellipses widget-label' target='_parent' href='\" + url +
\"'><p>\" + htmlEncode(data[i].name) + \"</p></a></li>\"');}else{ $list.append('<li><a
class='ellipses' target='_parent' href='\" + url + \"'>\" + htmlEncode(data[i].name) +
\"</a></li>\"');}} } else { $list.append('<li>No Recent Subscriptions Available</li>\"');}}
});render(); </script><style type=\"text/css\">#recentSubs {font-
family:HPSimplified,Helvetica,Arial,sans-serif;background-color: #7e7e7e;padding: 10px;color:
white;font-weight: 500;}#recentSubs > b > h5 {margin: 0px;padding-bottom: 10px;border-bottom: 1px
solid white;font-weight: 500;}#recentSubs > b > h3 {margin-top: 5px;}#recentSubs > ul {padding-
left: 0px;margin: 0px;}#recentSubs > ul > li {display: block;padding-top: 5px;padding-bottom:
5px;border-bottom: 1px solid rgba(255,255,255,0.5);}#recentSubs > ul > li:last-child {border-
bottom: none;}#recentSubs > a {color: white;text-decoration: none;}#recentSubs > ul > li > a > p
{margin: 0 0 5px;word-wrap:normal;}#recentSubs > a:visited {color: white;}#recentSubs > a:hover,
#recentSubs > a:focus {color: #1897d3;}</style>\"
}
```

Note: The key function in the above HP Propel-specific widgets is the url: `"/api/subscription?count=5&offset=0&sort=newest&status=Active"`.

The options to `"/api/subscription?"` can also be seen in the URL when the Subscriptions page is visited using the Marketplace Portal, as shown below.



Limitations and Requirements

From a performance perspective, additional widgets will take more time to retrieve the associated data. This will result in slower rendering of the Marketplace Portal Dashboard. This impacts actions such as:

- Login time.
- Refreshing the Dashboard view.
- Returning back to the Dashboard.

When you create custom widgets, be aware of the following limitations and requirements:

- Variables are shared by widgets. If you are using the same variable name, e.g. list, in multiple widgets, unexpected behavior will result if "list" is assigned different values in the widgets' content.
- Widgets cannot be shared between organizations. Widgets must be created for each organization.

Additional Ways to Customize Widgets

There are several ways to customize widgets in the Marketplace Portal. As long as you adhere to the security restrictions around running external scripts on HTTPS you can implement widgets anyway you want. However, some approaches are simpler than others.

For convenience, custom widgets have access to jQuery and SugarJS. When you use the `jquery.ajax()` function to make calls to the Marketplace Portal APIs, the user session authorization token will be automatically included. You do not need to extract the user data from the session.

Optionally, you can extend this code to include an inline `<style>` tag for more complex CSS cases:

```
</script><style type=\"text/css\">#expensiveSubsDiv>p{color:black}</style>
```

When you append a style tag, wrap your widget in a top level div, such as `<div id=\"expensiveSubsDiv\">`. This will ensure help that your styling does not contaminate elements outside of your widget.

Using jQuery, combined with SugarJS, should make developing widgets relatively simple, even when you use data from HP Public APIs, as shown in the example above. However, there are limitations when running external scripts in an HTTPS environment. Do not expect to be able to copy and paste an iframe from a widget-generating website when running in an HTTPS environment. This is not supported at the browser level.

Using the IFrame Technique

Due to limitations of widgets you might find that trying to store your entire mashup widget in the **content** value of the body can be quite a challenge when having to escape (\) quotation marks in your source, e.g. "<div id=\"recentSubs\">...". Using the IFrame technique will enable you to create much more complicated widgets with much less frustration.

For example:

```
{
  "type": "MASHUP",
  "content": "<iframe src=\"widgets/IFrameSample/index.html\"
height=\"100%\" width=\"100%\"></iframe>"
}
```

This is an example of code that will leverage the IFrame technique in a mashup widget. It's important to note that the height and width of the IFrame should be set to 100% to fill the space available within the widget. You are still required to escape quotation marks within the **content**; however the issue is significantly reduced by moving the bulk of your source code out of the widget's content in the Management Console's **Organization Customization** area, and onto your HP Propel system. The location for your files is: /opt/hp/propel/mpp/node_modules/mpp-ui/dist.

In the **dist** folder you can create a subfolder for your custom widgets. In the example above this subfolder was named **widgets**. The related files for this one widget were placed into subfolder **IFrameSample**. Add your .html, .css & .js files and reference the index.html file in your IFrame src attribute as shown above.

Note: When using the IFrame technique you will not have access to any scripts used by the Marketplace Portal (including jQuery, sugarJS & AngularJS), so you will need to reference a public version of any scripts or libraries your widget uses, or more ideally, search the mpp-ui/dist/bower_components folder and reference available libraries there.

Jquery: <script src="/bower_components/jquery/dist/jquery.min.js"></script>

SugarJS: <script src="/bower_components/sugar/release/sugar.min.js"></script>

AngularJS: <script src="/bower_components/angular/angular.min.js"></script>

MPP-Theme: <link rel="stylesheet" href="/bower_components/mpp-theme/dist/main.css">

You will have access to the HP Propel specific APIs as long as your IFrame source code is on the same domain as the Marketplace Portal (that is, as long as the source code for the IFrame is in the portal dist folder listed above)

Following is the content of the html, css & js files related to this sample IFrame widget.

widgets/IFrameSample/index.html:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css" />
  <!--You can optionally use the MPP-Theme styles by uncommenting the
following line -->
  <!-- <link rel="stylesheet" href="/bower_components/mpp-
theme/dist/main.css"> -->
```

```
<script src="/bower_components/jquery/dist/jquery.min.js"></script>
</head>
<body>
  <script src="script.js"></script>
  <div id="recentSubs">
    <b>
      <h5>
        Recent Subscriptions
      </h5>
    </b>
    <ul id="list">
    </ul>
  </div>
</body>
</html>
```

Widgets/IFrameSample/style.css

```
body {
  background-color:#7e7e7e;
  padding:10px;
  color:white;
  font-weight:500;
}
h5 {
  margin:0px;
  padding-bottom:10px;
  border-bottom:1px solid white;
  font-weight:500;
}
ul {
  padding-left:0px;
  margin:0px;
}
ul > li {
  display:block;
  padding-top:10px;
  padding-bottom:10px;
  border-bottom:1px solid rgba(255,
    255,
    255,
    0.5);
}
ul > li:last-child {
  border-bottom:none;
}
a {
  color:white;
  text-decoration:none;
}
a:visited {
  color:white;
}
a:hover,
```

```
a:focus {
  color:#1897d3;
}
```

widgets/IFrameSample/script.js:

```
function getSubscriptions(options) {
  $.ajax({
    type: "GET",
    url: "/api/subscription?count=5&offset=0&sort=newest",
    success: function (data, textStatus, jqXHR) {
      if (options.success) {
        options.success(data);
      }
    },
    error: function (jqXHR, textStatus, errorThrown) {
      console.log("Widget Error: Failed to retrieve 5 recent subscriptions");
    }
  });
}

function htmlEncode(value) {
  //create an in-memory div, set its inner text(which jQuery automatically encodes)
  //then grab the encoded contents back out.  The div never exists on the page.
  return $('<div/>').text(value).html();
}

function htmlDecode(value) {
  return $('<div/>').html(value).text();
}

function render() {
  getSubscriptions({
    success: function (data) {
      var $list = $('#list');
      $list.empty();
      if (data && data.length > 0) {
        for (var i = 0; i < data.length; i++) {
          var name = data[i].name;
          var id = data[i].id;
          var url = "/subscription/" + id;
          if (i < data.length - 1) {
            $list.append("<li><a class='ellipses widget-label' target='_parent' href='" + url + "'><p>" + htmlEncode(data[i].name) + "</p></a></li>");
          } else {
            $list.append("<li><a class='ellipses' target='_parent' href='" + url + "'>" + htmlEncode(data[i].name) + "</a></li>");
          }
        }
      } else {
        $list.append("<li>No Recent Subscriptions Available</li>");
      }
    }
  });
}
```

```
render();
```

Appendix A: Example Configuration File Format

Following is example format of content that could be included in the Dashboard configuration file found at `/opt/hp/propel/mpp/conf/dashboard.json`.

```
{
  header: {
    label:
    image: // Image takes precedence when both image and icon is set
    icon: {
      className: //predefined icon/glyphs available in the app
      url: //internal or external url to the image
    },
    description:
    link: { //see header section below for more info
      url:
      [target]: // url target for the link
      [label]:
    },
    style: // space delimited predefined styling class
    backgroundImage: // Image takes precedence when both backgroundColor and
backgroundImage is set
  },
  sections: [
    {
      // Custom Section
      label: // used for customized section
      seeMore: { //see section details below for more info
        url: // specify internal or external url
        [label]:
        [target]:
      },
      style: // space delimited predefined styling class
      role: // list of strings. See "Server Design" below for more detail
      tiles:
        default: {
          style: // space delimited predefined styling class
        },
        items: [
          {
            label:
            link: { //see tile details below for more info
              url: // specify internal or external url
              [label]:
              [target]:
            },
            role: // list of strings. See below for more detail
            hotKey: // hotkey to activate the tile. When used it behaves the
same way as clicking the tile
            image: //image takes precedence if image and icon are both
defined.
            icon: {
              className: //predefined icon/glyphs available in the app
              url: //internal or external url to the image
            }
          }
        ]
      }
    ]
  }
}
```

```













































































































        },
        style: // space delimited predefined styling class
        backgroundImage: // backgroundImage takes precedence to what is
set on the style
        count: { // optional
            url: // specify an URL endpoint where it would return a count
            style:
        }
    }
    ...
]
}
},
{ // Tiles will be loaded dynamically on the Node.js depending on the
section type ;
    [type]: {
        name: [CATALOG_CATEGORY | WIDGET],
        [category]: [FEATURED_CATEGORY | NEW_RELEASES | MOST_REQUESTED |
[CATEGORY_NAME] // only relevant if name is set to CATALOG_CATEGORY
    },
    label:
    seeMoreLabel:
    seeMoreUrl:
    tiles: [ // Optional ; In case of pre-defined section it will
automatically load the appropriate tiles and append it to this list
        {
            label:
            icon: {
                url: //internal or external url to the image
            },
            link:
        },
        {
            label:
            icon: {
                url: //internal or external url to the image
            }
            link:
        }
        ...
    ]
    },
    ...
]
}

```

















































































































Appendix B: Glyph Icons

The Marketplace Portal styling framework provides a collection of glyph icons that can be used throughout the application. To reference these, use the HTML class names shown in the following:
















































































































Marketplace Portal Widget Customization

 icon-zoom-out	 icon-zoom-in	 icon-youtube	 icon-wifi
 icon-web	 icon-view	 icon-video	 icon-validation-denied
 icon-validation-approve	 icon-validation-alert	 icon-user	 icon-user-delete
 icon-user-add	 icon-unlock	 icon-twitter	 icon-tv
 icon-trophy	 icon-travel	 icon-trash	 icon-tools
 icon-toolbox	 icon-ticket	 icon-target	 icon-tag
 icon-tablet	 icon-surround	 icon-subscription	 icon-stop
 icon-status	 icon-star	 icon-stamp	 icon-spell-check
 icon-speaker-mute	 icon-speaker-medium	 icon-speaker-max	 icon-speaker-low
 icon-shuffle	 icon-shield	 icon-settings	 icon-settings-verticle
 icon-settings-horizontal	 icon-services	 icon-server	 icon-search
 icon-scissors	 icon-rss	 icon-rocket	 icon-rewind
 icon-repeat	 icon-relationship	 icon-refresh	 icon-record
 icon-radar	 icon-quote-start	 icon-quote-end	 icon-prohibited
 icon-process	 icon-print	 icon-power	 icon-play
 icon-phone	 icon-pending	 icon-payment	 icon-pause
 icon-page	 icon-page-new	 icon-page-multiple	 icon-owner
 icon-options	 icon-notification	 icon-network	 icon-navigation
 icon-mystuff	 icon-mug	 icon-mouse	 icon-monitor
 icon-microphone	 icon-microphone-mute	 icon-menu-expand	 icon-menu-collapse
 icon-mail	 icon-loop	 icon-logout	 icon-login
 icon-lock	 icon-location	 icon-locale	 icon-loading
 icon-list	 icon-link	 icon-like	 icon-lightbulb
 icon-library	 icon-legal	 icon-leaf	 icon-laptop
 icon-labs	 icon-keyboard	 icon-key	 icon-information
 icon-inbox	 icon-image	 icon-hybrid	 icon-hp
 icon-home	 icon-heart	 icon-hd	 icon-hd-raid

Marketplace Portal Widget Customization

 icon-hammer	 icon-grid	 icon-globe	 icon-git
 icon-gift	 icon-fox	 icon-folder	 icon-flame
 icon-flag	 icon-first-aid	 icon-finder	 icon-filter
 icon-fast-forward	 icon-facebook	 icon-export	 icon-export-option
 icon-expand	 icon-emoticon-surprised	 icon-emoticon-sad	 icon-emoticon-happy
 icon-ellipses	 icon-edit	 icon-drop	 icon-download
 icon-download-option	 icon-dot	 icon-donut	 icon-documents
 icon-document	 icon-dislike	 icon-disconnect	 icon-desktop
 icon-delivery	 icon-delete-circle	 icon-date	 icon-dashboard
 icon-cube-solid	 icon-cube-outline	 icon-copyright	 icon-contract
 icon-contacts	 icon-connections	 icon-connection	 icon-code
 icon-cloud	 icon-cloud-upload	 icon-cloud-download	 icon-close
 icon-clipboard	 icon-clip	 icon-child	 icon-checkout
 icon-chat	 icon-chat-session	 icon-chart-pie	 icon-chart-line
 icon-chart-bar	 icon-catalog	 icon-cart	 icon-carousel
 icon-calendar	 icon-calculation	 icon-business	 icon-bullseye
 icon-bug	 icon-bookmark	 icon-blueprint	 icon-barcode
 icon-barcode-option	 icon-attachment	 icon-asterik	 icon-assistance
 icon-assistance-option	 icon-arrow-up	 icon-arrow-span	 icon-arrow-pan
 icon-arrow-forward	 icon-arrow-forward-option	 icon-arrow-down	 icon-arrow-back
 icon-arrow-back-option	 icon-archive	 icon-applications	 icon-alert
 icon-alert-option	 icon-address-book	 icon-add	 icon-add-document
 icon-add-circle	 icon-access	 icon-id	 icon-arrow-down-option
 icon-arrow-up-option	 icon-checkmark	 icon-home-2	 icon-stop-option
 icon-play-option	 icon-restart	 icon-cost	 icon-approvals-option
 icon-review-option	 icon-arrow-dash-forward	 icon-arrow-dash-back	 icon-arrow-line-forward
 icon-arrow-line-back	 icon-black-beard	 icon-halt	 icon-pointer
 icon-rocket-option	 icon-traffic	 icon-traffic-option	 icon-plug

Marketplace Portal Widget Customization

 icon-plug-option	 icon-magnet	 icon-organization-option	 icon-magic-wand
 icon-layout	 icon-bug-option	 icon-sort-low	 icon-sort-high
 icon-sort	 icon-brick	 icon-link-option	 icon-service-modify-fail
 icon-infinite-option	 icon-infinite	 icon-windows	 icon-apple
 icon-linux	 icon-date-option	 icon-calendar-option	 icon-checkout-complete
 icon-linux	 icon-date-option	 icon-calendar-option	 icon-checkout-complete
 icon-checkout-next	 icon-cart-option	 icon-cart-next	 icon-cart-complete
 icon-measure	 icon-measure-option	 icon-weigh	 icon-rule
 icon-rule-option	 icon-thermometer	 icon-thermometer-option	 icon-bookmark-option
 icon-peace	 icon-review	 icon-approvals	 icon-arrow-line-up
 icon-arrow-line-down	 icon-emoticon-happy-2	 icon-emoticon-smile	 icon-emoticon-angry
 icon-emoticon-poo	 icon-cpu	 icon-subscription-add	 icon-pause-option
 icon-refresh-option	 icon-saved	 icon-square	 icon-square-hp
 icon-application-off	 icon-applications-on	 icon-application-on-option	 icon-application-off-option
 icon-star-half	 icon-wallet	 icon-basket-option	 icon-basket
 icon-umbrella	 icon-radioactive	 icon-expand-option	 icon-contract-option
 icon-chat-session-2	 icon-conversation	 icon-sun	 icon-settings-option
 icon-linkedin	 icon-dropbox	 icon-internet-explorer	 icon-chrome
 icon-firefox	 icon-trash-option	 icon-organization	 icon-robot
 icon-ghost	 icon-diamond	 icon-club	 icon-sapde
 icon-archive-option	 icon-moon	 icon-phase-complete	 icon-phase-start
 icon-phase-crescent	 icon-phase-gibson	 icon-share	 icon-watch
 icon-pin	 icon-generic-user	 icon-battery-full	 icon-battery-half
 icon-battery-empty	 icon-in-progress	 icon-new	 icon-urgent
 icon-edit-field	 icon-active	 icon-active-option	 icon-inactive-option
 icon-inactive	 icon-dog	 icon-halt-option	 icon-service-offline
 icon-service-reserved	 icon-service-transitioning	 icon-service-deploy	 icon-edit-option
 icon-service-modify	 icon-service-fail	 icon-service-active	