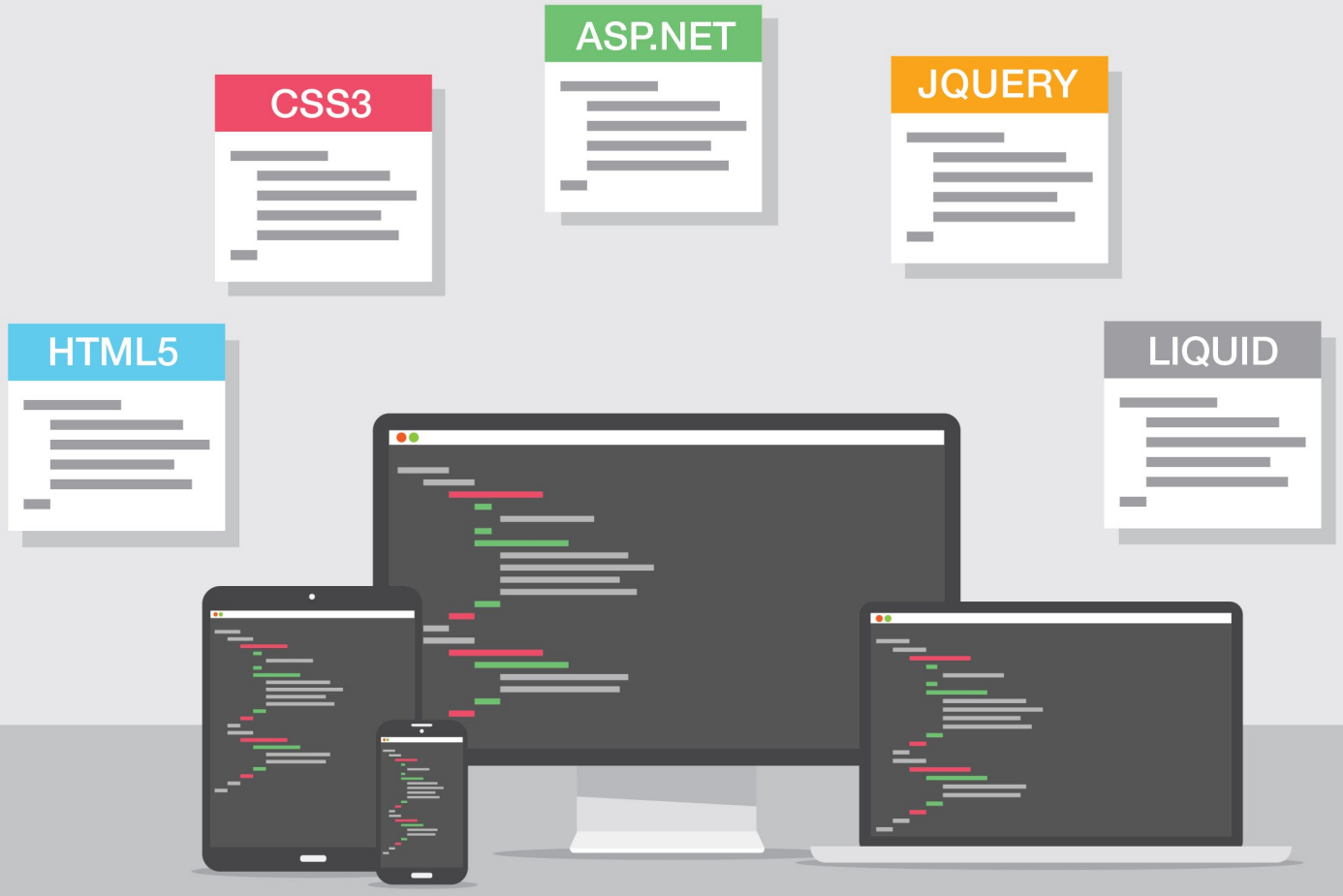
# Packaging

## Plugins & Themes



# Welcome

The power of Rock is that it isn't limited by what the core team provides. Instead the community is able to extend Rock to the furthest reaches of their creativity and abilities. Once you've made the next big thing, we make it easy for you to package it up and share it with others through the Rock Shop. This guide will show you how.

> ### We're Just Getting Started
>
> We're still in the early stages of deploying plugins using the Rock Shop. We highly recommend that you talk to us before you start development of new plugins so we can guide you through the process. This can save you time and frustration as your prepare to develop your first package.

# What Can Go Into the Store

Basically anything that you think could be helpful to someone else in the Rock community can be considered for the store. Generally though we're looking for:

- **Plugins:** New blocks and assemblies that add new functionality to Rock.
- **Themes:** Website templates that people can use to skin their external websites.
- **Workflow Actions:** New workflow actions that the community can use to extend their workflows.
- **Jobs:** New scheduled jobs that allow organizations to perform online processing.
- **External Applications:** Applications that extend Rock to the desktop.
- **Documents:** If you have a significant document that you think would be helpful for the community, you can place it in the store. This is really only necessary for documents you would want to charge for. Free documents can be shared easily through the community tools.

We highly recommend that you read through the following documents to learn more about the legal requirements of the Rock Shop. You'll be required to read and agree with them before submitting your store items so it's best to review them now before you get started.

- Sales & Refund Policy
- Developer Distribution Agreement
- Terms of Use

If you have an idea for the Rock Shop we highly encourage you to communicate it to us by emailing info@sparkdevnetwork.org before starting development. While this is not required, it can save you time by allowing us to point you in the right direction. We want to help set you up for success.

> ### Sponsored Apps
> Paid Rock Shop packages donating at a rate of 35% or more will be included in the "Sponsored Apps" section of the shop homepage.

# Package Guidelines

While our intent is to not have a lot of rules about what can go into the Rock Shop, we do reserve the right to decline a package's inclusion. Please see our Terms and Conditions for Rock Shop Vendors for details on our policies. Communicating with us before development will help improve the chance that your package will be included without delay.

In general, the evaluation process will include (but is not limited to) the points below:

- Make sure your package file meets the specs. We'll especially be looking to see that you are not putting files in core folders and that the uninstall steps you provide are accurate.
- Check to ensure that your plugin meets all of the guidelines below.
- We will do a very quick visual inspection of the code to make sure that it's in good standing. We will not be policing code heavily for quality nor will we be looking for bugs. We just want to make sure that it won't adversely impact the performance or integrity of the customer's system.
- Check to see if all of the proper information is provided for your package and that this information is of reasonable quality.
- We'll also check to ensure that the application functions and performs as advertised and documented post install.
- While we don't anticipate this being a problem, we will, of course, not post packages that are inappropriate or not germane to Rock's intended audiences.

Unlike many product app stores, we will not reject packages that *compete* with features already in Rock. We believe that choice is good and that rising tides raise all boats.

We do reserve the right to reject a package from the Rock Shop for reasons other than those listed above. Again our intent is not be rigid, but to ensure the quality and reliability of the packages that are in the Rock Shop.

# Anatomy of a Package

The unit of work that you share is called a package. This could be a set of custom blocks, a website theme, custom workflow actions or even an external application. Each package has versions so you can add additional features and fix any bugs (should they arise).

## Package Versions

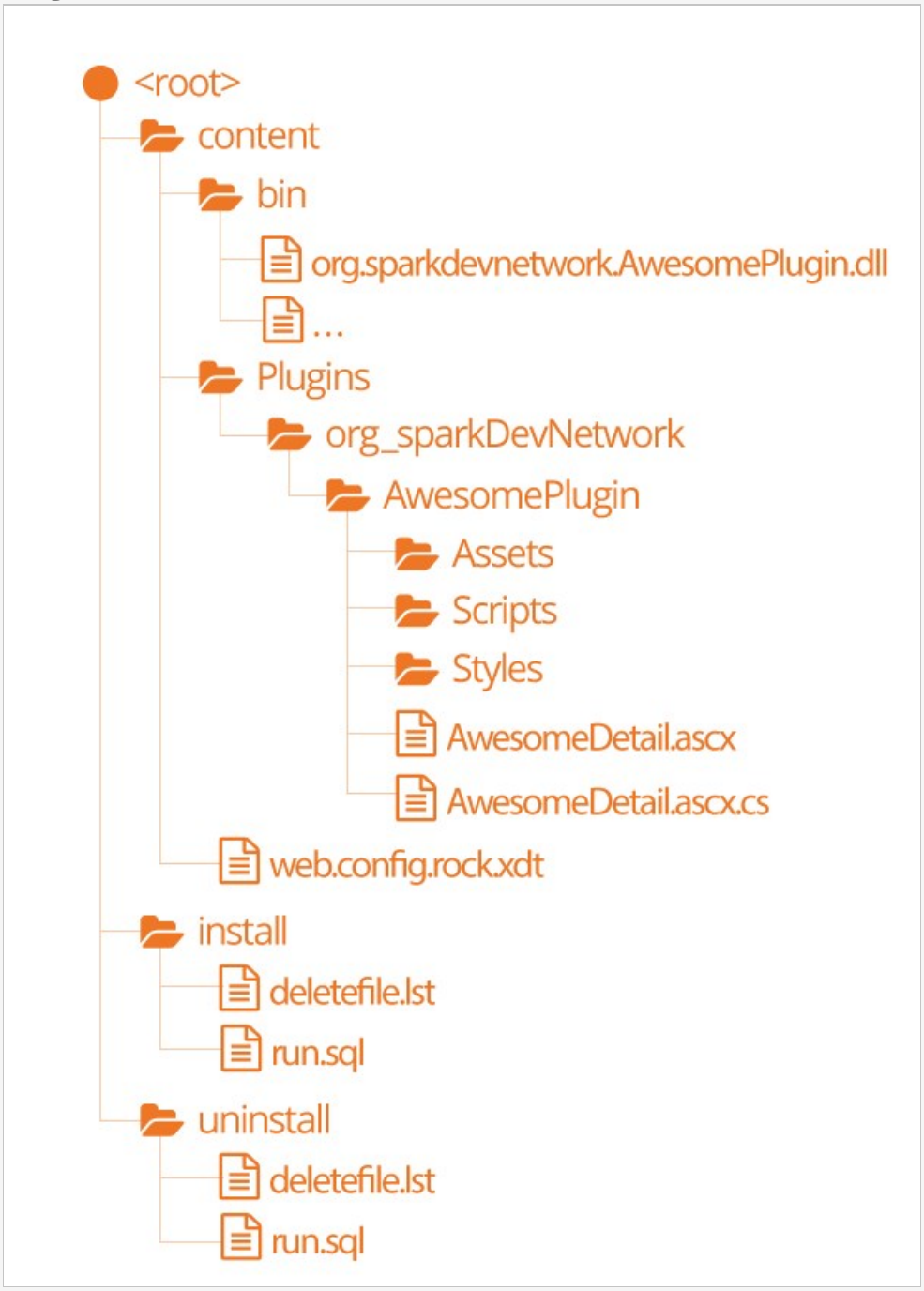
Each package version has an install file associated with it. Upon purchase (a term used even when installing a free package), the install files are downloaded to the client's server and installed. When there are multiple versions of a package, each version's install file is downloaded and installed in order. When releasing version 2 of your package, its install file is only responsible for making the changes needed to upgrade from version 1.

> ### Package Versioning
>
> Although we've provided a place to annotate a 'version' number for your plugin, this number is just a friendly label. It doesn't control the order of, or relationships to, other versions of your plugin.
>
> If you've already published Version 1 and Version 2 of your plugin, you can't create Version 1.5 to fix a bug found in Version 1. Instead, you'd have to create Version 3 or Version 2.x to deploy your fix. Package versions are always simple and linear.

## The Package File

What makes up an install file? Well let's look inside one.

Package Contents




- **Content:** (required) Files and folders in this directory will be copied into the Rock root directory on the client's server. You may only place files in the following folders under the Rock root: bin, Plugins, App_Data, App_Code, Webhooks, Content and Themes.
- **Install:** (optional) This is where configuration files specific to the install process will go.
    - Any files listed in the `deletefile.lst` will be removed *after* the package is unzipped.
    - Any SQL in the `run.sql` is executed *after* the package is unzipped.

- **Uninstall:** (required) Yep, files for uninstalling the package.
    - While there isn't an official uninstall process, the files listed in this `deletefile.lst` are the ones you would remove if you're uninstalling the plugin.

The `deletefile.lst` files should list the files to delete relative to the application root. Example:

```
bin\com.myorganization.MyPlugin.dll
Webhooks\MyWebhook.ashx
Plugins\com_myorganization\MyPlugin\MyBlock.ascx
Plugins\com_myorganization\MyPlugin\MyBlock.ascx.cs
```

## Creating The Package File

When you've assembled all your files in the directory structure above simply zip it up and rename the zip file with the extension '.plugin'. The the top-level directories should be "content, install, uninstall". The rest of the filename is up to you but you may consider adding the package name and version for your future reference.

# Database Migrations

One of the critical steps in the install process is making any needed database changes required by your package. We provide you with two options for making these changes.

1. If your package includes a custom assembly, we highly recommend that your assembly include both up and down migrations. These migrations will be run for both the install (up) and uninstall (down).
2. If your package does not require a custom assembly, you can include a SQL file in your install package. For installs, this file should be in the root *install* folder and should be named *run.sql*. Uninstall SQL should use the same file name, but be in the root *uninstall* folder.

# Uninstalling

The current version of the Rock Shop, released in McKinley v3.0, does not support uninstalling packages. We do see adding this support in the distant future. Therefore, the packages you write today must support the *plumbing* needed for uninstalling themselves. You are also encouraged to create a mechanism in your plugin to remove your plugin should the need arise.

In the *Database Migration* chapter we've already touched on how to support uninstalling your packages database requirements. Let's now discuss how to deal with the files on the server. During the uninstall, Rock will look for a file named *deletefile.lst* in the root *uninstall* directory. It will go through this file and remove all of the files/directories that are listed.

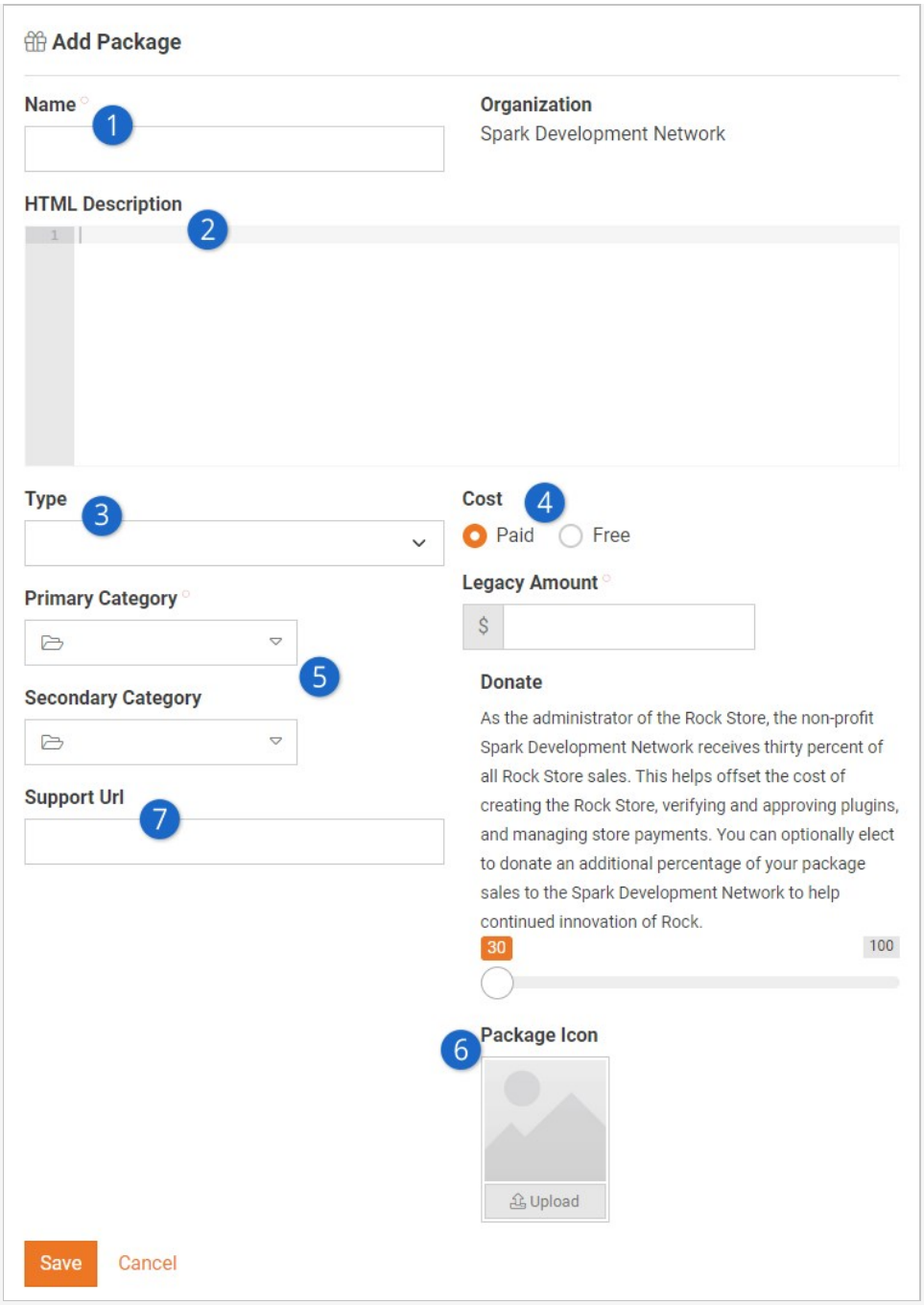
> **Be Careful Choosing Files to Delete:**
>
> While it's up to you to determine which files to delete we will pay attention to any files you request to delete out of the *bin* directory. It's possible that two different packages may use the same third party extension. Deleting these assemblies haphazardly may keep other packages from working. When you request that a file in the *bin* directory be deleted, we'll check that no other package has included that file in their install. If so, we'll refrain from deleting it.

# Submitting to the Rock Shop

Ok, your package is done and the only thing still preventing you from saving the world is getting it into the Rock Shop. Not to worry - submitting your plugin is a piece of cake! Just follow these simple steps.

1. Log in to the Rock Community website.
2. Select *Your Profile* from the user dropdown in the upper right of the page. This will take you to your user profile page on the Rock site.
3. Next select the organization that will be submitting the application. Most people will only belong to a single organization, but you could belong to several. This is helpful for those who operate as freelance ninjas at night.
4. To submit or edit your package, select the `Rock Shop Packages` link in the menu in the left sidebar.
5. You'll now see a listing of all the packages you have submitted to the Rock Shop. To add a new one press the `[+]` button in the grid footer.
6. This will allow you to submit a new package with the following fields.

Adding A Package



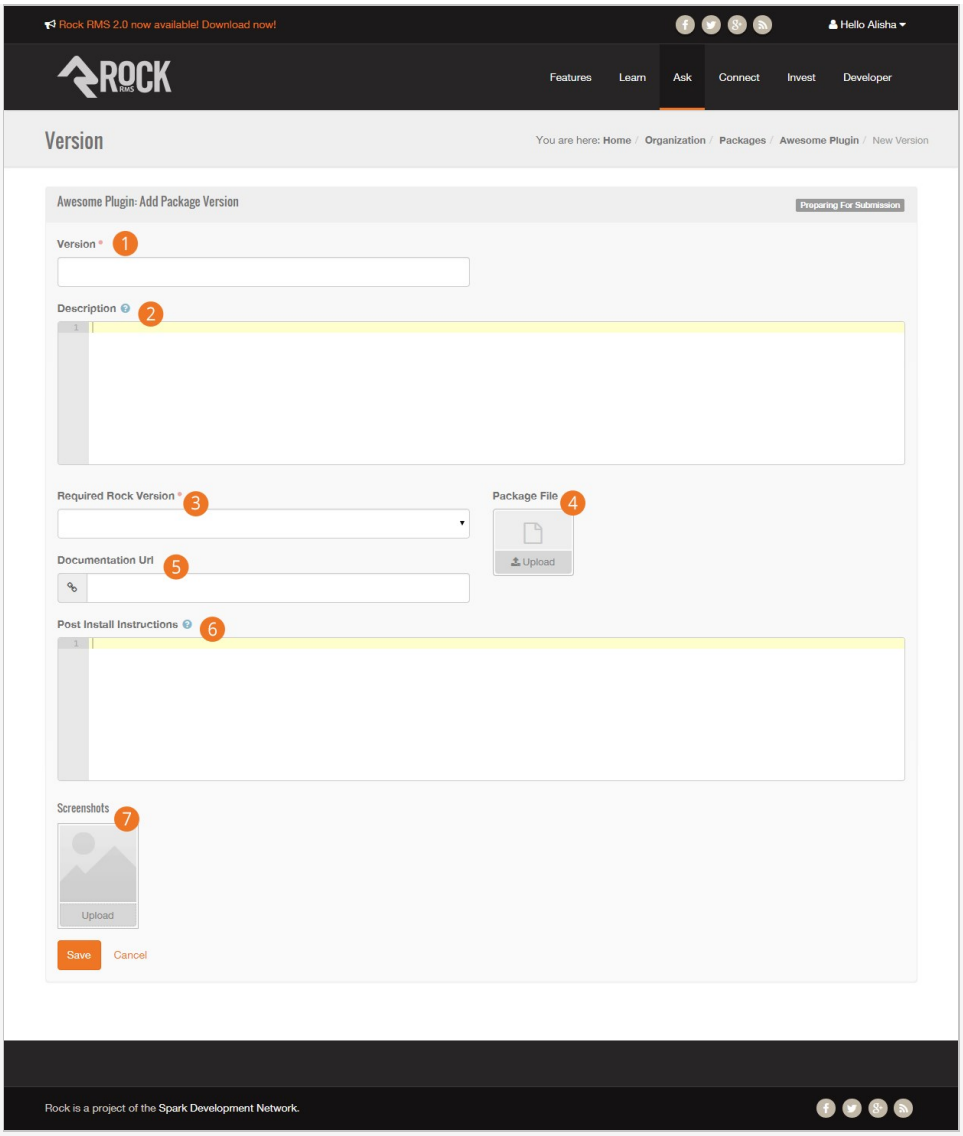

1 **Name:**
This will be the name of your package that will be displayed in the Rock Shop.

2 **HTML Description:**
Don't skimp here. Take your time and explain what your package does and how it can help the organization. This is a good place to list any limitations also. Remember, even if this is a free package the *customer* is investing time looking at your package and installing it. Respect that by providing as many details as you can. Submissions that do not contain enough details will be rejected. Be sure to leverage HTML to present your description in a way that's easy to read.

3. **Type:**
   There are a couple of different types of plugins you can submit. These include:
   - **Plugin:** This is the most common. These include things like custom blocks, field types, workflow actions, etc.
   - **Themes:** This option is for custom website themes you have developed.
   - **External Apps:** Choose this option for applications that run outside of the Rock website. Most likely this will be used for client applications that run on the desktop or mobile devices.
4. **Cost:**
   This field determines if your package is free or paid and, if applicable, what the cost is. Paid Rock Shop packages donating at a rate of 35% or more will be included in the "Sponsored Apps" section of the shop homepage.
5. **Categories:**
   Next select up to two (one is fine too) categories that make sense for your package. We reserve the right to change these especially if/when we add new categories as the Rock Shop grows.
6. **Package Icon:**
   This is an image that will be used in the Rock Shop. More details on the specifications of this image can be found below.
7. **Support URL:**
   We require every package to have a support URL where customers can go to get help with your package.

7. After creating the package, you must now create an initial version for it.

Adding A Package Version




1 **Version:**
This is the label you'll use to describe your version. We give you the flexibility to use any versioning scheme you prefer. Most people will use something like *Version 1*. See Package Versions for more details.

2 **Description:**
This field describes the specifics of the version. This is especially important for describing what's new in later versions. When filling this out for the initial version you may wish to include information about future features.

3 **Required Rock Version:**
Your package may not run on all versions of Rock. To limit who can install it, you will need to specify a minimum Rock version.

4 **Package File:**
This is where you'll upload the install package file that you created above.

5 **Documentation URL:**
This is where you'll link to the documentation about your package.

6. **Post Install Instructions:**
   Once the customer has completed the install these instructions will be shown to point them toward what to do next.

7. **Screenshots:**
   Nothing helps the customer understand your package as quickly as screenshots. You can upload as many as four different screenshots. Please upload at least one screenshot of your application. We suggest these screenshots be sized to 1280x960. Submissions without any screenshots will be rejected.

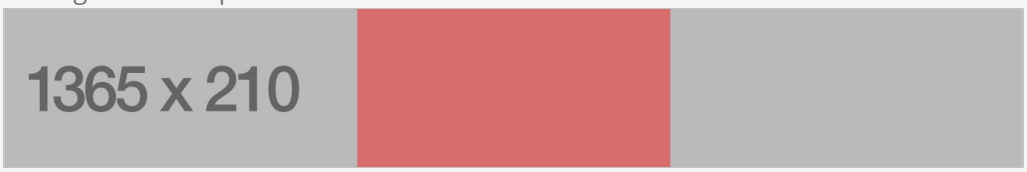
### Comping the Cost?

If you're developing a plugin for a customer who is helping to fund the development costs, please note that it is not possible to comp them the cost of the plugin purchase in the Rock Shop. To keep it simple, we recommend that you reduce your development fee by the cost of the plugin, then direct the customer to purchase the plugin through the shop.

## Package Icon Specifications

The package icon you submit must meet specific requirements to be included in the store. This single image will be used in several different places so it's important that you understand the requirements.

Package Icon Template




The image itself must be sized to 1365x210. The entire image will be displayed on the package details page of the store. The area shaded in red will be displayed when it's shown as an icon in a list of packages. Note, that this red section is not the exact size of the icon. Since the page is responsive the icon will grow and shrink in width depending on the browser width. The red area is more the target location for your icon.

# Review Process

Once you submit your package, it will go through a review process before it's available in the store. This review will ensure that your package meets the guidelines above and those in the Terms and Conditions. We'll do an install of your package and complete a quick inspection to verify that it does what it's advertised to do. We'll also do a quick review of the code to make sure we don’t see anything that could significantly impact the performance or reliability of Rock.

In general we hope to complete this review process within three weeks. This may take longer depending on the number of application submissions in the queue.

# Rock Store Recipes

While the packaging instructions above should cover most types of packages, we'd like to address some special cases in this section.

- **External/Desktop Applications:** If you would like to add an external application to the store, simply upload the setup file for your application in place of the package file. We will put this setup file on our file storage account and create a quick Rock Shop install package that will add a link to your application to the list found under `Admin Tools > Power Tools > External Applications`.
- **Documents:** If you are submitting a document that you would like to share in the store, submit the document as the package file. Like the desktop application above, we will link to your document from the External Applications list.