

# A Beginner's Guide to Marketing Your Application

*By Rick Sands and John Biondo*

## Introduction

In a nutshell, marketing your software is loosely defined as the whole process of packaging your software for distribution, getting it distributed, and perhaps being paid for doing so. I also feel that once you have distributed your software, supporting the users who have bought is also important to the marketing cycle.

There are many books about marketing, and specifically about marketing software. This tech doc will attempt to give you the basics of sharing your great application with others.

## Freeware vs. Shareware

One of the first things you need to decide is if you are going to give away your application or sell it. There are advantages to both. In all likelihood we are not software houses and are creating software for the fun of it. Of course, it does not hurt to make a little money from it either. Nothing is wrong with that.

Why would anyone give away software? Distributing your software as Freeware has some advantages. First, it can give you a chance to "test the waters" and get a feel for the process of distributing software. Freeware can be less demanding for you in terms of documentation, user support, and general stress - after all - it's free! But remember, just because it's free and, even unsupported, doesn't mean that your users will feel the same way. Being freeware gives you a valid out, but that doesn't necessarily make the user feel satisfied.

So, if you want to try and sell your software, there is a tried and true method for amateur programmers and that's Shareware. Shareware is a technique developed during the 1980's that has proved to be very effective for both the programmer and the user. Essentially, you make your software available for the user to download and give it test to see if it's suitable and meets their expectations. If the user likes the software, then they buy it, and if not, then they discard it.

There are many different models for shareware. Some are fully functional and rely on the user to buy, or register it. I call this the "Hope and Pray" distribution method. Most shareware, though, has some kind of software limitation that is lifted when the user buys it.

Unlike freeware, shareware is more demanding to distribute and sell. You should have documentation(!), warranties, and an End User License Agreements (EULA). You need to carefully place your software with sites that will sell it. You need to keep records about sales you've made and how much it cost you to make it for tax purposes.

You will find that after you have sold software, that users expect a certain amount of assistance. One of the biggest mistakes that shareware programmers make is not considering the cost of support in the price they set.

However, with all that, programming and selling shareware can still be extremely rewarding.

## **Packaging**

Before you can successfully sell or distribute your application online, you need to prepare it for distribution. Generally, this means taking your application and all related files, bundling them together and making them available for download somewhere.

Even if you are going to give your application away as freeware, you should treat it as if it was going to be sold. Many freeware packages end up becoming shareware at some point, and having a professional package for the user from the start makes that process more likely to happen.

## ***Physical Package Contents***

What should you distribute? You should include the items in the following list as appropriate to your app:

- Your hand-held application (for the hand-held).
- Libraries the application requires (for the hand-held).
- Databases required for your application (for the hand-held).
- Documentation.
- A ReadMe.txt.
- An End User License Agreement (EULA).
- A Warranty.

The files should be bundled together in a compressed file. For Windows™ users, this is generally a ZIP file. A zip file is typically created with WinZip. You can just place all your files in a WinZip file, or if you make a setup program, you can place the setup.exe into the zip file.

For Mac users, a SIT file is considered a "friendly" format. SIT files are created with StuffIt. Both Windows and Macs can use StuffIt to create SIT files for their Mac customers.

In the Unix and Linux worlds, there are a couple of compressed file formats, most notably TAR and Z files.

Palm OS® sites that distribute your software will generally give you the opportunity to distribute in ZIP, SIT, or PRC formats. If you package in ZIP format, your software will still be distributed just fine as all operation system platforms seem to handle it fine.

Avoid distributing the application PRC file directly. The plain PRC file is usually distributed this way to support users who are downloading via their hand-held. The problem for you is this: They download the application but they do not get the documentation or other materials you're including in your package. Also, a ZIP file contains information about itself so if the ZIP gets corrupted during a download, it will detect it. If you feel you must give this ability, then use one of the many onboard application installers to make a hand-held setup program.

### **To Setup or Not**

If you have many files to place on the hand-held, then you might want to consider using a Windows installer or a hand-held installer to install the files. Some users may not know which files belong on the hand-held and which belong on the desktop.

A Windows installer runs on Windows and typically makes the hand-held files ready for installation on the next sync operation. The drawback to this method is that your Mac or Unix users won't be able to use this so you'll have to provide an alternative.

A hand-held installer is similar in function - it installs the files onto the hand-held. The difference is that all the files are bundled, much like a ZIP file, into a Palm OS compatible PRC file that can simply be install as a normal application. Once on the hand-held, it expands and extracts your application and auxiliary files as normal. With this technique, any user on any platform can use it.

In all cases, the most important thing is that the user not get confused about how to install your application. As it is rumored that most users don't read documentation you have supplied until there is a problem, many developers include a README.TXT file which is a simple text file that contains any startup notes you absolutely want the user to see. In many cases a readme.txt file is displayed automatically at the end of an install from a setup program. ZIP files can also automatically display a readme.txt by using the ZIP Comments ability.

Even if you create a setup for your application, be sure to ZIP (or SIT) it. One benefit of a ZIP file is that it can detect if it's been corrupted during a download. If corruption happens it notifies the user instead of just crashing their hand-held.

## **Documentation**

Every application, no matter how small, freeware or shareware, should have some kind of documentation with it. It should be appropriate for the application and the price of the application. The last thing a user wants is a complicated program that has no instructions. And what you may think is a simple program, another may consider difficult.

Documentation may be written in a simple text editor such as NotePad for Windows™. I prefer to write my readme file with NotePad and use an HTML editor to write the

detailed documentation. HTML is a good choice since most everyone on all platforms has a browser to read it. Some developers have also use PDF files from Adobe.

The first file I create is a readme.txt file. This file is always a simple text file and needs to contain, at minimum, the following information:

**The Application Title:** Yes, but I've seen readme.txt files without it!

**The Version Number:** Always make every release you publish have a unique version number. Each should be greater than the last. They typically are a major release followed by a decimal point, and followed by another number. For example 2.05 means it's version 2 with the 5th minor revision. It is found in your application as the 'tver' resource and can be edited with RsrcEdit.

**The Release Date:** I used to rely on the version number, but found that many users relate more to a release date.

**Installation instructions:** Say something. It could be "install onto your hand-held and run "myapp" or it can be more detailed if needed.

Now at this point, you can certainly include the rest of your applications operating instructions in this file. Personally, I like to put the operating instructions into an HTML file which gives me the options of including pictures of the application screens. Of course, if you do include pictures, you'll need to distribute the images.

### ***Documentation Do's, Don'ts, and Other Tips***

Writing good documentation is an art form that takes practice and gets better over time. The goal of documentation is to give or teach a user about using your application. If the documentation is poorly written, the user will get bored and not read it. When writing documentation (or any technical document), keep these things in mind:

**Know Your Readers:** Do you have to teach basic concepts to people unfamiliar with your product? Are you writing for your peers, or people who may already understand your program but need specifics? Do you need to provide both styles?

**Write Casually:** Write as if you are teaching your friend or mother or co-worker. This kind of style helps make the material light and not boring. If applicable, sprinkle personal experience into your writing.

**Keep Topics Granular:** A user will find it much easier to locate topics of interest if the topics are small and clearly labeled. A table of contents or an index helps with this process.

**Never Assume Anything Is Easy:** Writing that something is simple or trivial assumes the reader is fully versed in your application. If the user reads something like "...the popup is simply linked to the database by filling in the database creator ID and Type

ID..." this is assuming they know the concept of a) linked databases, b) what a popup is, and c) what creator and type IDs are. It's easy for a user to get frustrated by this type of statement. Remember, things are never easy when you're learning new material. Save the simple and trivial for the marketing material, not the documentation.

So what topics do you include? Pretty much whatever you want, but here are some that I use frequently in mine:

**Title:** Clearly identify the Version Number, any Copyrights, and any other identifying marks so the user can see at a glance what they are reading.

**Installation:** You can put a list of files that get installed, inst

**Overview:** Briefly describe your program and program features. This should make them want to read the details in the following sections.

**Your Topics:** These are the various topics that describe your program functionality. These can include things like how to use your program and technical details about your application. These topics should make up the bulk of your documentation.

**Purchase Information:** If this is shareware package then you want to include the particulars about what is available in the unregistered version, the price(s) of the software, and how/where to purchase the software.

**Support Information:** This section should give details on how to get support. This often includes email, web site addresses, and sometimes phone numbers.

**Version History:** It is common for software to have a history of each release with it's version number, when it was released, and what changes were made. Users can see that you're keeping your current.

**Warranty:** You want to let the users know what is expected of your software and you as a developer.

**End User License Agreement:** A notice of ownership, license, and liability.

**External Copyright Notices:** Palm Inc., Microsoft, and other corporations require appropriate trademark and copyright notices when you mention their products.

Remember, if graphics are used make sure to limit them to GIF or JPG - no bitmaps. Ensure that the graphic files are in the same folder as the documentation.

## **Sample End User License Agreement (EULA)**

*Contributed by John Biondo*

All software that is sold should have a End User License Agreement, or EULA, that details the "fine print" of selling your package. Mostly, this is to protect you and to let your customers know what rights they have should your program go nuts and delete all their important, yet not backed-up, files.

There are many types and styles of EULAs from the extremely complex to very simple. You can look at the software you download to see many different examples. You will see that they mostly address topics such as copying the software, rights to damages, hacking, etc. They are completely open to what you put in them. The goal is to protect yourself, the user, and the software.

The following license agreement is a sample with explanations. ***NOTE: Neither PDAT Nuts & Bolts, Biondnet Software, or myself are not lawyers! This document is just for illustration purposes only!! Get real legal advice from a licensed lawyer! /NOTE***

#### BIONDNET SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold. This also applies to demo versions. (This allows the developer or company who wrote the product to "own" the product always.)

GRANT OF LICENSE. This LICENSE AGREEMENT grants you the following rights:

Applications Software.

You may install and use one copy of the SOFTWARE PRODUCT, or any prior version for the same operating system, on a single computer. The primary user of the computer on which the SOFTWARE PRODUCT is installed may make a second copy for backup purposes only. (This sets the definition for licenses on single use.)

Storage/Network Use.

You may also store or install a copy of the SOFTWARE PRODUCT on a storage device, such as a network server, used only to install or run the SOFTWARE PRODUCT on your other computers over an internal network; however, you must acquire and dedicate a license for each separate computer on which the SOFTWARE PRODUCT is installed or run from the storage device. A license for the SOFTWARE PRODUCT may not be shared or used concurrently on different computers.

(This prevents customers from installing a piece of software on a server and running it on multiple computers using only one purchased, licensed copy)

OEM License.

If you obtained this license agreement together with a hardware product, you are allowed to use this SOFTWARE PRODUCT as a part of the Hard- and Software package according to this license agreement. (For programs which are specific to certain hardware you may have sold with the software - and the software will not work without it)

SITE License

If you obtained this license agreement as a site license, you may install an unlimited amount of installations throughout your company, provided the locations have a single street address. You may install the product as a "Network" product and run the software from any other networked computer on your LAN, provided those computers are located at the same street address. (This is prevention from a customer using a site license at other company locations, which may be in other cities, counties, states, countries, etc.)

DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS

Demo Version.

Even if the SOFTWARE PRODUCT is a demonstration version, it is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. (Protects your demo program)

Limitations on Reverse Engineering, Decompilation, and Disassembly. You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. This also applies to Demo versions. (So another programmer doesn't decompile and recompile as his/her own)

Separation of Components.

The SOFTWARE PRODUCT is licensed as a single product. Its component parts may not be separated. This also applies to Demo versions. The only exception would be for Site License purchase. (Again, if this was a package, it can only be used with that package)

Freeware.

If you obtained the SOFTWARE PRODUCT as freeware, then you may freely use the SOFTWARE PROGRAM, and distribute it with all it's original components. (Keeps your "readme" and original files distributed, so others who obtain it may purchase other software you have)

Rental.

You may not rent, lease, or lend or sell the SOFTWARE PRODUCT. This also applies to demo versions. (You are the one who should make the money, not someone else)

Termination.

The license to use the SOFTWARE PRODUCT ends at the 31st of December, 2050. (This is similar to a patent license, it expires and then the software becomes freeware) - This is really not a big deal, because by 2050, we will probably be on Palm version ? or Windows ? - and this product would probably not work anyway.

Without prejudice to any other rights, Biondnet Software, John Biondo may terminate this LICENSE AGREEMENT if you fail to comply with the terms and conditions of this LICENSE AGREEMENT. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts. If the software contained with this agreement was purchased, the money will not be refunded due to a violation of this agreement. (This allows you or your company to cancel this license agreement with a customer who violates the agreement, without a refund of the money your customer spent)

UPGRADES.

If the SOFTWARE PRODUCT is labeled as an upgrade, you must be properly licensed to use a product identified by Biondnet Software, John Biondo as being eligible for the

upgrade in order to use the SOFTWARE PRODUCT. (This prevents someone from using an upgrade to obtain the software without purchasing the original)

A SOFTWARE PRODUCT labeled as an upgrade replaces and/or supplements the product that formed the basis for your eligibility for the upgrade. You may use the resulting upgraded product only in accordance with the terms of this LICENSE AGREEMENT. If the SOFTWARE PRODUCT is an upgrade of a component of a package of software programs that you licensed as a single product, the SOFTWARE PRODUCT may be used and transferred only as part of that single product package and may not be separated for use on more than one computer. (Same as above)

Biondnet Software: <http://www.biondnet.net>

## **An Online Business Model**

*Contributed by John Biondo*

This document's purpose is not to teach you how to make, edit or use web pages, so that will not be discussed here.

First, you have to create your business. The first thing to do is to establish an internet site or host for your pages/files. There are hundreds of internet sites who offer free pages, but beware of their advertising schemes and pop-ups, which your customers or visitors may get tired of seeing or dealing with. Also, you may want to register a domain name with a registration service. This will ensure your website has a typical www name. This allows customers and visitors an easy way to remember your site's name. Once a registered name is obtained, the host can "point" your www name to your website's actual URL address.

Once your website is set up and running, you must "advertise" it's existence on the web. There are tons of ways to do this. One is to actually pay another website, search engine, or company to place ads on the web for you. This can be quite expensive.

Another way to market your site is to make use of the web's architecture and search "bots" to find your website through the use of "Meta Tags". Meta tags are what search engines and other searching tools use to "look up" series of titles, names, addresses, and more. These Meta tags are placed on the website's first page, or "index" page. This way when a search engine sends out it's bots or crawlers, it "sees" them and loads the name of the page, and some four or so lines from the page into it's search results. (Do a search for meta tags on the web to find instructions how to create them.) Another way is to use banners on your site and by placing your banner on others.

OK, now we have a website, setup the meta tags, advertised, now what? Sit back and wait?

No, certainly NOT.

Using other websites which allow you to sign up as a developer (Handango, Padassi, Palm soft, PDAGreen, etc..) will also increase your chances of stumbling into buyers of software. It is also a place where people can have a link to software on your website. Here are a few things worth considering when using these types of sites.

1. Check the commission of the website (i.e. what % do they keep from a sale of your software, do they charge a fee for transactions involving credit cards, etc..)
2. Actually READ the development agreement, don't just agree and go for it. This is where you may find "hidden" things that you are agreeing to, such as % increases, authoring information, etc.
3. Check out the sites download counts. Is anyone actually downloading software from the site? Is it popular?
4. Check out what kind of software is listed, does yours "fill a void"?

There are most certainly other considerations, but these are the basics.

## Monitoring Sales and Generating New Business

*Contributed by John Biondo*

In order to have success as a company, you must be able to monitor your sales and generate more business. Let's face it, not every piece of software you write is going to sell. Just because you think it's a good idea, does not mean everyone else will too. While in operation, continually check the download rates of your software, I do this at least once a week. I keep a spread sheet of all my software, on all the sites they are located, and the downloads of each one. This tells me many things; how well a particular site is "known" on the web, how many people visit this site, how many people are interested in what types of software, etc. You can use this information to formulate how you want to present your applications, or which applications to place on a particular site. Each sale, should be considered as you would a customer walking into a store and buying something. Even though we are talking about the web, and a particular sale, each should be treated exactly the same. I always thank my customer via email and offer support if they need it. (This is usually included in an email I send them with their unlock code for the software). See example below:

Thanks for purchasing MilesGear. The latest version can be obtained at: <a href="http://biondnet.net">http://biondnet.net</a> Based on the Hot Sync Name of: Bob Smith
---

Your unlock code is: MG-53204691

If you have the time, please visit <http://handango.com> and post a review for this app.

To register this app on your PDA, run the app, select help from the menu, then "Register". Type in the registration code EXACTLY as it appears above.

If you need technical support, please email: [support@biondnet.net](mailto:support@biondnet.net)

Thank you,

John Biondo

Biondnet Software

The above example is short, sweet and to the point. This is done intentionally. People want the unlock code, so they can register immediately. They don't want to read a whole bunch of nonsense. I keep my messages simple and to the point to make it easier for the person to "get what they need" and nothing more.

Once the sale is made, I usually do not forget them. Every time I release a new version of a product, I'll send a quick and easy email to let previous purchasers know that a new version has been released. I also never charge for the newer release. (People LOVE free stuff). This makes them "talk" about your software to others. - Word of mouth has always been a good sales tool.

Speaking of free stuff, I usually release a free piece of software every quarter of the year. I will post it on every site I am registered at, as well as on my own website. This is easy and free advertising for your software. People will download it, use it, then "wonder" what else you may have for free. On my free software, I usually put in my web address on one of the forms, where it is plainly visible. This makes them check out my website.

## Supporting Your Users

Support is one of the most important aspects of running an internet business. People do not like to be ignored, or put off for a long period of time.

I have always responded to a tech support issue, within 24 hours of receiving it. Most of the time, I answer within a few hours. (Fortunately I am always near a PC). If you do not have that capability, then at least check your email as often as you can.

Because I answer as soon as possible, people have written and posted good things about my web support. I, in turn, use these writings for ad purposes. (I post them on my app

download pages). This lets potential customers know before they buy, that I have excellent support ratings.

Example support issue I actually had:

When I first released MilesGear, I was using a method of calculating with goto form, such and such, then goto form such and such, this cause an unrealized (At the time) issue with Handspring devices. They would lock-up and cause a fatal alert. Requiring a soft reset. This was devastating to me, when I first got wind of it. How could I have released this application with an issue? The answer was I just didn't know.

I received three or four emails regarding this within a few days of it's release, what a fearful state I was in! I immediately recompiled a manual version and offered it to Handspring Owners. This version didn't do the automatic calcs, but it allowed the app to work like the original, and they were happy with the new version. I also went right to all the sites it was posted, and placed a warning to any Handspring owners. - The sales decreased slightly due to the many Handspring owners out there, unable to use the self-calculating version. But, the emails for support of this issue stopped.

Then, when I figured out and used the "menu" item commands, to cut and paste the fields instead of using the goto commands in MilesGear, I was able to overcome this issue.

I then returned to all the sites I have MilesGear posted on, and re-wrote the warning to read that Handspring owners who had issues and would like to re-try the app should do so, because the issue was resolved.

Sales then returned to a normal rate.

When a customer is just arrogant, and nothing seems to help, then refund their money! These type of people cannot be dealt with any other way. (You will have at least one during your business dealings.) [Editor's note: Always be polite - more than once I've had a person I was ready to give up become a customer again!]

The simple rule is: The customer is always right, and do anything in your power to resolve each and every issue. The time and effort you spend to do this, will come back to you in increased sales!

## Refund Policies

For all developers, the time will come when you will have an unhappy customer who requests a refund of their money. This should be done with no hesitation and with a smile on your face.

In a poll on our forum a question was posed:

“I have a customer that bought one of my apps but is unable to use it because of his o/s. I have the min o/s clearly stated in every description. ... Do I give refund or stick to min requirements as stated in description?”

The comments that were elicited were varied but unanimous for giving the refund:

... Better to keep potential future customers happy since this person might write a bad review somewhere for poor customer service, etc.

... Always good to keep customers satisfied.

... A pain in the butt but it's the best way to go.

... You might lose a few bucks and a little time, but bad publicity can never be reversed. And good publicity can't be bought.

I think that last bullet point is words to live by. Personally, I have refunded money and still had the user return to buy more of my software. The ability to get a refund is very important to a user.

However...

Not everyone agrees with that. Most software bought at stores are un-returnable after the shrink-wrap has been opened. Software sold as shareware should have been tested by the user before being bought. So if you offer no or limited, refunds, a policy should be created, stated on your website and documentation, and consistently adhered to.

Here are some common considerations:

**No Refunds.** Often used when a "full" version is sent to a user after they have used the trial version. Used on high-profile software like games that attract younger (and poorer) audiences.

**No Refunds After 30 Days:** This avoids the users who wants a refund three months after they purchased it. Too bad, too sad.

**No Refunds Due To Device or OS Incompatibilities:** If the user bought Palm OS® software for their Pocket PC™ device or ancient Palm Pilot, then too bad, too sad.

## ***Refund Tips***

If you do decide to refund, here are a few tips for it:

PayPal allows you to refund within a short amount of time without penalty.

Don't refund money you have not collected. For instance, PalmGear and Handango will refund the user's money upon request without penalty.

Avoid charge backs to your money-handlers. This can result in extra fees for you.

Don't refund cash. If you can't do it electronically, write a check and send it to them. Clearly mark the check with "REFUND for APPNAME" so you will have the cancelled check as a receipt.

## **Taxes and Expenses**

Well, as I'm not a tax expert, I'm going to defer writing anything in this space.

I can write one general rule: *Always pay your state and federal taxes.*