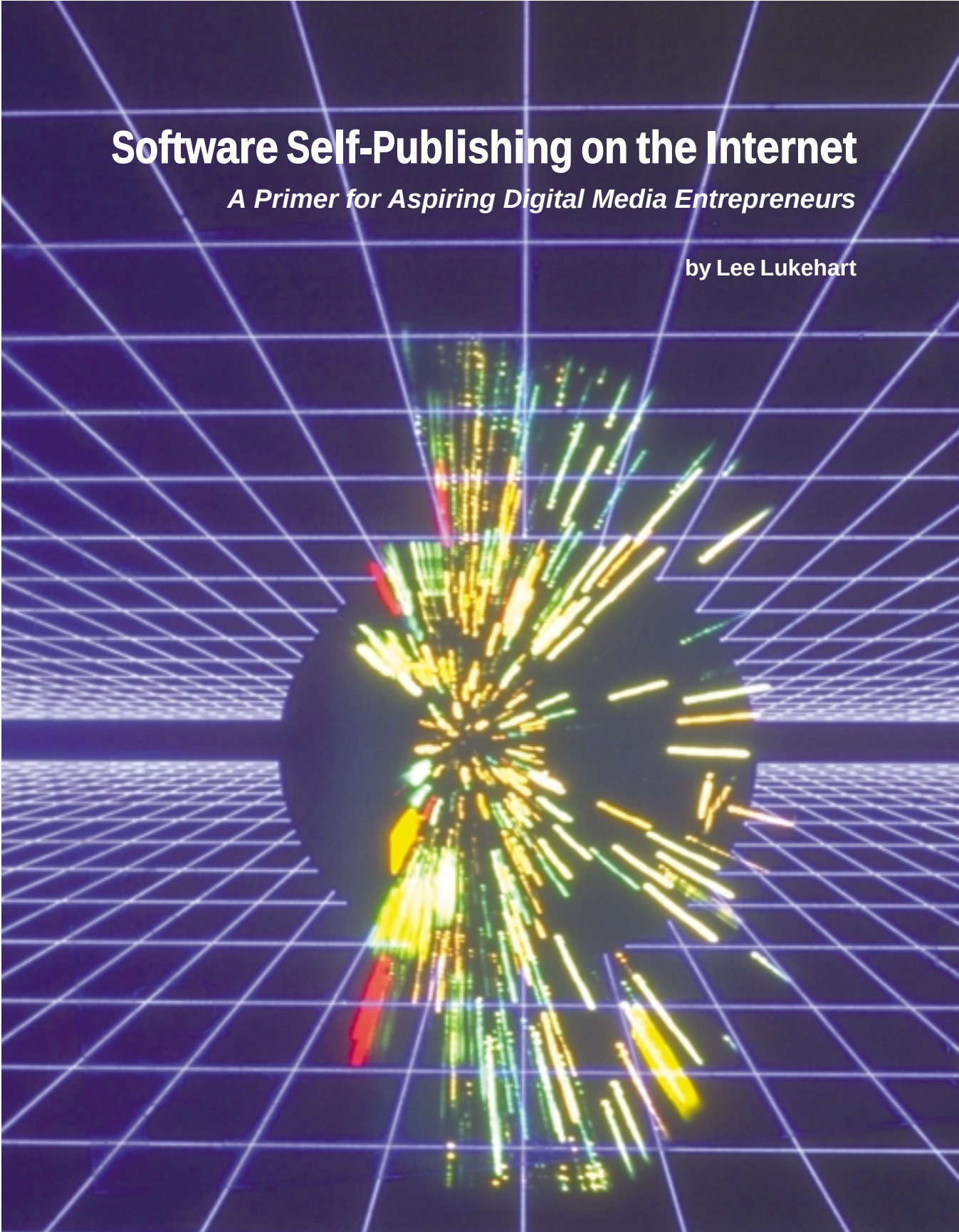
# Software Self-Publishing on the Internet

*A Primer for Aspiring Digital Media Entrepreneurs*

by Lee Lukehart



Published by QuestMedia, an imprint of Lifescope Inc.
Lifescope Inc. ❁ PO Box 10208 ❁ San Jose, CA 95157
e thefolks@lifescope.com

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1
ISBN 1-58847-050-4

# Table of Contents

# Preface

## The Intended Reader

Greetings successful software entrepreneur! The mere fact that you are reading these words confirms that, in spirit anyway, this salutation is appropriate. Whether reality has caught up to your dreams is a mere detail—which will be addressed over the course of time. By picking up this self-publishing primer you have proven that you possess at least one required trait of a successful entrepreneur: an enterprising mindset.

You have already produced, or have an idea for, a piece of software which you wish to distribute, extensibly for profit. Yours may be a software program in the traditional sense—an "executable"—or it may be some other form of intellectual property in digital form, collectively referred to as content. Examples of the latter are graphics, photographs, sounds, animation, video, music, fonts, templates, or textual content.

You strongly suspect your program or content would be valued by others, who would gladly pay a few dollars for its use, if only they knew about it and could easily obtain it. This is the subject of this primer, and events have transpired that make achieving this end easier than you may have imagined.

## This Historical Era

We live in a time of unprecedented opportunity. Three trends have converged which make *now* the best time in history to be a small software or digital media developer.

### ① Knowledge Work is the Status Quo

Knowledge work is typically described as where most of the "work" takes place in the head, rather than with the body—it is information-based versus materials-based.

Business management guru Peter Drucker prophesied in the 1970s that the rise of the knowledge worker would bring empire-toppling shifts in autonomy and independence. The balance tilted in 1956, when white collar jobs in the U.S. first outnumbered blue collar jobs. The momentum continued, to the present time

where knowledge worker positions outnumber manual jobs two to one.[1] In some sectors, like the computer industry, that ratio is closer to 4:1. With the transformation of what work is done, came a shift in *how* work is done.

The first significant difference is where the power resides. In stereotypical materials-based work, power is wielded by the manager, who controls the worker to induce a preconfigured behavior to obtain the desired tangible results. In knowledge work, however, the taskmaster has lost his whip. Now, the locus of control lies with the worker—who has become the contributing agent, not the compliant instrument. Accompanying this increase in influence over work's outcome, is more autonomy over how the work is done. The worker has, of necessity, become more self-directed, and thus more independent. Of note here, is that independent thought leads to independent action—the freedom to choose and use your product.

The second difference, and the more significant change relevant to software distribution, is that information has become the fabric of work. While specialized knowledge is necessary in manual work, it is used only as a tool—not as the object of the work. In knowledge work, information IS the material. We also use information tools to work with information material. Using a word-processor to write a memo or article for delivery to a web page is like hammering a nail into wood—but the hammer and the nail and the wood are all information-based digital media.

Unlike material in the physical world, however, digital information can be replicated and distributed without diminishing the original. There is virtually zero cost attached to duplicating your product when it is delivered online, and you'll never run out of inventory. Even the user guides, manuals, and help systems can be delivered digitally with no reproduction cost. Physical shipping costs aside, you also shift the cost of printing manuals to the consumer, should they desire to see that information in paper form. As you see, software publishing has some significant advantages.

---

1. Drucker, Peter F., *Management*, San Francisco, California: Harper Business, 1973, p. 34.

### ② Ubiquity of Computing Appliances

At work and at home, the means to employ these digital tools and digital materials are broadly available. As familiar as the effects of the knowledge worker phenomenon may seem now, twenty years ago these revelations were perceived by only a fraction of the business world, and an even smaller portion of the general population. Fifteen years ago fewer than 20% of American households owned a personal computer, the requisite knowledge worker enabler. Now more than 70% of households possess at least one.[1] An estimated 160 million computers are in use in the U.S. alone—550 million worldwide.

In addition, the transformation of general-purpose computing devices into specialized information appliances continues to expand software development opportunities. The wide adoption of personal digital assistants like the Palm Pilot and Windows CE devices are but one branch of the emerging digital appliance paradigm. Devices which resided solely in the category of consumer electronics are crossing the digital divide to become consumers of software. DVD players, hybrid CD-ROMs, digital cameras (still and video), and miniature MP3 music players are examples of devices for which you can supply digital media.

We are witnessing the fourth information revolution. The first three were spawned by: the invention of writing five to six thousand years ago in Mesopotamia, the invention of paper in China circa 1300B.C.E. (which begot books), and Gutenburg's invention of movable type and the printing press, in 1450.

### ③ Universal Access via the Internet

Twenty years ago the Internet consisted of fewer than 200 "host" computers serving academia and the military. Now, it connects the world, with literally tens of millions of computers connected. The latest surveys project that by the end of the year 2000, over half of North America's citizens will have plugged into the World Wide Web at some time.[2]

These trends have come true more quickly and more completely than most people expected or even thought possible. Alvin Toffler's books Third Wave (1980) and Power Shift (1991) verified the snowballing trend that information is power.

---

1. U.S. Department of Commerce. *Falling Through The Net: Defining The Digital Divide*, <http://www.ntia.doc.gov/ntiahome/fttn99/>, Feb. 1999.
2. Computer Industry Almanac, Inc. *North America is the Leading Region for Internet Users*, <http://www.c-i-a.com/199908iu.htm>, Dec. 1999.

These three developments—knowledge work as the status quo, the ubiquitousness of personal computers, and the almost universal and practically instant connectivity of the first two—converge to give rise to what is known as the fourth information revolution.

Never before in the history of humanity
➢ has so much information
➢ been created so quickly,
➢ distributed so widely,
➢ to so many,
➢ so easily,
➢ with so little incremental cost.

## The Goal of This Document

So what does this mean to you? The power to be independent! The end goal of this document is not modest. Its objective is to smooth your path to software publishing success. There are enough obstacles in the pursuit of earning your desired income, running a profitable business, and publishing software. Combining these objectives simply multiplies the difficulties.

Herein lies a clearly marked path to deliver your proposition of value—the product resulting from your knowledge work—to the buying masses. Admittedly this is only one path—but it has been followed by many, and the path has become a road. As is the case with any well-traveled byway, numerous merchants have sprung up to serve the needs of the many sojourners. You will find many of these tool vendors listed within this guide, with instructions on their prudent and effective use.

## How This Guide is Organized

This document is laid out to match the chronology of creating a product from scratch. Imagine that we could shorten the entire product life cycle into one minute.

Here are the recommended steps you would take:

- ❏ Conceive a product idea which appeals to a broad, motivated audience. (A service idea which can be implemented in software is equally valid.)
- ❏ Determine what features your software will provide to fill the identified needs. (What programming tasks should you expect? For what computing platforms [PC, Mac, Palm] will you provide the solution?)
- ❏ Do the application programming.
- ❏ Determine how you will deliver the software to your market. (Make the marketing decisions on selling it "retail vs. try-before-you-buy" and with "physical package vs. online.")
- ❏ Develop business relationships to enable the delivery and payment process (either by doing it yourself, or with a service provider partner).
- ❏ Make decisions on how your software must be modified to accommodate your delivery and marketing methods. (Those decisions have a direct bearing on how your software is coded.)
- ❏ Do the programming modifications and make arrangements for their back-end execution. (Put the post-sale fulfillment processes in place.)
- ❏ Create the installers for your software. (Make the process as easy and error-proof as possible.)
- ❏ Test to make sure *everything* works as planned. (Especially for online delivery, quality control is imperative!)
- ❏ Complete and polish documentation. (And update the Read Me! file.)
- ❏ Make your product available. (If people can't find your product they can't buy it.)
- ❏ Publicize and promote your program. (If people don't know about your product they can't buy it.)
- ❏ Begin working on the upgrade.

Excepting the last point, this guide covers all of these items, in this sequence. The assumption is being made, however, that at least one of your delivery venues will be the shareware/trialware model. Are you not wholly convinced of the viability of distributing software in this manner? Read on.

Chapter

1

# The Case for Shareware

"Small opportunities are often the beginning of great enterprises."
—Demosthenes

## The Evolving Software Marketplace

Commercial attempts at online software distribution are not new. Business have been formed for this purpose—and gone bankrupt—for at least the last fifteen years. Like advanced technologies in science-fiction stories, the dream of on-demand, just-in-time dispensing of software commands a powerful attraction.

The dominant software distribution models of the last twenty years have evolved, but to date the majority of consumer software sales have been transacted at the retail point-of-purchase with boxed, shrink-wrapped software. Charging onto the scene ten years ago, the mass-merchant chains (e.g. CompUSA, Fry's Electronics) and consumer warehouses (e.g. Sam's Club, CostCo/Price Club) quickly took dominant percentages of market share. While still viable major outlets, those retail sellers were eclipsed by mail-order houses (e.g. PC Connection, MacZone), where software is shipped rather than carried from the store. In all of the above cases, however, the retail model is functionally identical—you buy the software, then you get the software.

With the first personal computers, came the first hobbyist computer programmers. These talented lads (almost all of them were male) developed small programs useful in certain narrowly specific situations. User groups were forming across the land, and members congregated to trade useful programs they had written. The programmers did not expect any payment, mainly because their programs were small, simple applications that could not be considered marketable and for which the authors offered no support.

In 1982, a couple of these programmers, Andrew Fluegleman and Jim Knopf (dba: Jim Button), had written major applications (a communication program and a database program, respectively) on their new IBM PCs. Not wanting to invest the time and money in trying to get these applications into stores, they decided to take advantage of the underground distribution networks. They allowed their programs to be copied, but put a request in the on-disk documentation for the user to send money to the author to finance the ongoing development and support of the programs.

To legitimize this channel, Fluegleman called his software *Free-ware* and trademarked the name. Since the software wasn't quite free, though, Button coined the phrase "shareware," which became the generally-used term to describe this method of distributing software.

## Types of "Wares"

**Commercial Software.** The most common form of commercial software is the shrink-wrapped boxed software found in stores everywhere. If you read the fine-print in the box, you'll discover that what is actually being sold is not the software, but a license to use the software. This distinction restricts your ability as a purchaser to copy it, resell or give it to others. The illegal pass-ing-on of copyrighted software is known as "piracy."

**Public Domain.** Software applications created by the govern-ment or under direct government support (by universities, colleges, research institutes, etc.) are said to "reside in the public domain." Since their development was financed with tax dollars, such software is legally available to all U.S. citizens unrestricted and free of charge. Authors may also voluntarily designate their software as public domain, but unless they do so, they are natu-rally granted the copyright protection accorded all forms of intel-lectual property.

**Freeware.** Freeware is software which can be freely used without payment to the author, but for which the author retains the copy-right. Fluegleman's trademarked term gradually lost its original definition, and through popular usage it came to mean software for which no shareware fee was asked. Although some people refer to freeware (and even shareware) as being public domain, in

reality no shareware and very little freeware is truly public domain.

**Shareware.** Shareware is, in contrast to commercial software, distributed freely to all users. The author specifically grants the right to copy and distribute the unregistered version of the software, either to all parties or to a specific group. For example, some authors allow individuals to freely copy the program, but require written permission before it is commercially distributed. Like commercial software authors, many shareware programmers are accomplished code writers, and their shareware programs are of comparable professional quality.

Shareware is therefore really more a marketing method than a type of software or distribution channel. Shareware gives users a chance to try software before buying it, allowing them to test the software for a given time period without charge. If someone tries a shareware program and decides to continue using it, they are expected to buy a user license. If they don't pay the fee, they are no longer entitled to use that software. Shareware has the ultimate money-back guarantee—you don't pay for it until you're happy with it.

**Trialware.** Trialware is a type of shareware, sometimes also referred to as "demoware." The distinction is that trialware is distributed with the program's full functionality somehow impaired until a person buys and registers it. The limitations may be in feature (e.g. lack of print or save commands), or in term of use (e.g. expires after 30 days of first use or after it has been run 20 times). The power to unlock its full capabilities is usually self-contained, activated by a registration key code. To get the full-functioning version, the user is "forced" to register to obtain a code. This single difference from traditional shareware is believed to create at least a five-times increase in user payments. Throughout this document, the terms shareware and trialware will be used interchangeably.

**Other Types.** There are still some other types of "wares," but all are variations on the types described previously.

- Careware—send a donation to the author's, or your own, charity or cause of choice.

- Postcardware—instead of getting money, the author just requests a postcard from the user.
- Emailware—author requests an email.
- Beerware—author requests a bottle of beer.

### The Clear Winner

The unique synergistic combination of three factors makes the trialware model the unstoppable future of software distribution:

**① Free Initial Access**

Remove the barrier of advance purchase, and you increase exposure to potential buyers. Since the program can be freely obtained by anyone who wants to try it, it is more likely to get widely distributed.

**② Immediate Delivery**

Online access via the Internet provides the ability to get a registration key or a full, unlocked version of a software program practically instantly—no driving to the store or waiting for the mail. Consumer demand is skyrocketing for electronic software distribution (ESD).

**③ Digital Information**

There are no physical materials involved, so materials cost is essentially zero. The opportunity to benefit from economies-of-scale at small volumes makes the profit
potential huge for small independent software publishers.

Chapter

2

# Pre-launch Business Considerations

"If man has good corn, or wood, or boards, or pigs to sell, or can make better chairs or knives, crucibles, or church organs, than anybody else, you will find a broad, hard-beaten road to his house, tho it be in the woods."
—Ralph Waldo Emerson

## Choosing a Target Audience

One of the basic truths of successful business is the importance of the target market. Everything revolves around them. If your experience is in engineering or programming, however, your inclination may be to deal with development issues rather than marketing concerns. You probably are building a product you want to use, rather then focusing on what will sell. This is not inherently bad, but for a shareware product to sell well, it must appeal to a broad audience—or capture a large share of a niche.

This market myopia is a classic—and avoidable—cause of failure in many product introductions. Overlooking market needs leads to faulty design and benefit misfires. *Why* are you adding that whiz-bang function, and will it really be appreciated by most users? Sure, some features have strategic rationale—all the competitive products provide "X" so yours must, too. Or you may distinguish your product by having an aura of esoteric or expert-level appeal. But before you move full speed ahead, consider your target audience's reception to your latest improved mousetrap.

In fact, take a step back to view the larger picture. Some types of software are better suited to the shareware/trialware model than others. A successful shareware product must be attractive to a large, accessible audience who is ready, willing, and able to buy. Ideally, you will not only know you audience's wants, you will know how to reach them. Fortunately, the latter is not quite the requirement for shareware as it is for commercial products.

One great advantage of shareware is that the audience becomes self-selecting, meaning that you broadcast your program far and

wide, and the appropriate market will discover it and gravitate to it. This is based, of course, on how attractively you promote its capabilities, which is another of the marketing mix elements.

**Think Globally, Act Globally.** There are no geographic boundaries on the Internet. English is the primary language of software, but it is by no means the only language—so don't forget about foreign markets. While most computer-savvy consumers know English, their desire to buy will be greatly enhanced if your program is available in their native tongue. Substantial profit potential lies in providing multi-lingual, localized versions of your software, as most software authors ignore or dismiss international markets. Since fewer programs exist in those market spaces, it's easier to capture a larger share of audience.

Generally accepted, too, is that non-domestic versions often command higher prices. However, be careful to not set the price discrepancy so high that it becomes perceived as a penalty. You will offend your prospective foreign customers rather than impress them with your global ways.

Localization isn't cheap, however. If your product is a software application, the translation cost can be minimized. If, on the other hand, you provide extensive textual content, you may want to find more affordable alternatives. Standard rates for dependable translation vary from 10¢ to 30¢ per word, depending upon the language and the length of text. Especially significant within the context of a computer program, be aware that the length of text may change. English-to-German translations, for example, typically require 30% more space.

There are options, if you don't wish to invest in a translator, can't find a willing multi-lingual friend to help, or can't translate your program and documentation yourself. Translation software from companies like Systran <http://www.systransoft.com> and Synapse Adaptive <http://www.synapseadaptive.com> will batch-process an entire document. Automated Web services are available to translate text snippets for free. In fact, Alta Vista's Babelfish <http://babelfish.altavista.digital.com> will actually convert an entire HTML page on-the-fly, formatting intact. Of course, a machine can turn out some strange conversion, so be sure to run it by a native-language proofer so the obvious gaffes are avoided. Some

awkwardness is not necessarily bad, as long as it doesn't impair use of the program.

The Macintosh is especially translation-friendly for programmers. You can put display text in the file resource fork of a program, where it can be easily replaced without your further involvement.

## Product

After ascertaining you understand the needs of your target audience, your attention should next be directed to the surrounding four elements of the "marketing mix" model, as taught in Marketing 101: Product, Price, Place, and Promotion.

"If you want your shareware product to sell well, it must have an audience, frequent use, robustness, commercial-class features, and an attractive price."
—Rick Holzgrafe
Shareware Author

When considering Product, think specifically about the characteristics which influence the prospective trialware customer's perception of value.

**Quality.** Whether your program sells for $10 or $100, it must have a "commercial" attention to quality. Anything less is simply shareware sales suicide. Also known as robustness, your product must be stable under all situations—no crashes, no major bugs, no misbehavior—no excuses! Testing is a vital part of your development process. Once you get the major features in place, find people to try your program and give you feedback about its usability. When you think the product is ready to ship, test some more. *(See "Beta Testing" on page 42.)*

**Extensive Usage.** The more time the user spends with the program the more likely you'll get your shareware fee. A user who perceives they're getting good mileage from their trialware considers the shareware fee a fair price.

**Frequent Usage.** The more *often* your program is used, the more likely you'll get you shareware fee. Your program should be used daily if possible; constantly is even better. Examples of items with possible constant usage would be fonts, network tools, or system control panels and extensions.

**Uniqueness.** How does your program stand distinctly apart from the crowd? Familiarity does indeed breed contempt. The fact is,

most users won't pay for your product unless they are aware of its unique value to them.

### Price

Bargain pricing is one of shareware's primary attractions—prices are noticeably lower than similarly functioning commercial products. Retail software products have material cost of goods attached to them; the CD-ROM or diskettes, printed manuals, and retail box contribute anywhere from $8 to $20 to the raw cost. Shareware eliminates all of these. You can charge proportionately less, passing along some of the savings to your customers while making the same profit.

The two variables to consider while setting your pricing are the the perceived value to the user and the competitive environment.

#### ① Perceived Value

Cost-based pricing is old-school, especially with a product like shareware that has no material costs. So, you may think, why not base pricing on your cost to develop it? A perspective to keep in mind is that as programming tools get more sophisticated, an experienced coder might duplicate your results in one-tenth the time and effort you expended, with correspondingly lower development costs.

Besides, users don't care if you are living on bread and water because you spent two years writing your finely crafted programming pièce de résistance. They care first about their needs, which is as it should be. You must deliver a value far in excess of its cost. This distinction is also an advantage; you can charge a price far in excess of your cost, as long as the value to the customer is greater than *their* cost.

#### ② Competitive Environment

Another assessor of value is the availability of options. If people want a fix to problem "Z" and your program provides the only solution, then you can name your price. Programmers tend to be computer-centric, so recognize that similar technology isn't your only competition. For example, if you had the world's only program for logging car mileage, you still could not charge much for it because a non-software alternative exists: pen and paper.

To determine the viable price range for your product, you need to know the shareware and retail markets for your product. Research the price of similar programs and compare them to your software in terms of features and quality. For retail products, look at street price, not list price.

To be perceived as high value, your product must appear to be better and/or cheaper than the alternatives. If an alternative is deeply ingrained in people's habits, then your program must be greatly superior—not just cheaper—to motivate people to switch.

Beware of underpricing. Reducing your price generally has no effect on the size of your target audience. If your solution isn't needed, selling it for next to nothing won't induce a purchase. A low-price strategy may work if it is based on competitive comparison, but it frequently is an act of desperation. Price is only one aspect of value; if the benefits are slight then this strategy will fail.

### Place

Where does the shopper go to find your product? "Place" refers to your distribution channel—how you get the product to the customer. Marketing texts often refer to the "utility" of a product or service as its want-satisfying power. Two of the most sought utilities are *time* and *place*. Delivering software over the Internet scores high in both of these areas. You can provide your product *when* they want it (immediately), while they are located *where* they want it (at their computer).

So while conceptually satisfaction can be had, the tasks remain of getting your software listed and available. The user must be able to easily find and obtain your program. The essence of time and place is about making your program visible and accessible. *(See "Distribution Methods" on page 50.)*

### Promotion

Experienced salespeople know the value of a good "elevator presentation"—a fifteen-second concise, rehearsed explanation of your product and its benefits. Its name is inspired by the fact that your communication must be quick and clear enough to be successfully delivered during a brief elevator ride.

There are more than 40,000 shareware titles available, with new programs being spun every day. Drawing attention to your strand of gold in the middle of that huge haystack is tough. Nobody can try all the shareware available, so they download only the items

that sound useful and interesting. You have approximately thirty words to catch your audience's attention and convince them to try your product. *(See "Publicity" on page 51.)*

Chapter

3

# Limitation Scenarios

"Some persons are likeable in spite of their unswerving integrity."
—Don Marquis

## Market Realities

The sad fact of shareware sales: most people don't pay shareware fees. While an accurate accounting is impossible, an estimated micro-minority of 2% to 4% of all shareware users pay for the applications they use. Finding an appropriate strategy to boost that percentage is the topic of this chapter.

How do you persuade users to pay for your program? Adding an incentive like limiting the program's utility (also known as crippling) has dramatic results, immediately multiplying response in most cases by five times or more!

Limiting the functionality of a program until a person buys and registers it is the essential difference between trialware and any other form of software distribution. This bears repeating:

**Limiting the functionality of a program until a person buys and registers it is the essential difference between trialware and any other form of software distribution.**

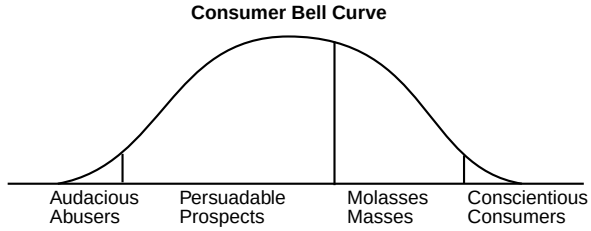
Every product distributed on the Internet has a place on the "limitation-payment continuum," ranging from *Full-Functionality-and-Payment-Optional* to *No-Functionality-Until-Payment-Received*. The trick is finding the optimal balance between these two to maximize your registration response rate.

You want to minimize obstacles to move smoothly and quickly through the purchase process:

user discovers your software
⇩
downloads the software
⇩
starts to use the software
⇩
learns its benefits first-hand
⇩
decides to keep using it
⇩
is willing and able to send you payment.

## Consumer Behavior

Similarly, there is a predictable continuum of consumer behavior.




**Conscientious Consumers.** These fine folks pay every shareware fee promptly, or else throw out the product—no incentive needed. They are dream customers. They are rare, so appreciate them, but don't rely on them

**Audacious Abusers.** At the other end of the bell curve are the "Gypsies, Tramps and Thieves" who will never pay for your program, or anybody else's for that matter. When we confront the criminal element, we expose a nefarious dark side. If your product is popular, it will immediately be hacked it so its full functionality is available for free. Or, someone will post one of your passwords. Personalizing your battle with them wastes time. Put reasonable protection in place, and focus on improving your product and persuading the next two groups to pay.

**Molasses Masses.** These folks would pay for your program if they could ever get around to it. Forgetful, preoccupied, or just

plain lazy, these folks have no ill intent—they just rarely get around to paying.

**Persuadable Prospects.** With some gentle prodding, these folks will do the right thing. If they actually use your software, they'll pay you for it... when prompted. This is probably the largest segment of your audience, so focus your persuasion efforts on them.

## General Guidelines

Carefully consider what inducements you will include to "force" people to pay. Most shareware authors do not completely disable their program at the end of the trial period. Time-bombing the program forces the user to make a choice of paying now or never, and they will often choose never when they would have chosen later had you left that as an option. Remember that unlike commercial software, trialware piracy generates sales.

If your program requires substantial user input to demonstrate itself (like a database-oriented or graphic-design program), provide a way for the user to save or export their data. Informing them up-front that they won't lose their efforts will make them more likely to put in the time and energy.

## Types of Limiters

### Simple Reminders


The simplest method is to display a "splash screen" when the program is launched, listing your requested shareware fee and payment options. You should also show this window when the user quits the application. "Nagware" as this is often called, doesn't deny any functionality to unpaid users, but it attempts to appeal to their sense of integrity. After registering, the user receives a way to kill the nag screens. For especially mild nagware you could include an option to turn off the nag even without paying. If you believe

that people are basically honest, then you'll feel comfortable that this is a boundary most users won't casually cross.

You should have two options to dismiss the reminder: one that closes the dialog (typically "Not Yet" or "Later") and one that lets the user enter the registration code ("Register" or "Personalize"). Don't allow keyboard input, as hitting a key can easily become habit or be automated with macro software. Require the user to read the screen by varying the button positions. A random number generator routine within your program can easily set which button goes right or left each time the dialog box is displayed.

**Interruptions.** Once or twice a session the software pops up a dialog box that asks for payment. The message is pleasant or neutral, but its unpredictable appearance creates an undesirable interruption. The software is fully functional, but users may pay to silence that annoying dialog box. If the user gets interrupted three or more times a session, they can get *really* annoyed.

**Delays.** An added dimension to the interruption technique is to make the user wait while your payment request is displayed. You can cause this to happen at launch, at random times during the session, or at vital program transitions. In general, a wait time of five to ten seconds is sufficient. This method can be particularly effective, depending on the type of application and the relative inconvenience of the delay.

**Branding.** If your program creates output of any form, you can embed or overlay an advertisement or message. For example, on image files you can overlay your logo or the word "Unregistered." When the user pays, the message goes away. If your application works with text-based files, you can automatically add a character string each time the user saves or prints.

**Functional Restrictions.** Another way to cripple your software is to disable the clipboard copying, file saving and printing features altogether. You may also confine the available functions to some subset of the complete feature set.

**Usage Limits.** You could limit the number of pages or records they can create, or require that they print one page at a time. You

might restrict the size of the document that can be produced, for example, to a maximum file size of 10 Kb, or a graphic image resolution to 400x300 pixels.

**Timeouts.** Timeouts are a highly recommended way to enable trialware. Keep in mind, however, that time limits are ultimatums—they force the user to decide by the end of the trial period. Given the options of register now or get locked out, the user may choose to withdraw. Be sure to allow sufficient opportunity for your prospect to gain an appreciation for your software before that decision point arrives. Timeouts come in three flavors:

***Startup Countdown.*** Each time the software launches, it decrements a counter. Limit the user to 20 or 30 startups, after which some important features are disabled. During the trial period all features of the software should be available. An alternative is progressive debilitation, where the longer the program is used without being registered, the less it can do. Warn them as they get close to the cutoff, as in "You have three sessions left in this trial. Register now to retain full use."

Games suggest a similar metering method, by counting number of rounds or levels of play. Show the user how often they really used the game, and they may be more likely to pay the shareware fee.

***Expiry Dates.*** Record when the user first runs your application, then provide full functionality for some reasonable period of time thereafter. Thirty days is a common limit, but be aware that users may get your program, launch it, then not come back to it for several weeks. Carefully think through the typical usage pattern to determine the optimal end date.

***Timers.*** The third and most adaptable way to limit time is by measuring actual usage. Track the hours and minutes the user has the program open. Each time the user starts the software you can have the splash screen show the elapsed time or time remaining in the trial. You may also offer an emergency extension feature at the end of the period. A "Give me a little more time!" button may endear you to the user. Of course, that button should have a limit, too. Alternatively, once the limit has been reached you can continue providing full functionality but show a message like

"You have been using this software illegally for 10 hours, please register now!"

Programmers use varying techniques to track the startup count, first launch date or accrued time. An invisible file may be written to disk, or the data may be stored in the Preferences file. Some users are savvy to this and will simply find and trash the file to refresh their trial status.

**Enhanced Version Upgrades.** You can offer advanced features for registered users. Do not exclude essential features in the trial version, but provide increased flexibility or extended options that add value to your program, which become available upon unlocking. Think carefully about which features belong in which version.

### DOs and DON'Ts

Don't use short trial periods. Few people will install your software and then devote the next two days of their lives to an intense testing of your software. Most people will download a program, launch it to see what it does, then not get back to it for a month.

Don't use a specific calendar date to determine when the software will cease to function. Shareware authors still get payments for three-year old software that people have just found and are just now trying. If you kill the software on a specific date, you will lose sales. Worse yet, you will increase your tech support load around that date when users complain your program broke.

Don't allow the payment request message to be dismissed through a keyboard action. Keystrokes can be automated, or become habitual. Instead, use on-screen buttons, and vary their position on the screen. This forces the user to read the message to determine which button to choose.

Some developers create complex password protection schemes to curb theft. Be careful—you don't want to risk locking out a legitimate user. Users sometimes move their software to a different hard disk or directory, or do normal hard drive housekeeping. Be flexible enough to accommodate those changes.

Don't threaten to erase their hard drive if they are using a known pirated registration code, because you will get blamed for the next bad thing that befalls their computer.

Do keep track of the pirated registration codes and lockout these registration numbers in future versions.

Do check for registration code validity in several places in your code, and vary the times and/or dates the checks are enabled (for example, odd weeks or even months). If a pirate publishes a patch for your software, your program will work for a while but regain its defenses. The cracker will have moved on to other programs, and everyone using the pirated version of your code will, once again, have incentive to register.

### Content vs. Application Software

Non-application content—text, sounds, pictures, or fonts, rather than an executable—presents a particular challenge in the trialware arena. You are unable to limit use once the content has been delivered.

You can create a delivery mechanism which encapsulates and dispenses your content, but this topic is outside the scope of this document. The shareware techniques that appeal to honesty or invoke guilt may still work, but there is no real post-sale control.

Once the user has the final file, there's no going back—the content has been released to the world. With application software, you can program it to revert to trialware or have the program display the registered user's name. Not so with content.

Two ways to effect a limit are to parcel out portions of the content or embed "tags" in the content. Examples of portioning are a clip of music or an incomplete character set for a font. Tagging can be done by overlaying a mark on a graphic, injecting static in parts of sound or music, or introducing random errors in textual content.

The best way to control distribution of content is in the initial sales process. Provide teaser samples and convince the prospect to buy the full package. After that, registration processes for content delivery programs are basically the same as any other application software.

Chapter

4

# Payment & Registration Schema

"Put not your trust in money, but put your money in trust."
—Oliver Wendell Holmes

## It's Official

The registration process is the gatekeeper to legitimizing your program's user. It is about confirming payment and unlocking your program. Registration is, therefore, one of the more important aspects of trialware. It deserves thoughtful deliberation.

- For you, the software developer, it is the harvesting link in the sales chain of your program.
- For the user, it is the key to ownership (or licensingship, to be more accurate).

Consider how the user will be able to send you payment and request a valid registration code, and how you will respond to the user. By what method will they enter the registration code, and how will your software need to be modified to make this whole process work?

In the "olden days" the user would find a program on a shareware compilation CD-ROM, decide to pay for it and mail a check to the shareware author. The author would then mail them an unlocked program on floppy disk, or send them a registration code by mail or email to unlock the software they already had.

Nowadays, the user probably downloads the program from one of the many shareware sites *(See "Top Shareware Download Sites" on page 57.)*, and when they decide to pay for it they want immediate access to the full program. They want to click on a button within the program that presents a form which handles the entire process online immediately. Also perfectly acceptable is taking them to an online order form, which immediately displays and/or

emails a registration code upon the confirmation of their payment.

There you have, in the contrast between "then and now," the two extremes of registration automation and speed. The closer you get to the current ideal, the more registrations—and hence, payments—you'll receive. Regardless of the actual mechanism used, however, every locked program needs a key. There are varied types of keys and each functions slightly differently. They even go by different names: registration key, activation code, licensing key, serial number, passcode, or personalization key. Here are some ideas about how to fashion your key.

## Types of Coding

**None.** The easiest, but least secure method, is to set up a Properties or Preferences item titled "I paid." If the user checks it, the reminder or limitation goes away.

**Universal Code.** You can create a single registration code that is given to every user. A variation on this theme is issuing codes based on source of sale. All users who buy through a certain affiliate get the same code, which is different from other users' codes.

**Random Code.** The secret registration code might be randomly constructed, but specific characters in the code are derived from other characters. Your software can check if the code matches the given pattern. For example:

- The numbers in positions 1, 5 and 10 total 15.
- The characters in positions 2 and 11 must be identical.
- Character 3 plus character 6 must be smaller or equal to characters 7/8.

**Based on User Data.** The registration key may be based on the data you get from the user. It is very common to create the key from the name of the user. Manipulate the ASCII values of the characters—multiply, add, invert, etc.—to get a unique code. You may also incorporate the current version number and number of licenses into the key.

**Secret Actions.** You can put a complete input dialog for the registration key into your application, but no value entered there

will actually unlock the program. Instead, the user has to do some secret action to free it. For example:

- "Open the about box three times and close it again, then quit the application while holding down the shift key."
- "Start the application while pressing the shift key, the open the Preferences dialog and type the keys U and L."

Regardless of the method you use, your registration system should collect some user information, which it then displays in the startup splash screen. This personalization removes anonymity from the copy, and deters casual piracy. Many software authors spend a great deal of time trying to prevent people from stealing their software. A saner approach is to accept the fact that some theft will occur, and redirect that energy towards improving the product. Some of the most successful software has the least restrictive protections.

After the initial registration process, a paid customer may delete the program, reformat their hard drive, upgrade their system, or misplace their registration key. You will need a fail-proof system which can verify a customer's paid status and re-issue a registration code. Deal with these requests immediately. Design a system that will minimize effort and grief for both you and your customers. "Hell hath no fury" like a paid user who is denied service because the developer is too protective of their codes. If in doubt, give it out.

In the case of non-application content, the registration process refers to the delivery mechanism encapsulating your content. The processes are basically the same as application software.

### Payment & Registration Service Vendors

Shareware registration services are intermediaries who ease the shareware author's burden in the process of collecting fees and issuing registration codes. These specialized services streamline the payment process for the shareware user, too, providing more payment options and ease-of-use compared to what the typical shareware developer can offer.

Services usually include accepting all payment types—personal checks, all brands of credit cards, and even invoicing—by fax, email, web page, or toll-free phone number. Many agencies now

accept multiple currencies and handle the conversion for you. The transaction is transparent to you, whether the user paid in Euros, Francs, Pounds or Pesos. Naturally, the easier you make it for someone to register your application, the higher the chance that they will do so. This outsourcing can take as much as 25% of the gross sale, but for most developers it's a wise alliance to off load that part of the operations headache. Besides, it is generally agreed that utilizing a known service can increase sales by 50 to 100%.

More than twenty different reliable registration agents are available, and they all operate under the same basic model. An identifier is assigned for each application you want enabled. With only a couple exceptions, no set-up fee is required. You then let users know the payment options via the documentation or in-program registration process. A shareware user who wants to register your program then goes to the registration agency, quotes the identifier of the program, and completes the payment transaction. The agent immediately provides a registration code picked from a list or generated by an algorithm you've provided.

The registration agency does all of the financial processing involved in clearing the payment and passes the proceeds, less their service fee, on to the shareware author. Also attached are all the user data, ready for importation into your customer database.

Note that shareware is more of a marketing strategy than a type of software, and the object of any marketing strategy is to increase exposure to your product. Therefore, it makes sense to select more than one service—especially where some have geographical specialization, as is the case for UK ShareReg (United Kingdom) and ShareIt (Germany), for example.

A comprehensive chart of agents is listed in the Appendices *(See "Payment & Registration Vendors" on page 55.)*, but here are several of the most popular.

***Atlantic Coast / CompuServe SWREG***

URL: <http://www.swreg.org>

This company is *the* father of the European shareware business, and has expanded world-wide. The involvement of the founder, Stephen Lee, dates back to 1984 shortly after the U.S. shareware

pioneers had gotten established. In January 1999 Atlantic Coast bought the CompuServe SWREG online software registration service from CompuServe—the oldest online store in the world—and became SoftShop. They now have phone and fax nodes in at least fifteen countries.

SoftShop charges a one-time $39.95 set-up fee, plus a per-unit fee with two commission options. You may choose either a flat percentage rate of 9.9% (minimum $2.50), or a $4 + 4% schedule. They offer modification to the latter plan at higher volumes.

***Kagi***

URL: <http://www.kagi.com>

Founded in 1988 by Kee Nethery in Berkeley, California, Kagi registers shareware, commercial software, hardware products, conferences, and events. Kagi is the Japanese word for "key," and is pronounced KA-gi: "KA" as in COpper, "gi" as in GEEse. They offer fax, mail, and Internet-based registration services.

Kagi offers two main ways for authors to enable software registrations:

1. Authors can include a small custom Kagi application with their products that users can use to generate an order form (with encrypted credit card details) which is emailed or faxed to Kagi to process. The applications can be run on Windows, Macintosh, or Palm operating systems.

2. Users can simply go to the Kagi site and register products. Kagi offers shareware authors an email alias and 5 MB of FTP space on their web server for dedicated product information pages, at a cost of 2% of sales.

Either method accepts many different payment forms, including US Check, Money Order, Cash (in several currencies), Visa, Mastercard, American Express, Diners Club, Discover, First Virtual, and Invoice (to be given to the

user's accounts payable department). Credit card information is scrambled by the register program. Each payer receives an email reply and once per month the authors receive a single payment from Kagi along with a detailed listing of all the transactions. Kagi is also able to pay authors by wire transfer of funds.

Kagi's website has an excellent storehouse of hints, tips, and FAQs for the shareware developer. They also just recently created an entire website on developing your own activation code generator (ACG), for which they supply the actual source code in Perl. This web-based CGI (common gateway interface) program, which you would host, keeps *you* in control of the key codes you release.

***UK ShareReg***

URL: <http://www.uksharereg.com>

UK ShareReg is a UK-focused shareware registration service. The obvious difference is that, while people can place orders using credit cards denominated in any currency, the products are priced in UK £ Sterling. Customers in the UK enjoy knowing exactly how much they are going to pay in their local currency. Authors can choose to be paid in either Pounds or US Dollars.

UK ShareReg is only an order-taker—you agree fulfill the sale, within their delivery guidelines:

- 48 hours if customer requested delivery by email.
- 3 days if they requested conventional mail within the UK.
- 6 days if they requested international mail.

UK ShareReg takes about 10% of the sale, and is the only service which provides an incentive for exclusive listings.

***Digital River / DigiBuy***

URL: <http://www.digibuy.com>

DigiBuy, owned by Digital River, allows the shareware developer to set-up, modify, and add products anytime via the DigiBuy website. You have complete control over pricing, order forms, and delivery options. You can even check order status and process refunds yourself.

The DigiBuy system allows you to store versions of your software for instant download by your customers. You can also

generate unique download links to your full version software located on their server. Like most other services, DigiBuy can automatically assign each customer a registration code for a particular product. Authors must supply their own list of predetermined registration codes or a link to a registration code program on a remote website.

***Digital River / Register Now!***
URL: <http://www.regnow.com>
Another service by Digital River (purchased from Universal Commerce), provides online registration and delivery. One-third of their shareware authors are non-U.S. They use an SSL encryption and possess a VeriSign certificate. They keep 20% of the sale, plus 1% if they host the download.

***Public Software Library (PsL)***
URL: <http://www.pslweb.com>
Established in 1989, Texas-based PsL provides both shareware registration and distribution for over 1,000 shareware authors. PsL began life as a shareware CD-ROM distributor, and lists over 20,000 shareware titles. Being an older service, their offerings are no longer competitive. They deserve mention, though, since they recently merged with DigiBuy (see above), and apparently will migrate all PsL accounts into DigiBuy's operations.

***RegNet***
URL: <http://www.swregnet.com>
Based in Santa Clarita, California, RegNet is a subsidiary of Wintronix, itself a Windows shareware developer. Like Kagi, RegNet has a small program that can be bundled with your shareware, allowing users to register through RegNet directly from their desktop. The user's name, address and billing information is encrypted before being sent to RegNet for processing.

***ShareIt***
URL: <http://www.shareit.com>
ShareIt is a German shareware archive and registration service based in Köln. They offer a standard web-based registration service, providing a registration key via email immediately upon confirmation of payment.

Shareware authors are offered a free custom web page on the ShareIt server. Remittances can be forwarded to a bank account upon transaction or on an agreed time cycle. Fees for money transfers are relatively cheap: money transfer to your account costs US $2, cheques US $5, Cash or Eurocheque with registered mail US $5. Eurocheque with regular mail US $1(Registered Mail US $5). Commission for each registration is US $4 + 4% of sale.

***BMT Micro***

URL: <http://www.bmtmicro.com>

An apparently stable business located in southeastern North Carolina, BMT Micro offers traditional distribution and fulfillment services in addition to all the standard online registration options. They provide diskette and CD-ROM duplication and world-wide shipping. They also offer package assembly and retail box shrink-wrapping. More uniquely, they are proactive in creating promotional opportunities for shareware authors, such as bundling or co-marketing alliances. All these functions come at a price, but if you're looking to enter the retail marketplace, BMT may be the one-stop shop you need. They also provide the option of author payment in Deutschmarks, and pride themselves on their customer focus.

***Get Software***

URL: <http://www.getsoftware.com>

A newer service, founded in 1996, GetSoftware offers some newer features. For example, they provide two post-sale customer acknowledgments—the standard confirmation of order, and a customizable message sent on your behalf. You can use the second message to thank the customer for their purchase or provide some startup or use tips. Other unique features include the ability to provide automatic upgrade pricing for existing customers, and to create special offers via "hidden" URLs.

***Quicomm***

URL: <http://www.quicomm.com>

One of the more affordable services, Quicomm charges a flat fee of 6.5% plus $0.30 and 2.5% for credit card transactions. There are no monthly minimums and no setup fees.

They can also supply you with C++ code to place a “Register On-Line” button in your software’s registration dialog. A click on this button takes a user of your software directly to your Quicomm registration web page via the user’s web browser.

Quicomm handles all credit card authorizations and charges, and immediately sends email to your customer with the registration code. You are immediately emailed complete details of the order, and can get up-to-date summaries of your account by month (listing each registration, transaction and Quicomm fee). Checks are mailed to you once a month or after a preset limit that you specify.