

Manual for Version 4.7
ServiceDesk for Windows, © Copyright 2014
by Rossware Computing, Inc.
290 E. Lighthouse Rd
Shelton, WA 98584
360-427-6000 voice, 360-427-7800 fax
www.rossware.net

- CHAPTERS -

1.	NOTES FOR THE NEW USER.....	7
2.	SYSTEM REQUIREMENTS.....	13
	A. About Your Video Display.....	13
	B. About Virus Protection	15
	C. Using Multiple Computers.....	16
3.	INSTALLATION AND SETUP	19
4.	GETTING THE BASICS.....	23
	A. Your First Look Around.....	24
	B. A Beginning Acquaintance Tour	26
	C. Thumbnail of the Structure	33
	D. Flowchart of a Typical Job Sequence.....	39
	E. Thumbnail of a Typical Daily Process.....	41
	F. Thumbnail of the Transition	43
	G. General Observations.....	44
	i. Simplicity versus Function	44
	ii. Messages: Cautionary, Steering, Informational	44
	iii. Bugs, Error Alerts and System Crashes	45
	iv. Adapting System to Practice: Should You Bend or Should ServiceDesk?	46
	v. The Scheme in Regard to Job Documents	47
	H. Making it Easy: Suggesting a Visit to an Operating Office.....	49
	I. Updates, Ongoing Support, Etc.	50
	J. The SlideShow Demo	52
5.	CALL MANAGEMENT.....	53
	A. Callsheet Navigation.....	53
	B. Desk Assignment and Status Options	54
	C. Handling Supplementary Information	55
	D. Methods of Auto-Insertion of Specialized Text	57
	E. Creating Addresses with Auto-StreetFind.....	64
	F. Using the ItemLocate and AutoDial Features	65
	G. Using the Customer Database.....	66
	H. Callsheets Hibernating, and the Overdue Alarm.....	68
	I. Scheduling From a Callsheet.....	70
	J. Creating a Job/Sale	71
	K. Miscellaneous Callsheet Utilities	74
	L. Callsheet Saves, and Name Interpretation	76
	M. Archiving Callsheets	79

6.	SCHEDULE AND DISPATCH MANAGEMENT	81
A.	The DispatchMap	81
i.	General Elements of, and Considerations in Map Design	82
ii.	How Your Map Displays Schedule and Related Information	83
iii.	Making Operational Changes to Your Schedule From The Map	86
iv.	The Check Off / Status System	87
v.	Automated Dispatching: A Cornucopia of Methods	89
vi.	Other Within-Map Tricks	93
B.	The ScheduleList Form	95
i.	Adding New Appointments	95
ii.	Changing or Canceling Appointments	97
iii.	Making Entries Regarding When Techs Are Not Available	98
iv.	Other ScheduleList Tricks	99
v.	Matters of Housekeeping	101
C.	The Zone-Scheduler System	102
D.	The Web- and Email-Based Dispatch Enablers	103
7.	JOB MANAGEMENT	105
A.	The WorkInProgress System	105
i.	The JobsCurrent Form	105
ii.	Scheduling on Existing Jobs	108
iii.	Reporting on Jobs: The PostVisitReport	110
iv.	The WipAlert System	118
v.	The JobsArchived Form	120
B.	Managing Special-Order Parts	122
i.	Birth of the Internal "PartsRequest"	122
ii.	Processing PartsRequest Items—General Concepts	123
iii.	Processing PartsRequest Items—Specific Operation	126
iv.	Taking PartsProcess Items "To the Grave"	132
v.	Managing Core Returns	138
C.	Managing Stocked-Parts Inventory	143
i.	Creating a MasterPartsPlan	145
ii.	Informing ServiceDesk of Your Already-Existing Stock	148
iii.	Maintaining Your Inventory	150
iv.	Advanced Features, Including Different-Stock-for-Different-Trucks	154
D.	Tracking and Depositing Funds	156
i.	Entering Funds Collected	157
ii.	Providing Absolute Funds Security	158
iii.	Dealing With Abnormal Situations	161
iv.	Housekeeping in the Funds Form	163
8.	POST-COMPLETION MANAGEMENT	165
A.	Recording Sales, Tabulating, etc	165
i.	ServiceDesk Utilities vs External Accounting	165
ii.	The SalesEnter Form	167
iii.	The SalesView Form	170
iv.	The Reports Form	172
v.	Separating Sales Into Multiple Departments	172
B.	Managing Accounts Receivable	173
i.	Standard Billing Reminders	174
ii.	Special HighVolumeClient Notices	177
iii.	Other Receivable Concerns	178
iv.	The Applications Journal	179
C.	A Note Regarding Organization	180

9.	THE FINISHED-FORMS INTERFACE.....	181
A.	Electronic Claims Submission	182
i.	Client Setup	183
ii.	Alternate Process Scenarios.....	184
iii.	Details on the Imported-data-Auto-Fill function, and the Re-Load Alternative.....	185
iv.	File-Saving Details as Relevant to Batch Uploads	186
v.	Reviewing for Claims Rejection, and How to Deduce Configuration Details	187
vi.	Integrating with the Sales Entry Process	188
B.	Point-of-Sale Operations	189
i.	Alternate Approaches to Initiation of the POS Ticket.....	189
ii.	Specifics When Using the Direct-POS Option	190
iii.	POS Integrations with Inventory and Parts Ordering	191
iv.	Executing a POS Transaction.....	193
v.	Accepting Returns.....	194
vi.	Customizing the Finished-Forms Interface	196
10.	OTHER FUNCTIONS.....	197
A.	E-Mail	197
B.	The Unit-Info System.....	199
C.	Making Reports	202
i.	Salary, Wage and Commission Reports.....	202
ii.	Completion Analysis	203
iii.	QOS Analysis.....	204
iv.	Tech Time Analysis.....	204
v.	Callback Reports.....	205
D.	Miscellaneous Printing.....	206
E.	Making a Customer Mailing List.....	207
F.	Finding Various Customer Records.....	209
G.	The Source of Jobs Survey	210
H.	Red-Flagging Problem (or other Special Situation) Customers	212
11.	UTILITIES	215
A.	SD-Backup	215
B.	SD-Tools.....	217
C.	Zips.....	218
D.	The On-screen Manual	218
E.	The Main Menu.....	219
F.	The Auto-Archive Feature.....	209
G.	The Settings Form	223
i.	Local Settings	223
ii.	System-Wide Settings.....	226
H.	Security Settings and Passwords	228
12.	REVIEW AND SYNOPSIS	231
A.	Command Summary.....	231
B.	Directory and File Structure	245
C.	Final Setup: Items to Check-off	251
13.	A BEGINNER'S GUIDE TO EASY, STEP-BY-STEP IMPLEMENTATION	255

- APPENDIX -

1.	PRINTING FROM SERVICEDESK	267
A.	The Physical Setup	267
B.	Creating a File that Specifies the Particular Format for Your Invoices	269
2.	SETTING-UP FOR ELECTRONIC RETRIEVAL OF MESSAGES	278
3.	CREATING YOUR YELLOW PAGES AD LIST	281
4.	TECHNICAL INFORMATION.....	283
A.	Specifics Regarding the Customer Database System.....	283
B.	Details on the StreetList System	287
C.	Underlying Machinery in the DispatchMap	292
D.	How ServiceDesk Interprets Appointment Notations.....	293
E.	How ServiceDesk Collects Data for a Finished-Form Document	296
F.	Dealing with Divergent Sales Tax Rates	299
G.	Resolving Possible Right-Click Problems	301
H.	Interfacing with Your System of Financial Accounting.....	302
I.	Configuring for Remote Use and/or Secondary Offices	310
J.	Details re Compatible Networking Strategies within Windows	313
K.	Focused Dispatching, A Solution for the Large Enterprise	316
L.	Automating Communication with your Customers.....	317
M.	Using our DispatchLink Utilities.....	318
N.	Splitting Commissions on a Job	319
5.	YOUR OWN ZIPS, CITY LIST, AND OTHER CUSTOM INFO.....	323
6.	INDEX	325

- EXHIBITS -

Sample Answering Service Printout Showing a ServiceDesk Usable Format	335
Sample Invoice and Job Printout.....	337

Chapter 1

NOTES FOR THE NEW USER

Congratulations.

You have wisely availed yourself of the very cutting-edge in service systems management. Unlike anything else on the market, this is custom software. Please understand, we've done something no other software developer has dared even attempt for the general market—creating a unique, electronic map of *your particular* service territory, one that will instantly show each of your scheduled jobs graphically, temporally, and in reference to the tech-assigned. We've created a list containing just *your* streets, with grid references to the paper map you use, and, more importantly, with each such street also ready to cross-reference to positions on that magical electronic screen (see section that begins at page 323 for a summary of your custom information). There is simply nothing else like this, offered anywhere, and we've only described the beginning. You're in for a treat in the coming weeks as you discover the plethora of wondrous things ServiceDesk will do for you!

This manual is structured to help you get ServiceDesk setup and productively running within the shortest possible time. You are probably anxious to experiment, and obviously there is no need to read very much here before doing so. Simply read what you need in terms of installation and setup from the second and third chapters (if needed). Then, you may either launch immediately into unguided exploration, or follow the “guided” exercises that begin in Chapter 4 (“Getting the Basics”). Regardless, those guided tours are helpful, and we seriously urge you to read and follow them before proceeding past in the manual.

After doing so, you'll find a series of discussions (still in Chapter 4) that are designed to give you a broad-brush understanding of ServiceDesk structure. A “big picture” comprehension is essential if you're to avoid drowning in details later on. It will give you a place to mentally “hang” every detail as it's later described.

Speaking of which, there is something important we must explain about ServiceDesk: Its initial implementation is not completely easy. There is so much the system does, and so much of is novel in terms of *how* it automates and simplifies, you have a lot to learn. Obviously, learning takes dedication and effort. Indeed, it will be a little like learning to ice skate. While still in the uphill effort, it's no fun, but as you start getting the hang of it, things will begin to fly effortlessly and with grace.

In designing software, there is a tradeoff. We could give you a system that would be completely easy from the start. Problem is, it wouldn't do much. If you want a system that makes your life *as easy as it can possibly be*, that system must include mechanisms to cope with the huge variety of tasks you perform. And there are many such tasks, for the service business is a very complex one. Unavoidably, if comprehensive, software must incorporate the same complexity, producing an irony. While ServiceDesk indeed makes every operational task more easy, simple and effective than you ever dreamed possible, your *initial* task (again, when learning how all the pieces fit) will not be so easy.

This is our way of saying you need to hunker down and commit yourself to serious work. You will not succeed without discipline and focus. It's back to school now, and must be. The world is charging ahead, and you mustn't be left behind.

In particular, please focus on the "investment" concept. You *know* what it is: make an extra expenditure now (perhaps even one that you're *stretching* to make, and whether consisting of money, *effort*, or whatever)—for the sake of reaping a long stream of much greater rewards (that repay the investment over and over again) later on. Consider that he who refuses to invest fails to reap that wondrous/endless reward stream. Time and again, we've seen businesses purchase ServiceDesk while being unwilling to invest anything more than a trifling effort in learning how it works. It's like they're shooting themselves in the foot. For months and years after, they continue to work much harder than need be (and often with greater frustration)—simply because they were unwilling to invest more reasonably up front. Please don't let this be you.

And please don't say you're "too busy" to make this investment. You purchased ServiceDesk for the very reason you need to get more work done in less time, and that transition cannot occur unless you *make* the time to learn (find it, create it somehow). You'll be in the endless circle of "not enough time, not enough time") until you do.

We have, of course, worked to ease this learning process—in your conversion to modernity—as much as possible.

To begin, please understand you're not expected to read all of this *very thick* manual. Much of what's here is *resource* material, to be consulted only when needed. What we do expect you to read is all of this chapter, and onward through Chapter 12.

As you're focusing in particular on Chapters 3 through 8, please bear in mind each of those has a corresponding video lesson. At the top of each such chapter is a boxed paragraph that indicates the particular lesson you should watch in its connection. Since "a picture is worth a thousand words," we strongly urge that you watch each such corresponding tutorial—in conjunction with reading the chapter—as a packaged exercise. The reading will be more meaningful if having watched the video, and vice versa. Indeed, for greatest benefit (and if you can make the time), we suggest sandwiching two viewings of the video around a single reading of the corresponding chapter.

As another means of easing your learning process, we've deliberately structured ServiceDesk so you can, if preferred, implement its various areas of operability one step at a time. This can make it easier, for instead of having to swallow so huge a chunk at once, you can do it in more easily digestible bits.

To implement this step-by-step system, you should begin by proceeding normally through Chapter 4 ("Getting the Basics"). Then, skip ahead to Chapter 13, which describes and will guide you through the step-by-step program. One of the benefits (besides the fact you'll likely find the transition easier), is you'll get started in actual ServiceDesk use much sooner. The reason is you only have to understand a single area before using it. By this means, you can start using particular areas of ServiceDesk within days (or even hours) of receiving your package.

As a general recommendation, we urge you against being too *impatient* in the desire to see each and every function from the get-go. The learning process is a layered one, and it's hard to understand more advanced layers until you've first developed a foundational understanding of more basic ones.

Besides the above aids, we are also available to give you direct assistance—whether via plain telephone conversation, email, or by linking directly to your computer and working as though sitting by your side.

In our view, the learning process is appropriately perceived as a team effort.

You must be the primary instructor, a self-taught student who diligently uses the self-teaching materials we've provided (including, of course, this manual). We expect you to put forth real labor and effort in this regard.

However, we do not expect you to struggle, especially to the extent of getting significantly frustrated—with anything. If there's a matter on which you've exerted reasonable effort, and you're still not finding the answer, please give us a call so we can help. While it's true that, on the one hand, you can't reasonably expect us to personally spoon feed you all the details about everything, it's also true that if we can save you from an hour of frustration with a simple and quick telephone call, we absolutely want to do it.

In short, we expect you to do your part, but we want to do ours as well.

In terms of exerting reasonable effort, *we do expect you to read*. One of the frustrating things for us (so that you know) is when someone calls wanting us to give them a long and detailed one-on-one teaching session regarding some aspect of operation, and they've not even read the applicable section in the manual. That's not fair. Your ServiceDesk purchase does not include the involvement of our time for that kind of instruction. It does include the involvement of our time if, by chance, you've done your own best possible homework first, and still need assistance.

In specific regard to the Appendix, it contains resource material that may be consulted if needed, but otherwise should be ignored.

Even in the chapters we expect you to read, there are many footnotes. You may skip or read these, at your discretion. However, with their added insights, ancillary information and sometimes humor, we think you'll find those notes are helpful.

In contrast, you'll also notice there are many cross-references. These we suggest you ignore unless feeling a particular need to pursue the referenced matter (they're really intended for the situation where, months or years down the road, you've gone back to a section when wanting to research a particular matter).

In regard to illustrations, you'll find this manual has few. At one point it had none. From time to time on revising sections (and when realizing an illustration would be helpful), we've added them. Eventually, we hope to have it well illustrated. In the meantime, when reading about any particular feature in ServiceDesk, you'll find it greatly aids your learning process if you'll simply display that same feature (and experiment with it) in ServiceDesk.

So, having now described the general plan for your learning process, we'd like to proceed with a conceptual overview, explaining how ServiceDesk fits with other software systems. This will help you get some bearing regarding the kind of system you've purchased.

To begin, you should understand ServiceDesk is designed to address the needs of any operation whose primary task is sending personnel to a series of locations, throughout the day, performing jobs that will typically be completed within one or a few trips. In other words, it's ideal for appliance servicers, pool

servicers, pest control companies, drapery cleaners, carpet cleaners, office equipment servicers, drain servicers, as well as plumbers, electricians, HVAC service companies and others.

In terms of business size, ServiceDesk is designed to fit anything from a one-man shop to operations up to about 60 techs. If your shop is small, please don't be tempted (on reading various descriptions in this manual) to believe ServiceDesk is not for you. Though it has ample capacity for large operations, we've simultaneously kept it optimized for even the smallest. Indeed, our *average* client employs just three or four techs, and we have more one-man shops than any other size. There is some discussion in this manual you may disregard if your operation is small (stuff that applies *particularly* to larger operations), but rest assured the fundamental systems remain ideally suited for you—and when harnessed will help you just as much as larger operations.

Whatever your business size, if it's to be modern, well-managed and efficient, it must address software needs in at least five areas. Four of these, bookkeeping/accounting, word processing, inventory control and customer database usage, are not unique to service. Such needs are found in virtually all businesses, and for years there have been many well-developed softwares to address them. For businesses involved in a service-call-performing environment, however, there's a unique fifth area of need, to manage the operations of call-taking, dispatch, and work-in-progress. A few packages have expanded from beginnings in other areas to peripherally address these more specialized functions, but ServiceDesk is the first to attack them directly, squarely, and thoroughly.

Happily, ServiceDesk also reaches beyond its own home turf to peripherally address some of those traditional areas, including inventory control and customer database functions. It also steps partially into bookkeeping/accounting, as it tracks sales, does billing, accounts receivable and funds control. Still, it does not do check-writing, compile your income statement or balance sheet, or do word processing. We've added to its core functions only to the extent there were no well-developed alternatives, or where it's easier to handle processes in the seamless context of a unified, single package environment.

Additionally, ServiceDesk now has superb PointOfSale functions (POS), which is an area we did not build into it initially.

Even so, we've intentionally avoided the temptation to make ServiceDesk do everything. Because of this, you'll need to separately address word processing, check-writing and financial accounting. Generally, you'll be very well-covered in those areas with a copy of QuickBooks and Microsoft Office.

But in regard to the rest, you'll be superbly covered by letting ServiceDesk handle the third and fourth traditional areas (inventory control and customer database usage), along with *some* bookkeeping/accounting, and many other incidentals too numerous to mention. In particular and most especially, prepare yourself to enjoy our uniquely-focused fifth area, where we so ably address the fundamental operations of call-taking, dispatch and managing work-in-progress.

In concluding this introduction, two words of caution:

First, ServiceDesk has a plethora of features designed to alert if any aspect of continuing maintenance, in regard to jobs and their management, breaks down. These work superbly when you are actually using ServiceDesk, but can be a nuisance otherwise. Specifically, if you create a few fictional records while initially playing around, then leave them for a few days, you're likely to see a few warnings pop up. To at least minimize these warnings, please do not turn-on either of the two WipAlert features (within your ServiceDesk Settings form, see page 118) until such time as you begin using ServiceDesk for real operation.

Our second warning concerns paying attention to ServiceDesk messages. To some extent, you're going to find ServiceDesk is like a devoted (but potentially nagging) spouse. If you pay attention and are solicitous to its communications, there will be harmony and happiness. If you turn a deaf ear and ignore, there is likely to be nagging, nagging and more nagging.

We're not just being metaphorical. ServiceDesk is involved in managing a lot of things, and sometimes it needs your attention in respect to particular matters. It generally requests attention via a message. If you pay attention, and address the matter as requested, all will be good. If you do not, you'll likely get that same message again, and probably others as well. If this is your pattern, before long you'll find yourself being all but pelted with messages, continually throughout the day, every day—because there are so many matters ServiceDesk is pleading with you to pay attention to. It will be, literally, like the misery of living with a constantly nagging spouse.

It's a "vicious circle" thing too. If you fail to pay attention initially, you'll get more messages, then more. Eventually, in this pattern, you'll find it makes "total sense" to ignore messages. After all, it's impossible to read so many. The only mystery will be why ServiceDesk "bugs" you so much. Any message that comes up, you'll just be like "click, click, click," right through, without paying the slightest attention. And you'll be monumentally less efficient, in all kinds of respects, because of it.

If you pay attention, by contrast (and address the matters being requested), you'll get very few messages – so few that it's supremely easy to pay attention when the rare one comes along. You'll be happy, ServiceDesk will be happy, and all will be well.

Pay attention, please. Don't go click, click, click.

Again, thank you for buying a ServiceDesk package, and congratulations. You are now well on the way to making your own business more modern, efficient and profitable.

Chapter 2

SYSTEM REQUIREMENTS

For its operating foundation, ServiceDesk requires Windows. Any Windows version, XP or later, should work. For sterling (and maximally reliable) performance, we recommend at least a Pentium level processor, the higher the clock speed (and more cores) the better, with at least 4 gig of ram. Hard drive space is not an issue, as ServiceDesk and all its data will hardly dent even the smallest drive you can buy today (the actual installation will consume less than 20 megs; and over time even the data is unlikely to exceed a gig).

As a general proposition, you do not need a high-end machine, especially if you do not load it down with tons of garbage.¹ We think mid-level is generally a good way to go. Regardless, if you're looking for machines on the economy side (or wanting to use an older unit), our biggest recommendation is *don't skimp on ram*.

We believe in snappy performance, and aside from a machine loaded with garbage, the second most common factor causing slow performance is insufficient ram. It can easily be added, and at moderate expense. We think 2 gigs is the bare minimum for decent performance. If you tend to run lots applications simultaneously, you'll want more, perhaps significantly more.

A. About Your Video Display

In recent years, the monitor world has changed dramatically. Superb LCD monitors are cheap and plentiful, to such an extent we recommend going big—either with a very large monitor at each station, or with multiple smaller ones. More display space enhances the efficiency of your work.

If you happen to still have any CRT monitors (it's the kind with a heavy glass picture tube), we recommend upgrading. While CRTs work fine, LCDs are so much better, and considering how economically they can be purchased, to us it seems like a no brainer. When we retired our last CRTs, it was painful because they were high-end units and still worked perfectly. But (and to be candid), it's great to be rid of them.

¹On connecting to clients' machines, we are frequently appalled to see horrendous levels of performance. On many, for almost every operation there's a lag, and you have to wait, sometimes for seconds. It's stupid. Computers are our servants; they should have to wait for us, not vice versa. For my own use, I refuse to accept such lags, and so should you. Significantly, I've often found these laggard machines have higher raw performance specs than my own systems, and yet run dramatically slower. The reason: they are laden with a plethora of installs that bog the systems down. Don't do that to your business computers. Install only serious and professional business applications that you truly need. Keep it clean. Implement good virus protection and invoke regular system maintenance. It really matters.

On picking a monitor, pay attention to *resolution*. In case you don't know, that word refers to the number of pixels² that make up the screen—meaning a resolution of 1024 x 768, for example, has 1024 columns of pixels running across the screen horizontally, and 768 rows running from top to bottom.

As it happens, that particular resolution (i.e., 1024 x 768) was until several years ago on the high end. It's also the minimum resolution ServiceDesk requires, so you'll need at least that much. It will not likely be an issue, because today that level of resolution is small potatoes.

Speaking of small potatoes, please do not go that way by way of resolution. This is a parameter where big numbers are definitely better (indeed, much better). Compare your monitor's resolution to a physical desktop. When the latter is bigger, you can spread more papers and other tools on it at once, and thereby have each such item more readily accessible for increased work efficiency. It's exactly the same with greater resolution on a computer screen. It allows you to show (and simultaneously work with) more stuff on your screen at once. It's our experience that every time someone moves to a larger resolution, they're enormously glad they did, and kick themselves for not having done it sooner.

Please notice our focus is on resolution, and not physical size. What's the difference? It's a good chance that in your living room there is a 60" big screen with 1080p resolution. The expression "1080p" is shorthand for a resolution that's 1920 pixels wide by 1080 pixels tall. It's a rather good resolution, but you can get exactly the same in computer monitors ranging anywhere from 21.5" to 27" in size. In other words, these dramatically smaller computer monitors will display every bit as much information (about 2.1 million pixels total) as compared to the much larger entertainment unit (obviously, the computer monitors' crosshatch of pixels will be much finer in scale). In fact, you can buy a 30" computer monitor with a resolution of 2560 X 1600 pixels (about 4.1 million pixels total). It's one-fourth the physical size of that 60" big-screen, but can display twice as much information (meaning its display is actually eight-times more dense).

Again, it's workspace you should be looking for, and workspace is defined by resolution—not physical size. Even so, you may want to pay some attention to size as well. To illustrate why, let's suppose you had really, really terrible vision. If your on-screen display was so large in scale as to be spread over the canvass of that 60" 1080p big-screen (with your face, say, 24" from it), you'd likely be able to make out normally-displayed computer text just fine, in spite of your terrible vision. Shrink the exact same imagery down into a 21.5" display, by contrast, and (with awful vision) you're likely to find the text is much too small to make out. What this means in essence is, for compromised vision, you may want to consider size as well as resolution.

If for example, you've decided 1920 X 1080 is indeed the resolution you want in a monitor, it may make sense to go to a 27" size in this res, as opposed to a 21.5". The imagery would be considerably bigger, and easy to make out for eyes over 50. For younger or well-corrected vision, on the hand, even the finest imagery should be fine.

Speaking of bigger on-screen imagery, there is another way to get it. Virtually all monitors can be set to run at resolutions lower than their native resolution ("native" is whatever resolution is their max). As an example, you might own the 21.5" monitor above-described (native resolution of 1920 X 1080), but have it set to run at 1280 X 800. This would make all the imagery on the monitor appear bigger than if it was running at its native resolution, but you'd have less than half the effective workspace (prox 1 million pixels as compared to 2.1), and would essentially be wasting the monitor's much greater capability.

²If you don't know what a "pixel" is, think of a sheet of graph paper. Imagine coloring some of the boxes red, some green and some blue—in an arrangement that assembles a picture. If it's a finer cross-hatch (thereby making smaller boxes) you can obviously make a more convincing picture, as compared to if the boxes are larger. All display screens work on this principle, essentially building a picture based on an underlying cross-hatch of squares that are each colored as needed to assemble the picture desired. Each particular box is what's defined as a "pixel." The word derives from the expression "picture element."

We hate to see people do this. We believe it makes much more sense to take full advantage of every monitor's native resolution (i.e., always set to use the highest available). Simply pick monitors that are bigger in size for a given resolution (for eyes over 50), or put on some reading glasses, if need be. This is how important, in other words, we think it is to maximize your effective workspace.

One thing we've noticed in regard to the above is when folks have in the past become habituated to seeing larger on-screen imagery (and then first move to a more condensed resolution by changing the setting to native), they will often at first hate seeing everything smaller. Sometimes, they really, really hate it. However, where there is a willingness to simply get used to it, we find the same persons often grow to love their greater effective workspace—to such an extent they soon feel the smaller imagery is a small price to pay.

Besides having a single monitor that's high in resolution count, another way to achieve increased on-screen workspace is by employing multiple monitors. Many Rossware clients do this. It's highly effective, and can be significantly more economical than going for the biggest single screens. A grouping of two, three or even four small to mid-size monitors can make a great expanse of combined workspace. The division of workspace into multiple screens can also work as an effective organizer.

B. About Virus Protection

The first thing to know about virus protection is, be sure you have it. The second thing to know is, be sure it is NOT Symantec's Norton, or McAfee.

To be very candid, we have grown to *loathe* those two programs. We have seen them cause trouble and frustration in system after system. Such problems have cost us huge amounts of time, and our clients too (even when not causing direct operational faults, they slow systems down). It's reached the point where we consider those two programs *viruses in and of themselves*!

It's true, they are extremely prominent in the marketplace—but we don't think it's related in the least to superior quality. Our conclusion is the two underlying companies are good at one thing. It is creating relationships with other large entities (especially computer manufacturers) via which they get themselves installed on the majority of machines without any regard to quality.

To add insult to injury, those two programs are also expensive.

If you're using Windows 7 or above, a much better option is to use *Microsoft Security Essentials*. It works well, does not cause trouble, and has an excellent price (free).

If you're dealing with a pre-Win7 platform, our present suggestion is to use *AVG*. It's also free (at least for the basic version), and has traditionally been trouble-free. Just Google "avg free" and it will take you to the website where you can download and install. You'll encounter an interaction that attempts to "sell you up" to a paid level of service, but we have encountered nothing to suggest the base level is not adequate for most applications.

C. Using Multiple Computers

If you're using only a single computer, there's no need to read this section. But, if your office has more than one person (like at least a boss and secretary), there's little doubt you'll want a separate ServiceDesk station for each person that's actively involved in taking calls—and for the boss too, whether he or she takes calls or not. After all, ServiceDesk will be your window into the realities of managing your operation, and in communicating each element of information throughout your office and between your office and its techs.

If you do not presently have each office person equipped with a computer, install ServiceDesk on whatever you do have. As you begin witnessing its utility, you'll soon want to expand. Don't be alarmed over the costs of such modernization. You're already licensed to use ServiceDesk on as many stations (as connected to one business) as desired, and adequate systems can be found at less than \$400 a piece. The investment will pay for itself quickly.

Obviously, if your various stations are going to drive the same ServiceDesk system, they must have a means of sharing information. This is where *networking* comes in. Many users are already familiar with networking. If you are not, be assured it's generally very simple. If you're networking just two computers, they can be connected with a single cable (called a crossover cable), and no other hardware is required. If you're connecting more, you'll need an added piece of hardware (alternatively called a hub, switch or router). Each computer connects to that hub, allowing each to share information, printers and internet connection, with others. You can make this system either wired (conventional) or wireless. Wired is generally cheaper and produces better performance, but if your situation makes it difficult to run wires, wireless may be an easier solution.

Once your network is physically setup, you'll need to configure Windows to properly talk (i.e., one computer to another). This is done differently depending on the Windows system. XP and Vista have Wizards to walk you through the setup, and usually that works fine. If it does not, this is one place where (if you do encounter trouble) you may want to call in the geek squad. Someone who's adept can solve most problems quickly. For the rest of us, network inoperability can lead to hari kari.

Even when Windows is talking, one computer to another, there's still one more Windows step, to make such talking accessible to ServiceDesk. You need to decide which computer is going to be the effective "server" (i.e., the particular one whose drive will contain the data that all will read and write their ServiceDesk data to).³ At that computer, you need tell Windows you want to make it's drive (or at least the ServiceDesk folder on its drive) accessible to other computers on the network (this is called "sharing"). Then, you need to go to the other computers (we'll call these the "stations") and tell each you want it to "see" the first computer's drive via a particular drive letter (this is called "mapping").⁴

From the perspective of ServiceDesk, running in a networked environment is all but automatic. You'll find that every station provides an equally convenient window into every facet of operation, and in communicating between stations.

³This does not need to be a special computer. It can be any ordinary computer, even one that does double duty, acting both as the effective sever and as an ordinary work station. You *can* buy computers that are specially configured for use as servers, but for ServiceDesk , at least (and for ordinary size operations), we don't think this is either necessary or needed.

⁴For a detailed discussion on sharing, mapping and other networking details, see the Appendix section that begins at page 338.

If you're not ready to setup on a network yet, don't worry. As stated, ServiceDesk is sufficiently happy running from just one station.

Most of all, please do not make the mistake we witnessed with one new user. We were connected to their system, helping with an issue, and discovered they had three computers. Each had ServiceDesk installed and running. However, they were not configured to share data at a common source. Instead, each was writing and reading data to and from its own unique C drive. So, they were each dealing with totally unique data. We don't know how they even pretended to function in this fashion.

Positively, this is NOT how it should be.

Emphatically, each ServiceDesk station should be a window—into precisely the same data set—as every other. If yours is not working that way, please fix it.

Chapter 3

INSTALLATION AND SETUP

Hint: Read this chapter in conjunction with watching Video Tutorial Lesson # 1.

Installation is the same as with any Windows software. Place the CD into its drive, and, when the auto-menu comes up, select “*Full Install.*” The setup program will start, and you’ll confirm a few thing along the way (i.e., that you want to go ahead and install, etc.). To make it easiest, in fact, you can simply hit the ‘*Enter*’ button on your keyboard as each query comes up (this is because in virtually every instance the pre-selected, default answer will be the best one to choose).

As long as you’ve got the CD’s auto-menu displayed (and assuming you’re in a business that involves appliance service), we suggest you also choose the option to “*Install SmartParts Files.*” This will install model and parts lookup data that, for appliances only, will be handy in several ServiceDesk contexts.

After the installation completes, you’ll need to run ServiceDesk. This is, again, the same as with any other Windows product. The install program will have created a menu group, within the Windows ‘*Programs*’ folder. Though it varies slightly with Windows version, the general path to get there is by clicking on the Windows ‘*Start*’ button, then selecting ‘*Programs*’.

Look for a new listing labeled ‘*Rossware Computing*’. Click on it, and you’ll see it opens to display a series of program shortcuts that were installed for you—including (most particularly) one for ServiceDesk.

At this point you could click on that shortcut, to run ServiceDesk. However, it’s slightly annoying to have to go through this menu sequence every time you want to start a program that you’ll be using as frequently as ServiceDesk. So, it would be a good idea to place a copy of the shortcut on your desktop.

To do this, click⁵ down on the ServiceDesk shortcut, using your *right* mouse button. *Hold* the button down as you drag to a vacant space in your desktop. Now *release* the button, and look in the popup menu that appears. Find the listing that says ‘*Create shortcut here*’, and click on it. That will place a shortcut on your desktop, which you may now conveniently use to start ServiceDesk.

⁵ In case you’re not aware, the convention in any Windows-related context, when discussing the use of mouse buttons, is to refrain from stating left or right when, in fact, it’s the *left* button that’s intended. Thus, be assured that if any instruction refers to clicking but does not designate a side, it’s the left button you must use. Conversely, in all cases if the right button is intended instead, you’ll see the word “*right*” somewhere in the instructing sentence (as, for example, in this instance).

A. First-time Running.

When ServiceDesk first runs after a new installation, you'll be directed to the *Settings* form. This is the place where you specify several things about your operation—things ServiceDesk needs to know in order to serve you properly.

You'll notice there are two sections in this form. The first, shaded in green, pertains to matters that are local to the particular station on which you're running. The second, shaded in purple, pertains to matters that affect all the ServiceDesk-equipped stations running in your network. Even if your system will consist of just one computer, these system-wide settings are also critical.

As prompted by the system, you'll begin in the *Local* (green) section. Initially, all you really have to do here (and again, the system should prompt you) is type your name (at least, assuming you're the one who'll be using this station; if it's someone else, type their name). Type it very simply, as just first and last names (e.g., two "words," as in "John Smith").⁶

If you're already appropriately networked, you'll likely want to also designate the drive designation for the server at this time (see page 223), but assuming you're a newbie (and just experimenting for now), don't even worry about it. Instead, after typing in the appropriate name for the person who'll be using the station, click on big button labeled 'Save Local Values'.

Now you'll be prompted to specify values in the 'System-Wide' (purple) section. In fact, ServiceDesk will itself fill-in the first value for you, which is the name of the person who'll be using the station (as just provided by you a moment earlier)—inserted to a box that's intended to contain a list of all the names as involved at each of your ServiceDesk stations (assuming, or course, that you have more than one person and one station).

You could at this early point go ahead and insert such names to that list, and you could as well insert names to another box, just below, that's intended to contain a list of all your technicians (or, of just the one, if you're a one man operation). Or, if you want to zoom forward quickly now and get back to those tasks later, just proceed straight to clicking on the big button labeled 'Save Net-Wide Values.'

This will place you completely into ServiceDesk run mode. The essential elements of ServiceDesk single-station setup are not complete.

B. Installing on New Computers, Months or Years Down the Road.

If you're like most users, you'll likely add a new computer or replace an existing one from time to time. If this is months or years after we first shipped your package, there's a small complication. Your original installation CD will be out of date. So what do you do?

⁶Please note that this simple, first-name-then-last format applies *only* to the names of you and your staff as entered into this ServiceDesk *Settings* form. By contrast, for customer names (particularly as entered into a Callsheet when taking a call), we expect you to use the somewhat more formal last-name-comma-then-first format (see page 27).

Basically, use the old CD to do the installation. Afterwards, simply do an update, using standard update methods.

In that regard, we'll mention here that ServiceDesk has a semi-automated self-updating feature (click on '*File Functions*', and you'll see the option to '*Update System*' about seven items down). We are constantly improving the system, and suggest you update your system with some frequency. In fact, we post updates to the website, on average, about twice a week. Most of these are minor, consisting of small improvements or corrections, and it's probably not worth your while to try to keep up with every one. Some are much more major, however, and regardless, over time the improvements accumulate. You should plan to update *at least* once per month, so you can take advantage of these continuing improvements.

Whenever you update, it's helpful to know what improvements have been made. After all, it's hard to take advantage of cool new features if you don't know what they are. For this purpose, we maintain an ongoing diary on the website, which summarizes each improvement as it's made. In each instance, we put in enough information to provide guidance that's needed for you to harness the new feature. ServiceDesk will automatically open that diary when it updates itself, and we suggest you read the recent entries (as deeply as needed to get to where you left off with your previous update). You can also navigate to the diary via links on our main webpage (rossware.net).⁷

To actually perform updates (and if you're a new user), you'll first need to setup a user name and password. Just give us a call and we can do it over the phone. Then (one time only) you'll need to enter that user name and password into spaces provided in the *About ServiceDesk* form (accessed under '*File Functions*').

C. Another Note — A Plea, Really — on Learning and Implementation.

Over the years, we've taken on a good many ServiceDesk users. As users learn to use the system, we learn about users. This little section concerns one of the very most important things we've learned.

Here's a simple fact.

Some of our users love the system to the absolute Nth degree. I mean, they are incredibly ecstatic about its power and all it does for them. We hear comments like "anyone who doesn't use this system is an idiot." We could, literally, almost fill a book with ecstatic quotes. Folks develop missionary zeal, anxious to spread the "good word" about how absolutely fantastic this system is.

Some users don't love it quite so much. Often they find that, yes, it *is* a good system. They even be *quite* fond of what it does for them. But as for shouting from the rooftops, no, they just don't quite feel that way.

What's the difference?

⁷To make the desktop icon, go on the internet to www.rossware.net. On the left-hand side under ServiceDesk, click on Support. In the page which then appears, right-click on the hyperlink for ServiceDesk Work Diary. In the menu which then appears, select Copy Shortcut. Then right-click in an empty space on your Windows desktop, and from the popup menu select Paste Shortcut.

It's very simple. The passionate people truly and fully use the system. The less passionate ones use it but marginally.

Given this difference, we want to plea with you—really, we want to beg—please find within yourself the willingness to dedicate real effort, determination and commitment toward the task of achieving a thorough, deep and robust implementation.

Our goal is to dramatically transform your business for the better. We've created the necessary systems, but real and true implementation depends, ultimately, on you. We can't do the pushups for you. You've got to be willing.

If you are but willing, we can and do promise you're going to find the rewards are great. You're going to get much more enjoyment out of running your business. Your customers are going to love you more. Your employees are going to be happier in their work, and feel happier toward you and each other. Not least of all, you're going to be substantially more wealthy—even while in the long term working less.

But not without pushups first.

The *great* rewards will come with vigorous, thorough and intelligent implementation. Your long-term rewards will be commensurate with the amount of work you perform right now.

Please let it be sufficient for the kind of implementation for which we've designed.

Chapter 4

GETTING THE BASICS

Hint: There is no lesson in the Video Tutorials that specifically corresponds with this chapter. However, our ServiceDesk *Presentation Video* suits the purpose moderately well. It may be played via your ServiceDesk Installation CD. Just load the CD, and select the appropriate option from the resulting auto-menu.

Please think of this chapter, in its entirety, as a “mini-manual” for using ServiceDesk (in contrast, chapters 5 through 12 may be considered the *in-depth* portions of the manual).

We approach our mini-manual topics in five sections.

The first will walk you through a quick look at some of the things you may be most anxious to see. It also provides some very basic, beginning orientation.

The second delves a bit deeper into orientation, walks you through simulated processes of taking a call, writing a service order, and so on. The intent is to give you a beginning sense, by actual *doing*, of how these most basic processes work.

The third section is more expansive in that it provides a narrative sketch of the entire ServiceDesk system, describing the overall framework so, when you are later dealing with specific parts, you’ll understand how each fits into the larger picture.

The fourth is also a narrative sketch, but from a different perspective. Instead of providing a snapshot of what you might call the factory’s *architecture* or layout (which was essentially done in the third section) it outlines the *sequence* of how a typical day’s work should progress, from start to end, within that factory.

Both third and fourth sections are important to get your general grounding, and may be referred back to if later (when in the chapters that describe each feature in detail), you begin to lose your bearings.

The final section discusses some incidental matters, as you’ll see upon getting there.

You’ll notice that, though a great deal is described in this chapter, we don’t go into such detail as to give you anything approaching a thorough understanding. Don’t despair. This is only a general description, and keen understanding is not yet expected. That will come as you read in the detailed chapters—and with actual practice as you begin implementing.

Please believe also, there is major joy waiting as you cross the river and round the bend. You don’t have far to go. Just muddle through as best you can, and, believe it or not, you’ll be there soon.

A. Your First Look Around

Most new ServiceDesk clients are anxious to peruse features. Go ahead, explore! Don't worry, there's nothing you can seriously hurt.

You may be most eager to see your on-screen *DispatchMap*. Hit your keyboard's **F5** button, and there it is: your own custom DispatchMap, showing a basic sketch of your service territory.

It's not much to look at now because there's no data in it, but that's something we'll remedy soon. The red lines in your map represent the Interstates, the blue-and-red dot is your service location, and of course city names are painted in each of their respective locations. Typically, we paint jurisdictional boundaries in green and coastlines in blue. Try hitting your keyboard's **cursor keys** (aka "arrow" keys) to view different portions of your map (moving around in such manner is called "panning"). Press and hold the **spacebar** to see your map in *overview* mode. Press and hold the '**Z**' key to see locations of your zip codes. Notice that the day you are viewing and quantity of jobs are listed in the title bar at top (you can hit your keyboard's **PageDown** or **PageUp** keys to change the displayed day).

If you've panned around to view different areas in your DispatchMap, you'll find that by hitting your keyboard's **Home** key the system returns you back to the position where your office is near the center. Similarly, if you hit your keyboard's **End** key, the display moves to an area that lists each technician (at least, assuming you entered in some in the appropriate box of the *Settings* form). This area will show each job in list format under the technician to whom it's assigned.

When satisfied with this initial review of your DispatchMap, hit your keyboard's **Esc** button. This key is used pervasively in ServiceDesk. Any time you wish to leave a form or your present mode in a form (or if you want to just back out of something), try hitting **Esc**. It's almost always functional, and striking it is generally easier than clicking your mouse on an 'Exit' or 'Cancel' button (which some forms are also equipped with). Indeed, when any form has an Exit button, its function is usually the same as if you'd simply hit your keyboard's **Esc** key instead.

On exiting from the DispatchMap you're instantly returned to ServiceDesk's primary operating window, which consists of four *Callsheets*, displayed below the main menu. Notice that the Callsheets have fields for entering all the data that is pertinent to a service order: name, address, telephone numbers and even email for your paying customer, and separate such spaces for the service location in case it happens to be different (as it is, for example, with an OEM, Home-Warranty or other third-party payer job). There is a field for entering the kind of item that needs service, along with a similar space to enter the applicable brand.⁸ There is a space for entering an appointment date and time, and finally a very large space for describing the problem or request your customer wants addressed.⁹

Notice that on the menu bar above there are twelve command buttons. These are arrayed in three banks of four each. This quantity and arrangement is very deliberate. If you look at your keyboard, you'll see that (somewhat mirroring the buttons arrayed across ServiceDesk's menu bar), it has 12 function keys

⁸Don't worry about the fact there are no spaces for model, serial and similar. These are done via a separate *attachment*, that you'll read about later.

⁹You can add still more information via auxiliary forms that may be attached to each Callsheet. Specifically, you may add various kinds of notes (see page 62), and detailed information concerning the machine being worked on, such model and serial numbers, purchase date, selling dealer, etc. (see page 213).

arranged in a row just above the standard keys. In fact, on a typical Windows keyboard these twelve function keys are arranged in three banks of four each—just like those twelve buttons on ServiceDesk’s menu bar.

Why did we do it like this?

The answer is, we want you to be able to conveniently use your keyboard’s function keys for a wide variety of instant access, single-button operations. But, it’s hard to remember what twelve different keys (with just numbered labels on them) do. So, we figured we could: (a) give you each such function via on-screen you can click with your mouse; while (b) also arranging the buttons so they serve as *labels* for what the equivalent/same-position function keys on your keyboard—will also do—if stroked directly.

Wouldn’t you say that’s ingenious? We try to do *lots* of smart things, in ServiceDesk.

In fact, ServiceDesk has a plethora of functions you might want to get to with a quick keystroke. It has more than twelve. A lot more. But there are just twelve function keys. So what do we do? It’s simple. We make some of the function keys do double, even triple duty. In other words, a given function key (say, F5) will produce one result when simply stroked, but a different result if you hold the Shift key while stroking it, and a different result still if you simultaneously hold the Ctrl key.

If this sounds complicated, don’t worry; it actually works out as being quite simple in practice. And, we definitely DON’T EXPECT you to make any effort to memorize what function keys (and combinations thereof) are used for various tasks. In fact, you’ll find yourself *accidentally memorizing*, through usage, as time goes by. Certainly, accidental memorization is the very best kind.

In the interim, remember that the buttons on ServiceDesk’s menu bar serve as labels for equivalent-position function keys on your keyboard (there are many other assists as well, which we’ll review later). Please also note that, if you hold down your keyboard’s Shift button, or Ctrl button, or Alt button, the labels on ServiceDesk’s menu bar buttons change—to show what the equivalent-position keyboard Function key will do when used in combination with that shifter.

You’ll find most things in ServiceDesk can be done, in the alternative, via either mouse or keyboard action. In this manual (and when the method is optional) we tend to emphasize keyboard methods. The reason is that, ultimately, you’re going to be a more efficient user—of most any computer-based system—if you learn keyboard shortcuts in preference to mouse ones. Though it takes very little time and effort to remove fingers from the keyboard to use the mouse, it’s cumulative over time. At the least, each day’s work product is likely to be worth a few minutes more if you’re comparatively adept on the keyboard.

For initial learning, the mouse is often preferable (especially because in many instances you’ll not yet know the superior keyboard actions). But, as time goes by, please try to notice (each time you use the mouse) the system prompts you as to what the keyboard action could have been instead. Little by little (and again, even accidentally), you’ll find yourself remembering what the keyboard action could have been, and one time (as it pops involuntarily into your head) you’ll decide to use it. The next time you will again, and soon it will be second nature.

With this we conclude our guidance (such as it is), on your “first look around.” Before proceeding to the next section, feel free to try whatever meets your fancy. Again, there’s no real damage you can do at this point, so have fun.¹⁰

¹⁰ While in your initial familiarization stage, you’ll be creating a bunch of what is essentially “fictional” data. Do it with abandon. The more you feel free to try things and see what happens, the more you’ll be getting a grasp of how things work. Don’t worry about how

B. A Beginning Acquaintance Tour

To develop your understanding of ServiceDesk somewhat more systematically (particularly of its basic call-taking and dispatch functions), you're now invited to try a hands-on exercise where we'll pretend, essentially, that it's a typical day in your office, with you using ServiceDesk to respond to the ordinary and ongoing minute-by-minute needs of business. For this purpose, we must assume you are the secretary today, whether that's truly your job or not.

As you begin work in the morning, your first task, likely, is to deal with requests for service (or other attention) that came in overnight—in fact, any that came in *since* you closed the office yesterday.

If you use a live answering service to handle such requests (it seems few people do any more, but *if* you do),¹¹ you'll need to retrieve the messages your service has taken. For most people, this is an unpleasant task, but with ServiceDesk, you'll find it's no work at all. Simply press **Alt-R** on your keyboard (a mnemonic for Recieve messages), or use your mouse to select this feature from the *MainMenu*. In response, ServiceDesk will display the *Communications* Form, a device that allows your computer's modem to download messages directly from your answering service—right into each item's own awaiting Callsheet. Or, if you've used our *EmailedDispatchReceiver*,¹² you'll find your messages will already be in Callsheets when you arrive. Similarly, if you're using our *CyberOffice* feature, messages and service orders—as generated overnight via very nice interfaces on your website—will also already be in Callsheets waiting for you.

Regardless, you'll not be able to mimic the above pleasures just yet, because the particular foundations are not setup. Instead, try to imagine how delightful it *will* be, as you superbly manage all overnight-generated requests (some being actual service orders with appointments, others being various kinds of requests for attention), on the basis of Callsheets that automatically appeared for you within ServiceDesk.

Now, suppose you've pleasurably finished the above, and the phone starts ringing.

The first call is from a Mr. John Doe, who says he has a toilet plugged up. Go ahead: on the first line in your first ServiceDesk Callsheet, type "**DOE, JOHN**".¹³ After doing so, hit Enter on your keyboard (or Tab if you prefer) to move to the Callsheet's next line.

much of this pretend stuff accumulates. It will be very easy to *delete* it when you're ready to begin genuine operation. Specific instructions, in this regard, are provided in the last section of Chapter 12 ("Final Setup: Items to Check off") beginning at page 271.

¹¹ As a matter of business policy, we believe it's a false economy—at least in any *COD-oriented* service business—to leave your phone to be answered by a machine after hours. There is a significant percent of callers who, if they are privileged to speak immediately with a live person who is ready to book them with an appointment, is easily made into a profitable customer—and who, just as readily upon getting a machine will be forever lost to competitors. We don't think it's smart, if you've invested significantly in getting those new people to call you, to potentially squander the opportunity when they do. If you must use a machine, the one saving grace will be if the recorded message invites your potential customer to go on-line, to your website, and immediately book their appointment there (or communicate with you regarding any other issue). For newer generations, at least, this works very well, especially if implemented via our *CyberOffice* suite of functions (see our website for details).

¹² It's a separate utility from ServiceDesk, and is an alternative, even more automatic means of retrieving messages from a live answering service. It can also be used to auto-receive (and accurately parse into Callsheets) dispatches from home warranty companies that send dispatches via email (e.g., American Home Shield, Old Republic, etc.). Please call (or see our website) for details.

¹³ Notice that here it is very important to use the formal *last-name-comma-then-first* format. This is because we're going to be looking up customer's records, in various contexts, on the basis of the last name. The last-name-comma-then-first format gives ServiceDesk the basis to deduce what's last name and what is not. You might notice we're also entering in all upper-case. The reason is economy. Unlike in the *Settings* form (where we enter names just once), here we'll be entering names (and other text) many times each day. For that much typing, the convenience of not having to worry about upper versus lower case is a significant savings.

Now, suppose Mr. Doe tells you his address is "132 Sunset Drive" (or some other street name that happens to exist in your area). Go ahead, begin typing that in the second line of your ServiceDesk Callsheet, but in this instance please watch as you're typing in the word "SUNSET".

Please notice that when you've typed just a few characters, a drop-down list appears—which displays streets, from your actual territory, that match whatever character string you've already typed. Please also notice that abbreviations for applicable city names also appear, along with zip codes. If there are multiple streets of the particular name involved (each displaying its own city), naturally, you'll ask Mr. Doe which city he's in. When the appropriate street is identified, press your cursor keys (aka *arrow* keys) to move into the list and highlight it, then press **Enter**. Alternatively, you could **click** on the appropriate listing with your mouse. Go ahead. Try it.

As the street is selected, you'll see that its full name instantly inserts to the appropriate section of your Callsheet, along with its map-grid reference and corresponding city name. In fact, your cursor is simultaneously moved for you, past all this, into the first telephone number box, because that's the next item of information you're going to take from Mr. Doe. Suppose he now gives it to you, and you type it in. Go ahead, type in an imaginary telephone number, or perhaps use your own.

Now, using your keyboard's Enter key, cursor keys, tab keys, or the mouse (whichever is easiest), move into the Callsheet's "Item Type" box, and type "**TOILET**". Then hit Tab or Enter to move to the "Item Make" box, and type "**ELJER**" (assuming this is the make Mr. Doe has provided).

Now, move into the "Description/Request" box. Here you can type "**PLUGGED UP, OVERFLOWING**" (or anything similar that pleases you).

Once this information has been obtained from Mr. Doe (in real life it might require 30 seconds), the next question is: when will you schedule the work for. For considering this, you'll want to know where Mr. Doe's location fits in comparison to other jobs that are already scheduled. This interest poises us for some major magic.

Using your mouse, do a **right-click**¹⁴ anywhere on Mr. Doe's address line. Now you'll see a whole new display, called the ServiceDesk *DispatchMap*. This is that broad/overview sketch of your territory—and, its most important attribute at present is, it shows Mr. Doe's actual, physical location, prominently circled, in bright-red. And, if you were already setup and genuinely running in ServiceDesk, it would also show little reference flags at the locations of all other jobs as scheduled for today, along with route-lines for each, color-coded for the techs assigned (for now, please try to *imagine* it).

The point is, this display makes it into *child's play* to deduce a date and time-frame in which Mr. Doe can be fit into your schedule—importantly, with maximum convenience to both him and your own routing and scheduling needs. As noted, the DispatchMap first shows you the *current day's* schedule (look in the caption bar at top). Thus, you can immediately see if scheduling today is practical. If not, hit *PageDown* on your keyboard, which changes the display to tomorrow's schedule (again, look in the caption bar at top), and so on, with *PageUp* bringing you back, etc.

In this case, suppose that as you look at other jobs scheduled for today (again, we're *pretending* in this regard), you see that your technician Jim will already be passing close to Mr. Doe's location, en route

¹⁴An alternative is to **double-click**. In most any ServiceDesk context that suggests the right-click, double-click is an alternative. In some system the right-click does not always work smoothly. If either that's the case, or you find double-clicking more convenient, by all means use the latter method (for further discussion, see page 328).

between two jobs in the early afternoon, and Jim still has a bit of spare capacity. Given this, you offer Mr. Doe a time-frame for today between 1:00 and 4:00, which he accepts.

Now, how do we enter into ServiceDesk the fact the Mr. Doe has accepted this appointment?

Making sure you actually have your map displayed to the day for which you're scheduling Mr. Doe (in genuine operation, this is instinctive), do a simple mouse **click** on his red-circle location reference. Immediately you'll see a drop-down list with time-frames. **Click** on the time-frame wanted (in this case, "1-4").

Now, you'll see that you're instantly transported back to the Callsheet, where text denoting the appointment has been inserted for you.

At this point, you may *think* you've done all that's needed to create the job and its corresponding appointment. In fact, so far as ServiceDesk IQ is concerned, all you've done is create meaningless text. To prove this, hit **F5** and look in your DispatchMap. Do you see the appointment? No, you do not. ServiceDesk gave you a handy means of creating text that describes the appointment, but to ServiceDesk, that's all it is at this point (i.e., text in the Callsheet). In fact, other text in your Callsheet is also *non-operative* so far as being seen as meaningful job information by ServiceDesk.

To make it meaningful, you have to do one more, simple step.

Notice, over on the right side of the Callsheet, there's a column of five option buttons under the heading "Status". Click on the button labeled "Job/Sale" (or press **Alt-J** as the label suggests when you hold your mouse button down on it). Upon doing so, you're presented with a little yellow form, labeled *Create Job/Sale*.

You'll use this form to specify details about the job and appointment you're creating. Specifically, this form proposes an invoice number and date, and invites you to specify the technician you intend to dispatch. For now, don't worry about those details. Just hit your keyboard's **Enter** key to accept what's proposed (of if preferred, you could click on the form's "OK" button).

At this point ServiceDesk will do several meaningful things.

First, you'll notice it prints the ticket (aka physical invoice, work-order, etc.).¹⁵ Also, the Callsheet goes dim. This tells you *its* work is done. You may also notice that in those 'Status' options on its right, "Job/Sale" is selected. This tells you this Callsheet has completed its purpose via the act of creating a job (another away of stating this is, the Callsheet was the *conduit via which* a job was created). There's also documentation, in the lower right corner, indicating when the Callsheet was created, and by whom.

Next, press **F5** to look again at your DispatchMap. Now you'll see a little reference for Mr. Doe's appointment, nicely displayed in its correct location, as part of today's schedule. And, if you look in the *list* area of the map (you can use your cursor keys to pan over to it, or hit your keyboard's **End** button to go there instantly), you'll see he's listed under "Unassigned," since you haven't yet assigned the appointment to any particular tech.

¹⁵This initial invoice is one that's been pre-configured for you, to let you immediately begin printing tickets without any need for advance setup. It is not a form you're tied to. You can use any form or format wanted. ServiceDesk will do your complete bidding in this regard (see discussion that begins on page 291). Particularly and especially, you'll likely find it much more efficient, ultimately, to use tickets where the form *image* is pre-printed for you by a local printing shop, leaving to ServiceDesk the sole task of printing text into spaces on such pre-printed forms (i.e., as opposed to printing both text and the form image, as it's preconfigured to do). Even better (and when you move to truly advanced stages), you'll likely find it most efficient to refrain from printing service tickets at the office (at least for *field* work), at all, instead relying on more efficient electronic dispatching via SD-Mobile.

Also if you hit **F6** on your keyboard, you'll see the appointment displayed in your *ScheduleList* form. This is the actual list that keeps track of each appointment (your DispatchMap reads from this very list to *graphically* display your appointments). Quite simply, when you did that Job/Sale process, ServiceDesk read from the otherwise inoperative appointment text in the Callsheet, and on its basis created this actual (and *operative*) appointment entry in your ScheduleList. As an entry here, that appointment is real, and ServiceDesk recognizes it as such.

Even more important than the above, remember we said the Callsheet, it's task now done, had been "the conduit via which a job was created." Now we'll point out what that really means.

If you press **F7** on your keyboard, you'll see what's called the *Jobs-Current* form. Here you'll see what is really the most important consequence of what we commenced when clicking on the *Job/Sale* button, in that originating Callsheet. Quite simply, we created a *JobRecord*, which is precisely what you're looking at here in the F7 form.

Please don't be confused by the fact that a portion of this document looks very much like a Callsheet. Don't be confused by the fact that, within that portion, it contains precisely the same text as was pulled from the Callsheet. It may look similar, but it's an entirely different animal.

You should understand, it is this JobRecord that *represents* the job. It is this JobRecord from which the job will now be managed. It is this JobRecord that will maintain a running, historical narrative, detailing everything that happens on the job (notice in its *History* section to the right, there is already an entry detailing its own creation).

By contrast, the Callsheet (in which you initially typed the job-creating information), has (again) gone dim, because it was nothing but an entry point—in which we typed the initiating information, in order to create the JobRecord that you're now looking at. (In fact, having done its duty, that Callsheet will be moved to an archive with the next housekeeping event.)

Assuming you understand the above (it's an important concept that some people miss),¹⁶ hit **Esc** on your keyboard to return back to the main interface.

Now create several Callsheets involving imaginary requests for service, much as we did the first, but use different names, addresses and descriptions for each.

¹⁶Try as we might, we can't seem to end confusion on this point with everyone. At least yearly, we find some new user has been treating Callsheets as though they are JobRecords, while ignoring the actual entity that's intended for the purpose. To make certain you are fully clear on the subject, imagine an old-fashioned paper and pen managed office.

Without doubt, there is some kind of pad, kept on the call-taker's desk, on which he immediately begins scribbling notes when taking an incoming call. ServiceDesk Callsheets are a direct equivalent of that pad. Now, some incoming calls involve an actual order for service. For those that do, the call-taker waits until there's a moment of respite between calls, then reaches to where there's stack of formal service tickets. He pulls one, reads details from his scribble pad pertaining to an order just taken, and writes the same neatly onto his formal ticket. ServiceDesk *JobRecords* (again, accessed via F7) are the direct equivalent of that formal ticket— and the *Job/Sale* sequence in ServiceDesk (as initiated *from* a Callsheet), is, likewise, the direct equivalent of the paper-and-pen transition from scribble notes on the desk pad to neatly writing onto the formal ticket.

Please note, also, a further parallel.

Once the old-fashioned call-taker finishes writing all the pertinent details onto a formal ticket, it's that ticket that is henceforth used to manage the job (same as with JobRecords in the ServiceDesk F7 form). The old scribble notes on his desktop pad, by contrast, likely get big "X" drawn through them, and are never used again—same as should be the case for any ServiceDesk Callsheet that's served as the instrument via which a JobRecord was created.

Again (and to emphasize) the Callsheet was just a scribble pad. The mistake some new users make is to mistakenly think it is the JobRecord instead. Please do not make that mistake. Manage your jobs from *JobRecords* in the F7 form. Let Callsheets that have completed their task go their way.

Please bear in mind you can move between Callsheets using your mouse, and to a new page of blank Callsheets (once you've used up all four on a given page) by hitting **PageDown** on your keyboard. **PageUp** brings you back to earlier pages. There are also several handy keyboard tricks for moving between Callsheets, but we'll defer discussion of those until Chapter 5.

For each of these Callsheets that involve pretend orders for service, click on *Job/Sale* to do the package of processes that includes printing the ticket, creating the JobRecord, and entering the appointment to the ScheduleList.

In consequence, you'll end up with several appointments being viewable in your DispatchMap (F5) and ScheduleList (F6). You'll likewise have several JobRecords viewable in your JobsCurrent form (F7). Notice that in the JobsCurrent form only a single record is displayed at once. You can use your keyboard's **PageUp** and **PageDown** keys to peruse between each record there, do searches, etc.

Among the pretend jobs you're creating, suppose one is for a landlord, requesting service at a tenant's residence. In this case, be certain to place the landlord's name and address in the top section, and the tenant's in the second. In ServiceDesk, we consider the paying party our true *customer*, and it's important (for a variety of reasons) that you always have the *paying party* in that top name and address section. The second section needn't be used if the paying party and location are the same—but if they're different, obviously, use that second section for location-party info.

You might also pretend you've gotten some calls that don't involve service orders. Maybe a customer called to speak with your boss regarding a matter you cannot assist with, for example, and your boss is presently unavailable (remember, even if it's not so, we're assuming you're the secretary).

Move to a new Callsheet, type in a fictitious name, telephone number, and description of a pretended request. Now, look in the Callsheet's upper-right corner, in the area titled "*Active Desk*."

Depending on what you've done so far in the *Settings* form (in particular, in its *List of Station Names*), you may see *one button* in this section, or *two*.¹⁷ Assuming there are two buttons, you can click on the second to transfer present ownership of the Callsheet from your desk to someone else's. Please go ahead and do so. Hopefully, you've already set it up so that your boss is listed as another party. If so, transfer ownership to him, by clicking on his listing. If you have not (or if you actually are the boss pretending to be secretary), click on anyone else's name, pretending it *is* your boss.

In response, you'll see the Callsheet on your desk *goes dim*. This signifies that (for the time-being, at least), it's no longer your responsibility. Instead, it's now owned by (and is the responsibility of) the person to whom you transferred it. This means, among other things, it has now "*lit up*" on that other person's desk.

Again, we're supposing that other person is your boss, and he was not immediately available to take the call. However, since you've transferred ownership of this Callsheet to his desk, it's akin to having taken a physical note (saying "Please call this customer"), and placing it on his desk. Much as he'd see that physical note, he sees this new Callsheet lit up on his screen—and realizes he needs to call the customer.

Now let's make a transition, and—instead of pretending you're the secretary—we'll pretend you're the boss to whom ownership of that Callsheet has just been transferred. Thus, regardless of whether you were

¹⁷ If you only see one, go to the Settings form (**Ctrl-F1**, or select from the list provided under *File Functions* in the MainMenu). Once there, add one or more additional names to your *List Of Station Names* (purple section upper-left). Type a name, first and last, then hit **Enter** on your keyboard to insert to the list. Once done with your insertions, click on the form's '*Save Net-Wide Values*' button).

away from your desk and just stepped back to it, or were sitting there all along, you see that new Callsheet “lit up” on your desk, and realize—since it’s lit up in such manner—it’s presently your Callsheet, and your responsibility to deal with it.

So you quickly peruse its contents, grasp what it’s about, and determine you need to call the customer. At this point, obviously, you could pickup your phone and dial the number that secretary previously typed into the Callsheet. But dialing is hard. It takes ever so much effort. Instead, a simple **right-click** on the telephone number will cause ServiceDesk to *dial for you*.¹⁸

It may not at first *seem* like a huge deal to have the computer dial for you, but once you implement this feature and get used to it, you’ll find that, very quickly, you no longer want to live without it again.

Suppose now that, upon dialing the customer, you get a recorded voice greeting. Per invitation, you leave a message, explaining your attempt to follow through. Having done so, you’ll likely want to internally document the fact that you just did your duty. How do you easily do that?

Very simply, do a simple **click** on the Callsheet’s ‘*MoreInfo*’ button (or, as you’ll see it suggests if you hold down your mouse button *while* clicking, you may instead do an **Alt-M**—where the ‘M’ stands for ‘*More Info*’—from your keyboard). In response, ServiceDesk will open the Callsheet’s connected *MoreInfo* form, which is supplemental to the main form, and engineered to allow you to attach added elements of relevant information.

In this case, you’ll see it has auto-inserted a time-and-date-stamp, for you, with your initials, making it ready for you to add your own note, perfectly so annotated. So, just type some simple text that’s relevant to the situation. You could type it right out, but for this kind of situation I like to use a simple abbreviation. Specifically, for this precise situation, I’d type “**LMOR**”. It happens to be my abbreviation for “*left message on recorder*.” I do many variations, such as “**LMWC**”, for “*left message with child*,” or “**LMWF**”, for “*left message with female*,” etc. You, obviously, may do whatever you wish.

The important thing is, you’ve used this very easy method to fully document your own due diligence—and if a question ever comes up as to whether you were, in fact, so faithful, the memorialization of it is right there.

At any rate, having done this documentation, you can **Esc** out of the *MoreInfo* form, and back into the body of your Callsheet. But there’s a remaining issue. There it is, still “lit up” on your desk, staring at you, as though you still have some present duty in its regard. But you *don’t have* any present duty. You just fulfilled your present duty—so that Callsheet shouldn’t “ought-a” be asserting itself for attention.

How to fix it?

Very simply, look to the right edge of Callsheet, this time in the section (titled ‘*Status*’) that contains five option buttons. Among those five, look for the button labeled ‘*Hibernate*’. **Click** on that button (or, again as the label suggests if you hold down your mouse button *while* clicking, do an **Alt-H** on your keyboard).

In result, ServiceDesk will present another Callsheet-related form, this one (somewhat obviously) called the *Hibernate* form. Its purpose, as the name suggests, is to put a Callsheet “to sleep” for some period of time.

¹⁸At least, it will, assuming you’ve done the necessary advance setup, as needed for this function (see ____). Likely you’ve not yet done that, so just pretend for now that you’ve enjoyed the feature (and look forward to actual enjoyment later), then move on.

In this case, let's suppose it's already in the afternoon, and since you just left a message on this customer's recorder, you figure you really have no duty to reinitiate the return-call effort until tomorrow morning. From the Hibernate form, select a "sleep" period of **1 day**, then **click** on *Okay* (or hit your keyboard's '**Enter**' key).

In result, you'll now see your Callsheet goes dim—appropriately signifying that, in fact, for the moment you don't need to worry in the least about its contents. Your duty for now is done. Make all your Callsheets dim, and you can truly celebrate.

Now assume you've gone home, patting yourself on the back for having done a good job, taking care of all tasks that were presented to you via Callsheets. You've have a good night's rest, and are not back in the office on the following morning. Lo and behold, that Callsheet from yesterday is not "lit up" (it "awoke" from its sleep), and in such state is "telling" you that you ought to try calling the customer again.

So, again do a **right-click** to auto-dial (or *pretend* to do the same, if you've not yet setup that function), *and* pretend this time the customer answers. You discuss the matter on which you were supposed to call. The customer is happy; you're happy, and you both hang up. Now, you've accomplished the task for which the Callsheet was created. Its task is done. However, unlike the other Callsheets that we've had you create as part of this exercise—which completed their tasks by being the vehicle via which a JobRecord was created—this one did not prove to be such a conduit, or to have that kind of purpose. But you're done with it, nonetheless.

What to do?

You don't want to do the *Job/Sale* sequence, because that would create a JobRecord. Instead, you want to indicate this Callsheet was completed via other means (specifically, you talked with the customer regarding his or her inquiry). So, and to emphasize, you're "done" with this Callsheet, but via means other than doing a Job/Sale transition on its basis.

Given this, please look again along the Callsheet's right edge, where there's a group of five option buttons under the heading '*Status*'. Look for the very last, labeled '*othrws Done*'. Hopefully the meaning is obvious, but if not, "othrws" is an abbreviation for "otherwise," and the notion is you pick this option to indicate you've completed the Callsheet's task via means "other than" doing a Job/Sale transition.

Upon so picking that option, you'll see the Callsheet goes dim, meaning you've got no present task in its regard. In fact, with the next housekeeping process, it will be moved to an archive where, unless an odd should arise, you're likely never to see it again.

We titled this section "A Beginning Acquaintance Tour." We should have emphasized the word "beginning," for, in fact, we've only shed just a bit of light on the most beginning of functions. We've not even flirted with the multifarious aspects involved in dispatch processes, job management, inventory control, accounts receivable management, and so on. For the sake of brevity, we'll leave those discussions for later.

C. Thumbnail of the Structure

As noted in the chapter's introduction, the purpose in *this* section is to give you a basic conceptual overview of the entire ServiceDesk system. You must understand a building's overall architecture if you are to comprehend, when dealing with a specific part (which we'll do in later chapters), how it fits in with the whole.

You may most easily think of ServiceDesk as being divided into five broad categories of function: Call Management, Schedule/Dispatch Management, Job Performance Management, Post-Completion Management and Ancillary Functions. The lines between each category are not always distinct, sometimes overlap, and a few processes may crisscross between and betwixt. Regardless, if you'll keep the broad categories in mind it will help to conceptually organize the various features we'll be discussing, both in this overview and in the detailed chapters that form the bulk of this manual.

In that regard, you'll notice the major chapters of this book mirror the small headings of the five sections that follow. That's because these sections are a brief overview, organized in a parallel fashion, for the detailed discussions of those main chapters.

i. Call Management

If your business was a shoe store, most of your sales would probably originate from what's called "walk-in traffic." And once you sold a pair of shoes, that would likely be the end of it in terms of your performance to earn the sale. Thus the telephone would not be a particularly important instrument for you, at least in terms of generating sales. For a service call performing business, obviously, the situation is radically different. You may not even have a storefront, and regardless the telephone is your lifeline. Virtually every job will be originated via a call received, and further telephone calls may be important steps on the way to job-completion. Further calls still (regarding complaints, callbacks, etc.) may be important in maintaining customer satisfaction. Obviously, it's very important to your business's success that incoming calls be handled with a maximum of efficiency and effectiveness.

ServiceDesk offers the most comprehensive and easy system imaginable for this purpose. At its very heart are the Callsheets. Four to a page (filling the primary screen in ServiceDesk), these are the primary tool for taking information on a call, managing it, responding to it, recording responses, and converting a service request (i.e., one that occurs during a call) into an actual job.

As is needed for the purpose, Callsheets have spaces for virtually any item of information you may need to enter in connection with an incoming call. And they are equipped with many tools to help you fill-in spaces. There's a *CustomerDbase* system, for example, that (among other things) allows you to instantly insert a customer's information set. There's the *QuickEntry* system that allows you to instantly insert a frequent client's info. There's the *StreetFind* system that allows you to lookup and insert a first-time caller's street name, grid reference, city and zip. There's a *UnitInfo* system that allows list-based insertion in the Callsheet's Type and Make boxes, and attachment of a data set applicable to a particular machine. And there's an *Appointment-Creation* system that fills-in the Callsheet's Appointment box with whatever day and time you've indicated from a DatePicker type of calendar, or from within the DispatchMap. With the aid of all these features, it should typically take less than a minute to enter all information relevant to an incoming call.

In addition the Callsheets have features to assist in *acquiring* information. Besides facilitating insertion of past-customer information, for example, the CustomerDbase system facilitates lookup, from any Callsheet,

of past and pending jobs for the same customer. And once an address is entered, the *Item-Locate* feature instantly shows precisely where the customer's location fits graphically, and in comparison to other jobs.

In a third category of process, the Callsheets have many feature to assist in *handling* a call. If a secretary has taken the call, for example, and needs to transfer it to the boss, it's a simple matter to switch it to his (or her) computer station. If you've returned someone's call and want to document having done so (or make any other extraneous notes), there's an attached *MoreInfo* box in which to do so. If you've discharged your present duty on a Callsheet and don't want it lit up on your desk, you can use the *Hibernate* feature to have it sleep. And if you've neglecting to do anything in terms of servicing a Callsheet for an excessive period of time, there's a *Callsheet Alarm* system to alert you.

Finally, in terms of process (and perhaps most importantly), Callsheets are the *conduit* for creating every new job. Basically, if the incoming call involves something more than fielding a complaint, responding to a request for information, taking a message for the boss, or something similar—if in fact the caller wants to order service—then obviously it's all the information pertinent to a service order that you'll be inputting to a Callsheet. Then it's from that Callsheet that you'll invoke the actual job-creation process.

For all these reasons, let it be understood that as standard and habitual practice, the moment you pickup an incoming call your fingers should be poised on the keyboard, ready to begin typing into a new Callsheet as the customer begins speaking. In this manner every call of significance is instantly logged, and you'll have the beginning basis in every instance for responding further as needed. Do not use pen and paper. Throw it away! The information belongs on-screen where it can be more ably and perfectly managed.

As a final matter, you should know that once a Callsheet has completed its task, it is then ready to be retired out of the *current* Callsheet space into an *Archive* where, if ever wanted, it may later be reviewed. This removal does not occur, however, until you run a periodic housekeeping process (preferably a few times each day). Until then all completed Callsheets will remain dimmed (but not moved out of) your current Callsheet workspace.

ii. Job Management

As explained, with every significant telephone call, you'll be creating a new and separate Callsheet to document and handle the situation. This creates one set of records: the Callsheets, which deal more or less directly (and usually, at least) with work done *while on* the telephone.

Now let's distinguish a very different kind of record. What happens when, rather than simply taking a message with a Callsheet, you're taking an actual service order? Does that Callsheet, with all the ordering information thus entered, represent the job? Emphatically (and to review). In this context it simply represents the details of a particular service request.

For any job to even exist as such in ServiceDesk, there must be a '*JobRecord*' describing it. In fact, each and every job must be so represented; otherwise, there is no basis by which ServiceDesk comprehends it as a job. We explain this strongly in the hope of steering you away from what is otherwise a common misconception among new users.

To simplify, think of how a primitive office may have formerly used an old-fashioned, paper-only system. Usually there was a "Call-book" kept on the desk for jotting down information from incoming callers. When the call involved a service order, the information was first written into this Call-book, just as for any other matter. But then, after the caller hung up, it was typically re-written (or perhaps manually typed) onto some

kind of service-ticket or job-order form, which was then used as the basis for the job's management (i.e., the technician took it to the job with him, wrote charges on it, left a copy with the customer, etc), even while the original call record remained in the book as exactly (and only) that. It is similar in ServiceDesk. Callsheets are nothing more than those same kind of entries. While they may *describe* a job order, for a job to fully exist as such, another document must be created.

In the early days of ServiceDesk, this additional document was known as a “*Work-In-Progress*” record (or “WIP” for short). Since many users found the acronym confusing, however, we have largely abandoned it—in favor of the more obvious term of just “Job” or “JobRecord.”¹⁹

And regardless, please also understand that just as Callsheets reside within your computer, so does each JobRecord. There is sometimes confusion between the two records, for a couple of reasons. First, Callsheets are the *conduit* via which each JobRecord is created (i.e., job-creating information is always placed first into a Callsheet, and as entered there is the basis for importation of the same information into a newly-created JobRecord). Second, a portion of each JobRecord *appears very similar* to a Callsheet. This is in consequence of the fact that Callsheets are designed to, among other things, collect the information that needs to go into a JobRecord, so they logically have spaces for much the same information, and for clarity we even arrange the spaces similarly on each form. Even so, they are completely different records, and we'll ask you to please avoid letting the similarity in appearance fool you.

As mentioned, every job (or even an over-the-counter parts sale if it's only that) is *initiated* from a Callsheet. The process is invoked (once all the job-creating information has been placed into a Callsheet), by then clicking on its ‘*Job/Sale*’ button. In response, the system brings up the ‘*Create Job/Sale*’ form, which manages actual job creation. When you consent it performs several tasks, the most important of which is to create the actual JobRecord. In addition (and unless set otherwise), it will print the ticket, and will an appointment into the ScheduleList (assuming an appointment was placed in the Callsheet's Appointment box).

Also as part of this process, ServiceDesk automatically endows each new JobRecord with an *InvoiceNumber*. This is essentially an ID for the job. Each job must have its own, unique such number, and that number will never change. Forever and always, a job will be known by the number it initially receives as part of this process.

There are two different forms for viewing (and doing many kinds of work within) JobRecords: the ‘*JobsCurrent*’ and ‘*JobsArchived*’ forms (press **F7** or **Ctrl-F7**, respectively, to view). As the names imply, the first form is for work with jobs that are still pending; the second for those that have been completed and moved into what you might think of as a “jobs-completed bank.” Regardless, you'll notice (again) that each form includes a portion that looks *very similar* to a Callsheet, for it echoes the job-initiating information that was carried from one. In addition, however, there's a large textual area in which ServiceDesk maintains a narrative history of everything that happened on the job. Most entries in the JobHistory are made for you as various tasks are done in connection with the job (such as dispatching it, for example, ordering parts in its connection, etc.). But you may also make your own entries as wanted.

In the broadest sense, part of job management is scheduling and dispatch. These topics loom so large, however, they are separately categorized and discussed in the next section. We mention this for context, for after a job has been dispatched and the tech's been there, the next task is to report on what happened. There is a well-developed process in ServiceDesk for doing this. We do this via what's called a

¹⁹The acronym survives in reference to the overall system of job control, which is still (at least in some contexts) called the WorkInProgress, or WIP system. This is why the sentry system, that's used to make sure pending jobs are adequately serviced, produces “WipAlerts.”

PostVisitReport. Basically, there's a form for making these reports (called the *PosVisitReport* form), which walks you through a dialog asking what time the tech started, what time he finished, what he found or did, what parts he used from stock, what parts he needs to special order, what funds he collected, and so on. It's important for this report to be made after every technician's visit, for it's by this means the system keeps track of all these kinds of information, and helps you manage them.

In particular, if the technician has used any parts from stock, information regarding this is collected during the *PostVisitReport*, then it's forwarded to the *InventoryControl* system, which keeps track of all your normal stocking parts. At all times this system (which consists of various records and forms) can and should know how many of each item you have in your storeroom and on each truck. It facilitates reordering/restock of parts, and on the basis of the *PostVisitReport* knows what's been used off a truck, and therefore what needs restocked to it.

Additionally, if it's disclosed during a *PostVisitReport* that non-stock parts are needed, that information is forwarded by *ServiceDesk* into the *PartsProcess* system. Here *ServiceDesk* helps you manage all your inquiries and ordering of parts that you do not normally stock. It's another system that involves multiple forms and records.

Also of great significance, if it's indicated during a *PostVisitReport* that the technician has collected items of money, this information is forwarded by *ServiceDesk* to the *FundsControl* system, which keeps track of every such item of money, assures that enough is collected to satisfy each claimed sale, facilitates and assures accurate deposit, and so on.

Closely connected is our *Virtual Terminal*. It allows you to run credit card transactions directly within *ServiceDesk*, without any separate machinery or software, and with each element in the process perfectly integrated with what you're otherwise doing. It means you can throw away whatever other "terminal" system you were formerly using. Also, if you do in-person transactions (i.e., with customer and her card present), you can purchase a simple swiping device (and attach it to your computer) for as little as \$33.

As a final major process that's based on the *PostVisitReport*, the system makes entries into the job's narrative history regarding all items reported. This makes it possible for anyone in the office to instantly see such pertinent facts simply by bringing up the relevant *JobRecord*.

As you can see, the *PostVisitReport* is a major factor in managing a job's performance. As stated, it's important to see that it's always done. You may in this regard either have an office person do it on behalf of the technicians or have them do it themselves. Regardless, there are systems to help you assure it's done in each instance, and much as there's a system of *Callsheet Alarms* to alert if one of those is being neglected, there also is a *WipAlert* system designed to alert if a job is neglected.

Also as something of a parallel with *Callsheets*, when a job is finally completed, its *JobRecord* is no longer pending, and so needs to be moved out of the *JobsCurrent* file. Thus, much as there's a process for removing old *Callsheets* from current work space to an archive, there's similarly an "archiving" process for *JobRecords*. And once completed jobs are thus moved, they may then be viewed via the *JobsArchived* form.

iii. Schedule and Dispatch Management

As mentioned, when you create a job (by invoking the *Job/Sale* process from a *Callsheet*), *ServiceDesk* simultaneously creates a beginning *JobRecord* for it. Additionally, if the *Callsheet* indicates

there's presently an appointment (usually the case), ServiceDesk inserts an entry describing the appointment into a file called the *ScheduleList*.

The *ScheduleList* is exactly what the name implies: a listing of every appointment that is on your roster at any point in time. It's the primary underlying data, in other words, of the Schedule and Dispatch system.

Basically, there are two different venues in which you can use and manipulate data from the *ScheduleList*.

First is the *ScheduleList* form (accessed by pressing **F6**). This form shows each *ScheduleList* entry as one, simple line of text. From this context you can edit for corrections, add new entries if wanted, delete an existing entry (in the event an appointment cancels, for example), sort for correct sequence of date and time, and similar tasks. It's the utility we use, simply, when wanting to manage the data in a purely *textual* context.

Our second venue for managing the *ScheduleList* is the *DispatchMap* (accessed by pressing **F5**). Though this form seems radically different, it displays exactly the same data—from the very same underlying file—as does the *ScheduleList* form. The difference is that rather than showing the data textually in its raw format, it interprets the data *graphically* to show the temporal (i.e., time), spatial, and Tech-assigned relationships of each job. But again, always bear in mind it's showing the same data—from the same *ScheduleList* file—as does the *ScheduleList* form; it's just showing it in a very different manner.

While the *ScheduleList* form is ideal for manipulating data in a manner that involves direct access to text, the *DispatchMap* is much better suited for the kind of manipulation that's concerned with temporal, spatial and Tech-assigned relationships. So, logically, it's the venue where we do things like setting up each technician's route by deciding which job to assign, checking off the fact that we've actually given any particular job to the assigned tech, changing a pre-existing tech-assignment, looking for a particular tech's route to see where he might be at a given time, and so on.

Of course, manipulating already-scheduled appointments is irrelevant—unless you've already created the appointments, and entered them into your *ScheduleList*, in the first place.

There are, in fact, two contexts in which you may be scheduling a customer: when they're *initially* placing an order for service; and when they making *subsequent* appointments, such as after you've ordered in some parts. In the first context you'll be inputting their order information to a Callsheet, naturally, and the appointment reference goes there too. In the second context the Callsheet will have long since been retired (and be irrelevant now regardless). At this point the job is being managed from its JobRecord, so it's from there (using the JobsCurrent form) that the scheduling process is initiated. Regardless, the system has tools in both contexts to make it very easy, including an *Item-Locate* feature that shows you the customer's location on your *DispatchMap* (and in comparison to everything else that's scheduled), a DatePicker calendar, and so on. Just remember an appointment entry must ultimately be inserted to the *ScheduleList* (whether during job-creation from a Callsheet or via the 'Scheduling' utility in the JobsCurrent form), if ServiceDesk is to comprehend and treat it as such.

iv. Post-Completion Management

When your technician completes a job, it may be the end of work for him, but, obviously, some of the office's work has just begun.

For any job that involves warranty work, for example, you need to make a warranty claim. In general, this is done via the *FinishedForms* context, accessed (among other means) by pressing **Alt-F4**. Here ServiceDesk will fill-in for you (at least mostly, if not entirely) any of several different complete forms. We say “complete” because unlike the up-front ticket (that ServiceDesk prints, typically, with creation of the JobRecord), these forms have places for a description of work done, materials used, and what the charges are for everything. In particular regard to warranty claims, ServiceDesk will fill-in an on-screen NARDA for you, give you the chance to review and edit, then (when your ready) it will electronically transmit the claim to your processing entity (or save to a file for your later upload).

In every instance, regardless, your sale must be recorded to some kind of sales accounting system. Plus, if it's a billed job, you must keep track of the amount owed long enough to assure payment, send out reminders if payment is not timely received, and properly check-in payment once it is received. And you must compile periodic reports showing what your sales have been, what you've collected in sales tax, and so on (as necessary for tax purposes, at least). Naturally, ServiceDesk has systems for all these functions.

Somewhat more specifically, for each and every JobRecord that's created, the system expects (indeed, demands) that you eventually report a completed sale in its connection (even if for zero amount). The utility for this purpose is called the '*SalesEnter*' form (accessed by pressing **F9**). When you enter a completed sale there, you're required to indicate the applicable InvoiceNumber, customer's name, and sale amount in each of several categories. In response, the system makes an entry with this information in your *SalesJournal*, a file that will hold a record for every sale you make. It also checks-off the JobRecord (as having been so recorded) so it's then ready for movement to the JobsArchived file.

There's a form for reviewing and editing your SalesJournal. It's called the '*SalesView*' form (press **SHIFT-F3**). There's another form for making periodic reports of several kinds. Most important, in this connection, it will compile a report about your sales. It's called the '*Reports*' form (press **F11**).

When entering completed sales, the system checks in your FundsJournal to see if enough has been collected to satisfy the sale. If not, it attaches a code to the Sales entry indicating it was a billed rather than paid-at-the-time sale. Plus it creates an *AccountsReceivable* record for the job. These records are kept, logically, in an AccountsReceivable file, and may be viewed, edited and otherwise manipulated via the *AccountsReceivable* form (press **F3**).

When payments are received on any Account Receivable, you need to enter the fact via a utility that's provided in your Funds form (press **Ctrl-F9**). This creates an appropriate entry for the item in the FundsJournal, and records the amount to the applicable AccountsReceivable record. When sufficient payment has been received, the A/R is retired.

To deal with customer's who are late in payment, there's what's called the 'Dunning' form (press **Ctrl-F3**). You can use it to print statements, reminder notes, and (if preferred) create dunning letters.

v. Other and Ancillary Features as Packaged with the Core System

Some functions are so interwoven, between and betwixt those already discussed, it's difficult to categorize them under just one heading. Others are more supplemental in nature, standing apart as auxiliaries to the primary facets of operation. Regardless, these are matters we'll touch on here.

The first such matter concerns the *CustomerDbase* system. Briefly touched on when discussing Call Management, it's really far broader than that. Basically (and unlike most systems), ServiceDesk does not

maintain any master file with a unique listing describing each customer. Instead the history of all your past jobs provides equivalent information, without the burden of having to keep a master list accurate and up-to-date. Your past jobs are, of course, recorded in your *JobsArchived* file, and this is the underlying basis (along with, partially, your *JobsCurrent* file) for the *CustomerDbase* system. It works specifically on the basis of *indexes* that are compiled from these files. These indexes sort for customer names, addresses and telephone numbers. Whenever you are doing a *CustomerDbase* search (such as when inputting information to a *Callsheet* for example, or from the **F12**-based context), the system searches instantly in these indexes to find all references to past jobs that match whatever you've typed. In this manner you can reference all past jobs (and the customer information from them) instantly.

Another feature is *SD-Mail* (press **Ctrl-F12**). It may sound like an almost silly thing for a typically small service office, but when put it into operation, you'll be surprised at how handy it can be (especially if you have technician's logging into the system daily to make their own *PostVisitReports*).

Of course there are systems for many other kinds of reports (such as wage, salary or commission reports, for example), there's a *TimeClock* function. There are various system to accommodate automatic reception of dispatches from any disparate sources. There's a method of integrating *CallerID* into *ServiceDesk*. And there's a system to scientifically survey customers (while taking job orders) regarding why they called you, which yellow page ad they used, and so on.

And there are utility-type features. Some are built-in to the main *ServiceDesk* program, and some are in programs you run separately.

As examples in the first category, there's the '*Settings*' form (press **Ctrl-F1**), which you'll use to specify all the particularities of your own *ServiceDesk* setup and how you want it to run. There's a *Security* form that allows you optionally password protect a whole plethora of functions, and create different passwords designating for each which function they can unlock, and which than can't. There's an *Auto-Archive* feature that will run each of the housekeeping events (needed in several different contexts) automatically each night.

As examples in the second category, there's an *SD-Tools* utility you can use to customize your up-front service ticket. There's an *SD-Backup* utility that automatically makes hourly backups of all your *ServiceDesk* data onto a drive in your network other than the server (so if the server fails, you can be back up and running again almost instantly). And of course there's an on-screen, .pdf version of this manual.

All this and remember, this is but a thumbnail sketch. There are complete systems we've not even mentioned. To really understand what's here, you'll have to read the detailed descriptions found in the main chapters.

D. Flowchart of a Typical Job Sequence

Based on the above, we want to give you a simple list explaining a typical job sequence, from beginning to end. For simplicity, we'll do this with short sentences, and not much elaboration. We'll be describing a *typical* scenario. Actual details will vary, depending on circumstances:

- 1) You initiate a job by typing its information into a *Callsheet*.

- 2) While the customer is still on the phone and you're still in the Callsheet, you schedule the appointment via *On-Map scheduling* (**right-click** on customer's address line).
- 3) With all the needed info typed into a Callsheet, you click on it's *Job/Sale* button to create a *JobRecord* (**F7**). At first this record contains only information that was pulled from the Callsheet, but it's still its own independent record, and is the document from which the job will now be managed. The Callsheet has done its job and is no longer needed. At the same time you created the JobRecord, ServiceDesk also entered the appointment into the ScheduleList (**F6**), making it viewable in the DispatchMap (**F5**), and so on. Also in the same process, ServiceDesk will print the service ticket (i.e., invoice) for the tech to take on the job.
- 4) When the scheduled day arrives, you'll give the printed ticket to the assigned technician. In the DispatchMap (**F5**), you'll *check-off* the fact he received it.
- 5) When the tech returns with the ticket, he should have handwritten a description of the work he did, parts he needs to order, parts he used from stock, items of money collected, and so on. You'll now do a *PostVisitReport* in ServiceDesk (**Alt-F7** or **Alt/Ctrl-F7**), reporting on each of these details.
- 6) Assuming parts needed ordering, those particular details from the PostVisitReport are fed by ServiceDesk into your *PartsProcess* screen (**F8**), which you then use to manage the parts inquiry and ordering process. When the part arrives, you also check it in via this means. Once all needed parts have arrived, the job's *status* switches automatically into '*Working to Schedule*.'
- 7) Based on the above (and while reviewing those JobRecords that are in '*Working to Schedule*' status, for which purpose you may use the *JobsPerusal* form, **Shift-F7**), you call the customer seeking to get them scheduled for completion (perhaps using the *auto-dialer* for the purpose of dialing). If at first you don't succeed in connecting with a live human, you document your efforts via added notes in the JobRecord's history section. When you finally do connect, you'll again use the On-Map-Scheduling feature to figure a mutually convenient time, with the difference that here it's initiated from the JobRecord, while earlier it was initiated from the Callsheet which originated the job.
- 8) When the scheduled day again arrives, you again give the printed ticket to the technician. Again, within the DispatchMap you *check-off* the fact that he received it. He pulls the parts that were ordered, and does the job.
- 9) Again the technician returns with the ticket, and again you do a *PostVisitReport*, this time inputting information regarding what was done on this second visit.
- 10) Assuming no more visits are required, it's time to *closeout* the job by entering it's completion into the SalesJournal (via the SalesEnter form, **F9**). In other words, you'll now be recording the amounts that were involved in the completed sale. This essentially "puts the job to bed," and of course is essential to your bookkeeping. If it's a billed job, the system will simultaneously create an *AccountsReceivable* record (**F3**), from which you'll then manage that concern. If warranty claims are involved, now is the time to do that as well, using the *FinishedForms* (**Alt-F4**) system.

That's the typical sequence, in a nutshell. There are many more potential details, a plethora of systems designed to augment the above, add flexibility and security, etc., but those at least are the highlights of how a job *typically* progresses from beginning to end.

E. Thumbnail of a Typical Daily Process

Having now examined the basic structure of a ServiceDesk “factory”—along with the typical sequence for a single job—we’ll now overview how that all stacks up in terms of the typical sequence of work tasks, throughout the day, in a typical ServiceDesk office. In fact, to make it more concrete, we’ll describe a typical day in our own office.

Beginning at approximately 7:00 am our secretary Brenda arrives and immediately un-forwards the phones (we forward the lines to an answering service when we’re gone at night). She then calls the answering service and informs them we’re ready to receive our messages. They push a button on their end; their computer dials ours, ServiceDesk picks up, receives the messages, and inserts each into a waiting Callsheet. Brenda then begins dealing with the messages. Some are from people who want return calls (which of course she does). Some describe service orders (with appointments) that the answering service booked for us. If the latter (or if she takes a service order herself on the basis of a return call), she finesses the information on the Callsheet somewhat (it’s rarely perfect as received from the service), then creates a job from each (often I’ll hear the printer bang out about five service tickets in short order at this time).

When each job is created, its appointment goes automatically into the ScheduleList (along with ones previously scheduled), and is viewable from the DispatchMap. So, about this time, Brenda takes a look to see what’s shaped up there. She’ll do some juggling, moving this job to one tech’s schedule, taking this from someone else, and so on, trying to give each a sensible route that’s as efficient as possible. If she’s having difficulty, she may ask for my help and I’ll go to work on it (or I occasionally just volunteer).

At the same time, the phone is ringing with new calls. Some are people inquiring about existing jobs, and it’s always a simple matter for her to instantly find all the information she needs to intelligently respond. Others are from people wanting to order service. Most she handles entirely on her own in about a minute, writing the order, scheduling, deciding which technician to assign, and creating the job (as I hear the printer banging out still more service tickets), and so on. Some people want to discuss price first, and these she transfers to me (at the same time transferring the Callsheet, on which she’s already taken some information, to my desk). After I’ve discussed enough to persuade the caller to schedule service, I’ll usually transfer back to Brenda. If she’s presently got a couple of lines going, however, I’ll take the rest of the information myself (filling in the Callsheet), schedule with them, and create the job (thus hearing another service ticket bang out on the printer).

By the time technicians begin arriving at around 8:00 am, we’ve typically got their schedules pretty much filled. At any rate, usually they’re not ready to head out immediately, because first (unless they’ve done it the afternoon or evening before), they need to make PostVisitReports on their previous day’s work. We provide a dedicated desk and computer for them to use for this purpose, and their arrivals are staggered so they’re seldom in each other’s way.

Once each technician has made his reports and turned in the previous day’s tickets, he asks for restock to his truck. Brenda or I strike a few keys and instantly see what he needs, walk into the storeroom and pull it for him, then strike a few more keys to report what’s been transferred. Then he gets the stack of invoices that’s assigned as his current day’s work (which Brenda has put together based on how we sorted out each technician’s route from the DispatchMap), and he heads out to do it. At this point Brenda goes to the DispatchMap and (since the technician has now actually received these jobs) checks off each one as having been dispatched.

When the buzz of arriving and departing technician's is over (of course with calls coming in intermittently all the time), Brenda is usually ready to begin transmitting Parts Inquiry/Request documents to various suppliers, in response to the requests for non-stocking parts made by technicians in connection with their PostVisitReports (i.e., those just made in reporting on the previous day's jobs). For this purpose she uses the PartsProcess system, and completes the task in very short order.

With that task done, Brenda will probably next turn to the stack of invoices the technicians turned in. Most are completions, and with them are various amounts of money. She separates completed invoices from those still pending and places each into their own stacks. Checks and Bankcard slips go into a place we've established for the purpose, and cash into a normally-locked cash drawer (she keeps a handwritten journal there indicating how much cash was received in conjunction with each invoice; and this can be compared later to the ServiceDesk FundsJournal if there is any discrepancy). She further separates among the completed invoices between those that are paid in full and those that are to be billed. Then with these two stacks she begins making entries into ServiceDesk to record the completed sales. Each invoice is then stamped (with a simple little date-stamper) to indicate the fact that it was recorded to the SalesJournal. Those that are fully paid go into a stack (eventually relegated to boxes) where we keep all completed-and-paid invoices. For those that are being billed she prepares envelopes to mail the billing copy, and puts our office copy into a system of slots we maintain for that purpose.

Other invoices turned in by the technicians, obviously, represent jobs still not complete. Most involve part-order/inquiries (which she's already done via the PartsProcess system). She puts these into a slot indicating their status. Some may involve jobs where the customer stood us up, or for some other reason needs rescheduled. In any event, she'll call in the attempt to reschedule each. If she succeeds, she immediately creates a new appointment via facilities in the JobsCurrent form (with relevant JobRecord loaded). Otherwise, she records the fact of her effort in the job's History (also via the JobsCurrent form), and the invoice goes into a slot that's provided for this category of situation.

During all this time, naturally, we're also getting jobs dispatched to us by fax. Brenda's pulling them from the fax machine as they come in, inputting the information into Callsheets, and calling homeowners in the effort to schedule. When she reaches them, she immediately schedules and creates a job (more banging of invoices from the printer). When she fails, she documents the effort in the respective Callsheet's MoreInfo form.

By late morning we usually get our UPS shipment. Now it's Brenda's task to unpack the boxes, see what's arrived, and match it to orders we've placed in connection with various jobs. For the last purpose she again turns to the PartsProcess system. This helps her match the parts to previous orders and their connected jobs, and she uses the system to document the part's arrival, its cost, and so on. She then pulls the invoices for the connected jobs, and begins calling the customers to schedule for completion of their repairs. Again, if she reaches and can schedule with them, she does this immediately from the JobsCurrent form, then places the invoice into a slot (hanging on the wall) for the day and technician it's been scheduled for. If she fails, she documents the effort within the job's History.

While all these processes are taking place, naturally, the phone continues to ring with new callers wanting service. If they are anxious for service the same day, Brenda does a quick look at the DispatchMap to see if it looks like a suitably near technician can fit it in. If so, she schedules the customer and creates the job (more banging of invoices from the printer). She then pages the technician and places the job's ticket on her desk as a reminder that it's not yet officially dispatched. When she finally speaks with the technician and informs him of the job, she'll go to the DispatchMap and check it off as having been dispatched.

Around noon or so, we usually get our mail. With it there are typically a few checks in payment for past jobs. Now it's Brenda's task to match these checks with the physical invoices they pertain to. Then, with the checks and connected invoices in front of her, she reports on these payments using the Funds form. Assuming the invoices are paid-in-full at this point (usually the case), she places them into the same stack of completed-and-paid invoices where we also put paid-up-front invoices (these simply had a detour first before making it there). ServiceDesk will have done the work in the meantime (based on her report) of checking off the Receivable record, making a final entry to the SalesJournal, and so on.

By early afternoon Brenda will have received responses from most of our suppliers on the Part Order/Inquiry requests that she earlier transmitted. If they confirm they are shipping an item, or if they provide a requested price, she'll enter information accordingly into the PartsProcess system. If it's a matter that involves calling a customer with a price for their approval, she'll do this, and document the process within the job's History, and so on.

In any event, these are the typical ServiceDesk-managed processes that occur every day in our office—and that should (for the most part at least) occur in yours when you're operating in the ServiceDesk system. Hopefully, this narrative will help you have a general comprehension as to how the various pieces fit—that several can now, in fact, be considerably more automated than described here. Regardless, the basic process is that you manage calls and incidentally create jobs using Callsheets. You then manage jobs from their JobRecords, using several connected systems, including PostVisitReports, the PartsProcess system, the InventoryControl system, the Schedule and Dispatch system, and so on. Finally, you manage completion of jobs by recording each to the SalesJournal, then manage receivables, and so on. That's it in a nutshell.

Of course (and as an aside here), also remember there are processes that occur more intermittently. About twice a week, for example, Brenda uses the FundsManagement system to prepare and make a deposit. Once a week or so, I'll use the InventoryControl system to create an order for restock (or perhaps two or three different orders, to various suppliers). Once a month, we'll create and send billing reminders. And so on.

F. Thumbnail of the Transition

There was once a time when we pretty much just threw all this stuff at you, and said “go for it; make it work!” At that time, this section contained a long set of suggestions—tips on swimming once you were thrust (ready or not) into deep water. Now (as mentioned elsewhere) we have step-by-step plan that can be summarized, here, more succinctly: Read through the remainder of this chapter (up to but not including page 53), then skip ahead to Chapter 13 (“A Beginner's Guide to Easy, Step-by-Step Implementation,” page 231), then follow its specific prescriptions.

As an additional note, you may notice that for several of the systems in ServiceDesk, there will necessarily be a period of overlap, during which old processes (initiated under your previous system) persist in the pipeline, as it were. Don't worry. While in the abstract this sounds like a major complication, in practice it's seldom so bad. There will be minor complications that you simply must think through logically. Since you've been running a business already, we're sure you're up to it.

In any such case, of course, there will be a decision as to whether old, already-in-the-pipeline items should be allowed to persist within the old context (until such pipelines are finally emptied), or whether to

retroactively import them into ServiceDesk. Typically, we think its probably easiest to let the old items (whether consisting of job records, accounts receivable records, or whatever) run their course within their old pipelines. But of course the choice is up to you.

Regardless, please know that while the transition into ServiceDesk may be somewhat challenging, as a practical matter (and for most of the functions, anyway) there's very little work aside from mere learning. The biggest thing is to just "do it." With very little upset to the flow, you should be quickly operating, and with far greater efficiency and ease than before.

G. General Observations

There are a few matters of generality that, with a bit of explanation, will help you understand ServiceDesk better.

i. Simplicity versus Function

We have been determined in designing ServiceDesk to keep everything as simple as possible. At the same time, we've also been determined to provide maximum utility and convenience. Unfortunately, it's a sad reality that these ends sometimes work at cross-purposes. To accomplish a particular function or convenience, we almost invariably have to add an accompanying complication. Roughly stated, the result is a package with many features that may seem complex when viewed abstractly. Yet every complication has, as its ultimate purpose, the goal of simplicity and ease in actual use.

As one small example, you'll learn on following pages that you can instantly link to an item's JobRecord from its reference in the DispatchMap (simply by **Ctrl/Right-Clicking** on either of its DispatchMap references). To know you can do this (and try to keep track of the fact) is a complicating factor. However, having learned and eventually memorized the trick, you'll find yourself possessed with a powerful tool, one that makes instant access to wanted information much more convenient, intuitive and easy. It's a complication, yes, but ultimately a very simplifying one.

Virtually all ServiceDesk complications are like this. There're many of them, which means there's much to learn if you're to understand (and take advantage of) every detail. Once you learn them, however, you'll find the power of ServiceDesk is ultimately, and gladly, one that simplifies your life grandly.

ii. Messages: Cautionary, Steering, Informational

More than any program you've ever used, you're going to find ServiceDesk *talks* to you. Not in the spoken word, of course, but with messages. It's speaking to you for the very purpose of helping you use it more properly, efficiently and optimally. If you "listen" to what it's saying, you'll find it helps you greatly. If you don't, you'll soon find yourself calling us with some frustrating problem. You'll then be embarrassed as we show where ServiceDesk has been all but "shouting" at you, explaining how you need to do something differently to avoid the very issue, and yet you ignored it. Please don't let that happen.

To emphasize, when we get calls as just described (and respond by showing the user they were *ignoring* help ServiceDesk itself was presenting), a common response is: “*we get so many of those messages we just ignore them.*”

Well, duh!

Would you care to guess why these people get “so many messages?”

It’s because they *started out* ignoring just one or two. Because they did not address those, they kept coming. Then new matters arose, and they ignored those messages too. Pretty soon, they were getting messages with such frequency that it was impractical to pay attention. But they only got into such a state because of repeatedly ignoring things ServiceDesk was trying to say. Marriage with your spouse is not so different; if you don’t listen, the relationship fails. Please listen, and respond.

Consider, furthermore, when you maintain a practice of “listening” and appropriately responding, you’ll encounter very few messages overall. You’ll avoid that “vicious cycle” where, having failed to respond to some, you get even more, and pretty soon it’s such a nagging “yak, yak, yak” as to render you numb and deaf.

As a rule (and to be clear), ServiceDesk should be starting itself virtually every time without a single nagging message, and any other messages should be the exception rather than the rule. This will be your experience so long as you address matters that messages ask you to.

iii. Bugs, Error Alerts and System Crashes

Any and every software of more than minor complexity contains bugs. ServiceDesk is no exception. The consequence of a bug is that either you’ll see a result different than it should be, or you’ll see a message announcing some kind of error has occurred.

Usually, these latter situations will involve what are called “handled” errors, meaning the program itself has caught the error and is dealing with it as best as it can. In particular, in almost all such instances ServiceDesk will describe the kind of error, its location, and offer to send an email to us containing this information. We strongly encourage you to proceed in forwarding such emails. They help us amend whatever weaknesses as may continue leading to such annoyance. We pay close attention to every such item, and report back to you regarding whatever progress your information helped us to make (typically, we’re able to pounce right on the underlying problem and correct it). To make this emailing work, you *do* have to have an installed and operational email program in your computer (one that’s “MAPI-compliant such as Outlook, Outlook Express or Eudora). Outlook Express comes with Windows, so if you’ve not been using it, it’s probably just a matter of setting it up.²⁰

In regard to *any* potential for bugs, please be assured ServiceDesk is rigorously tested in actual, day-to-day operation in many different businesses—and we’re very determined to make and keep it as bug-free as possible. Certainly, all bugs that happen with either consistency or frequency (and that we also learn about) are very promptly fixed (typically, in the same day we learn of them). Overall, you should not encounter bugs—in day-to-day operation—more than once every several days, if that.

²⁰ Once in a great while, you may encounter what’s called an “unhandled” error, which means one that ServiceDesk did not trap and handle on its own. In this case the resulting message will be less informative, and it will kick you out of ServiceDesk, meaning you have to re-start thereafter. As we’ve continued to refine ServiceDesk, this happened less and less, until now it’s almost unheard of. Regardless, if you can let us know of any such situation that occurs repeatedly, we’ll appreciate it.

When and if you do encounter bugs, it will likely be for one of a few reasons:

1. Each time we introduce *new* features (which, of course, we do frequently), there is an enhanced possibility for new bugs. While we test carefully before releasing anything new, occasionally it requires the more strenuous environment of real use before a new bug is discovered.²¹
2. Occasionally a particular user (probably you) will find a way of doing something (some sequence in steps, some different value within a string, etc.) that is different from what other ServiceDesk users have yet done. This unique situation may expose a weakness that, because other users have not yet presented the same circumstance, was not previously apparent.
3. Most of the ongoing, real-world testing of ServiceDesk is within actual service operations that have already *been* using it for some time. This means the various data files already have information in them. If you have just installed ServiceDesk and are in the beginning processes of experimentation, by contrast, most of your files are still *empty*. This presents a different environment than the established, ongoing one, which you'll also soon occupy. For obvious reasons, no user stays in this state for very long, so it's an environment that is much less rigorously tested for bugs. In consequence, you're almost certain to encounter a significant number of bugs when first experimenting with ServiceDesk—far more than would be acceptable in real, established use. Unfortunately, we do not know of a practical method to debug as thoroughly for this mostly *empty-file* environment (for the simple reason that no one stays in it for very long). Please take comfort in the fact that, as you begin building information in your various files, virtually all of these errors should go away.

Regardless of the reason, if under any circumstances you encounter an error that can be reliably reproduced (going through the same sequence that produced it in the first place), we ask you to please inform us, either by calling or by forwarded email messages. We are extremely determined to keep ServiceDesk as bug and error-free as possible.

iv. Adapting System to Practice: Should You Bend or Should ServiceDesk?

No two companies do things exactly alike, and neither do software systems. For this reason, one of your major concerns when checking out a system such as ServiceDesk is whether it can adapt sufficiently to your particular needs and ways of doing business, and to what extent you can (or should) adapt to it. There are a few assurance we'd like to make in this regard.

First, we have a pretty large base of clients at this point, and from them have already fielded many requests for added capabilities and/or flexibility in methods. Unless it involved a major structural change or something so unique there's little chance anyone but a single client could use it, we have almost always responded by adding the capability asked for. In consequence there's a lot more flexibility than may at first meet the eye, and whatever your particular need for something that may seem different from how it's first explained, chances are we've already encountered it with someone else and built-in the wanted solution.

²¹ Since we always ship the latest product to *new* clients, there's an enhanced probability that our new users will be the ones discovering such *newly introduced bugs*. We apologize for this. While it's less than ideal, it is unfortunately a Catch 22. On one hand we'd rather be more certain new users encounter no bugs. But on the other hand, we want to be sure they have the latest features and improvements. Unfortunately, it's impossible to have it both ways.

Second, if you still have a particular need for something to be done differently, and if it meets the criteria described above, we'll be delighted to add it for you (this is, after all, one of the ways we make ServiceDesk better).

Third and finally, please don't be unwilling to do at least a small bit of bending yourself. The fact is, no significant software system can ever be a perfect match for your former ways of doing business (unless you spend hundreds of thousands having it completely custom-written, of course). We understand change is hard. But remember, the system that we're asking you change toward is very well-proven (and is constantly being honed to a finer edge still) within the bowels of real, well-run and super-efficient service operations (it's not, in other words, some product based on abstract theory as thrown together by a bunch of programmers in an office tower somewhere). It's a terrific system. To feel good about it (and get over the emotional pain of transition), you simply must adapt at least slightly, and make it your own.

v. The Scheme in Regard to Job Documents

There is sometimes confusion in a new user's mind concerning exactly what we intend in ServiceDesk concerning various documents as might be connected to a job.

Unlike many others, ServiceDesk uses what you might call a *unified job document system*. What we mean by this is that we do not intend that you'll have one paper document to initiate each job (sometimes called the work order), a different paper for job quotations (i.e., a quotation form), and perhaps still another that after job completion specifies final charges (i.e., the final invoice). On the contrary, we generally intend that one paper document will be used for all such purposes, managing each of these needs in job performance from beginning to end.

In terms of nomenclature, we've noticed that even among businesses that already use this strategy, there are a many different terms for the one managing document. We've seen it called a "service ticket," "job ticket," "work order," and often simply an "invoice." We may use any such term interchangeably in this manual, but tend to prefer the simplicity of calling it either a "service ticket" or "invoice" (although, when wanting to contextually remind of its multipurpose character we may use "work-order/invoice" as a clumsy alternative). Regardless, just remember it's one and the same document we are referring to when using any such language.

More specifically, this document should typically be a pre-printed form (such as you may have a local printing shop make for you), the kind with your company's logo and similar information printed at top and various labeled spaces for all the different kinds of information that will need filled-in. It is ServiceDesk's task (as part of creating a job) to type the initiating job information (i.e., customer name, address, description of problem, etc.) onto this paper (see exhibit on last page of this manual). It's the same paper you'll give to the technician as his job assignment and on which (after arriving at the home) he'll write-in his diagnosis, calculate costs for the anticipated repair, and quote the customer. And it's the paper on which he'll write-in his start and end times, findings and actions, and on which either you or he must write-in the completed charges to your customer, note payments if they are received at the job, and give a copy to the customer as a receipt (or mail a copy if it's a billed job). In sum, this one document is also used as the final invoice (with everything except initial order information written-in by hand). And, most importantly it's the primary (typically the *only*) job-specific paper we intend to use in ServiceDesk.

At least, that's the general scheme, the standard mode that we expect will typically be followed. For a number of other and specialized purposes, we've allowed for a number of exceptions and variations (primarily

associated with the *Finished-Form* system, see page 180), but aside from dealing with such other needs, we think as a practical matter (and particularly when dealing with in-field service) the single document system makes the most sense.²²

Part of the philosophy in ServiceDesk, after all, is that we don't want to create more work; we want less. This is best accomplished if you're willing to abide by the one-job-document concept, at least for normal service routine.²³

Specifically, we suggest you use a three-part, self-carbonizing, pre-printed invoice. They can be of any size and design you like, but we suggest using the specific *type* where each part is glued at top so that each assembly (i.e., the three glued parts) is separate from every other (i.e., not connected to other invoices in a tractor-feed/fan-fold or similar format). If you have an appropriate printer with optional feeder on top, this allows you to simply maintain a stack of such forms in the feeder bin, and every time a job is created and the print command issues from ServiceDesk, a new form drops automatically into the printer and the job-initiating information gets typed onto it. It's super-efficient and works very well.

With this single, three-copy document, you should have all that's needed (in terms of a job-specific paper) for most every situation. To illustrate, consider these scenarios:

A. It's a regular, direct-paying customer. As in any other case, you create the job from within ServiceDesk and print its initiating information into appropriate spaces of your standard, three-part invoice form. Your technician takes that document, goes to the job, diagnoses the problem, and writes a description of the work needed on the invoice, along with anticipated charges. He then shows the quote to your customer. The customer approves and signs, and the technician immediately does the repair. On completion, your technician collects payment and notes doing so on the invoice. He then leaves its third part with the customer (to be used as his or her receipt), and returns to your office with parts one and two. Since part two is not needed in this case, it's discarded in the office, while the first part is used as the source of information for input to ServiceDesk regarding what happened on the job. Following this, it (the first invoice part) is placed in permanent office archives.

B. Same as scenario A, except here the anticipated repair requires ordering parts, and it's your policy to collect a deposit from the customer in such circumstances. Therefore, after he provides his quote (as written on the invoice) and has the customer sign, your technician will further request and receive the customer's deposit. While doing so, he'll note the amount received on the invoice, and leave its third part as both a deposit receipt and as the customer's record of the quotation. When he later returns to complete the job, he'll write final charges on the remaining two parts, collect the balance, and leave the second part with your customer as his or her final receipt. In this case, only the first part returns to your office, where of course it fulfills the same functions as in A.

C. It's a landlord/tenant situation, where you're doing work in a rental but billing the landlord. Your technician leaves the invoice's third part at the job location, for the tenant to keep simply as a work performance document. He returns to your office with parts one and two, the second of which your office mails to the landlord as a bill, while again (and as always) using the first for standard office functions.

²²The arguments for multi-document systems are that they allow each document to be specifically tailored to need. However, multi-document systems require more computer entry time, greater storage demands (both physical and electronic), and ever more documents to buy and print—all of which translates into less efficiency and greater expense.

²³We are also assuming here that you're still operating in the more-or-less traditional fashion that involves sending the tech out with a paper document in the first place. If instead your intent is to modernize even further (and have each tech operating electronically in the field, with no paper from the office), then much of this discussion is academic.

D. It's a home-warranty job. Your technician writes-in the deductible collected when at the job site, and leaves the invoice's third part as a deductible receipt (or merely as a work performance document if no deductible is collected). The second part, of course, is mailed by your office as a billing to the home-warranty company, and the first is kept as always.

As you can see, all normal service situations are covered, with just that one document to buy, type initial order information onto from ServiceDesk, write job performance information into, and store.

The primary exceptions are if you're doing across-the-counter, up-front sales from your headquarters location (whether of replacement parts or of new merchandise), or if you have clients that, after a job is complete, require special claims formats such as the infamous NARDA. ServiceDesk meets these needs via use of what we call the "Finished-Form" system, a setup of tools that allows you to create a *fully-filled-in* invoice image on-screen (i.e., one that includes everything, including items sold, amounts charged, etc.), and to then print it out, transmit it electronically, or whatever else is needed.

Even so, please remember that for the service end, at least (particularly for in-field service), we expect that the primary document will remain as that single work-order/invoice—the one that's common to every *job*, and that sees it through from beginning to end.²⁴

If you've been using a multi-document system for general service, we think you'll find the transition to a unified one most pleasant. Of course, when you're using one document for all these purposes, it requires a well-designed format. An example of such a design is provided in the Exhibits (last page of this manual).

H. Making it Easy: Suggesting a Visit to An Operating Office

In spite of our best efforts to make it easy, we have seen that, for some at least, the transition into ServiceDesk use can be intimidating. This is mainly a *conceptual* problem, for the notion of doing so many things by automation can seem new and foreign. As such it may be discomfiting, especially because it may be hard merely to grasp, at first, how the daily procedures are expected to unfold.

In this regard, we have found a very easy and effective solution. If you're having a hard time getting there (and perhaps feel the basic comprehension is just not sinking in), you'd be amazed what a difference it makes simply to see the processes in real-life. A picture, as they say, is worth a thousand words.

So, if you're in this category, may we suggest that you simply visit a service office where ServiceDesk is in operation. Formerly, we used to invite people to the office where ServiceDesk started, but we no longer own that business. Fortunately though, we now have a client base sufficiently large to that there is a good chance that someone else is using the system not far from you. It's our experience that most users are happy to share their know-how with others, and in fact are only too thrilled to have a "newbie" visit their office and see how it's done (at least assuming you're not a competitor with them).

²⁴Of course, as you'll see on more detailed reading, there are variations even on this theme. If preferred, for example, you can arrange to print the entire invoice image onto previously blank paper, rather than using a pre-printed form (see page 295). As another option, if you're a sole practitioner and carry a laptop and portable printer; it may be very practical to manage the whole process electronically (i.e., this is apart from having multiple technicians manage things via a live electronic connection, which is another option). As another variation, you may want to dispatch jobs remotely to technicians at their homes (of via their cell phones, pagers, etc.). In this case, you may want to have such technicians keep some blank invoice forms, which they can either fill-in manually, or print via a variety of electronic means.

Chances are, you'll be able to watch over the shoulder of some ServiceDesk veteran as he or she goes through the ordinary tasks of a real day's work with all elements functioning in a real-time, authentic environment. We've found that after a mere day of such observation, most "newbies" are excited as never before with an "Oh, that's how it works" kind of comprehension, and anxious to return to their own office to begin implementing the many forms of magic demonstrated.

As a an easy means to find another company that might like to host you in this regard, may we suggest you use the bulletin board on our website to post your interest, and see who responds. We think it likely you'll get at least a few invites in short order.

Alternatively, if you'd like a Rossware staff person to travel to your office for hands-on training there, we can certainly do so, but naturally need compensation for time and travel expenses, making this option probably more expensive, and we think typically less effective than seeing the process unfold in an *established* setup. At any rate, assuming you want to avoid *any* additional expense, we've found by experience that virtually everyone can successfully muddle through on their own. It's just that with a simple day of observation, you can easily make a quantum leap in vivid comprehension.

I. Updates, Ongoing Support, Etc.

Like any "living" software, ServiceDesk is constantly being improved and expanded. As a user, you'll probably want (and *should* have) whatever is the current and "best" state of the product. Additionally, you may have questions or problems, from time to time, and wish to consult with us for help in resolving them. Finally, you may want to have your underlying custom data (particularly the custom StreetList we built for you) updated from time to time, to reflect changes in your area, or perhaps just improvements the Census Bureau makes periodically in terms of the underlying data it supplies to us.

For all these purposes, we have a system of various support services that, collectively, we refer to as "Ongoing Services." Basically, most ongoing services are free for the period of one year following an outright purchase, or during the entirety of any rent-to-own period. Following expiration of your "free" period, the rate for continued ongoing services (at time of this publication, at least) is \$25 per month. We will plan to contact you when your free period is nearing expiration to see if you wish to "subscribe" for paid continuation. Regardless, you should plan to make full use of ongoing services during the period it's free.

A few details in this regard:

First, if you need help with questions or problems (and have not been able to solve the problem on your own), there are three basic methods you may use in seeking help from others:

1. You may go to the chat-room that's featured on our website (www.rossware.net), and post your question there. There are a good many users of ServiceDesk who've gained substantial expertise, and we've created this feature in the hope that some, at least, will be anxious to share what they've learned with others. By posting a question, you'll give them an opportunity to be of service and extend the benefit of their experience. And, of course, the help will be beneficial to you, and in some respects might be even better coming from a complete peer, rather than from one of us here in the office.

2. You may email a question (or description of problem, etc.) directly to us. We work hard to answer all such inquiries quickly and with as much specificity as the circumstances demand. Often, this is much more efficient than trying to discuss matters over the telephone, so we encourage this method.
3. You may simply telephone here. In general, we rather like hearing from you. But, of course, our time is limited, and if matters can be resolved via methods 1 or 2, it's usually better. Still, if you need to talk, please don't hesitate to call. We want to help.

In regard to keeping the ServiceDesk product fresh and up-to-date (including not just the actual program code, but also the ServiceDesk manual and other non-custom elements), your task, historically, was as simple as logging into our website (www.rossware.net) and there downloading whatever is most current (we typically post updates every few days, though the majority involve only small, incremental improvements).

Now, for at least the actual program (which will involve the majority of your updates), we've made it even more simple. There's a facility built-in to ServiceDesk by which it can update itself. To access the facility, click on 'File Functions' from the ServiceDesk MainMenu, then select 'Update System'. ServiceDesk will use your active Web connection to connect to our site and determine whether an update is available. It will inform you accordingly. Assuming there is one, it will offer to download and install for you. With your consent, it will then do so.

Whether you use this new automated method or the older one of logging into the website manually, you'll need a user name and password. To set these up, you'll need to call us (800-353-4101).

Also (and as earlier discussed), please remember to periodically review our ServiceDesk-WorkDiary—where we maintain an ongoing log of each improvement as its made (and made available for download), together directions on how to use the new feature. You can go to our website (www.rossware.net) and navigate manually to this document, or you can place the following url as a shortcut on your desktop, allowing you to open the document with a single click:

<http://Rossware.net/SdWorkDiary.htm>

In regard to updating your custom StreetList with new data over time, this can only be done, obviously, as often as the Census Bureau publishes new data—which has historically been every one to two years. It's a substantial and specific-to-your-case labor investment for us to re-do this process for your particular territory, so it's not something we do lightly. In particular, we'll not likely do it during a "free" ongoing services period unless there's a commitment from you to continue paid services for some minimal period after the "free" era expires. Details will be provided you in each instance (again, historically this has happened only every two or three years). If you are enrolled in paid ongoing services, on the other hand, this updating process will be done for you without question.

A final concern may arise if you change your territory boundaries, or switch to a different map book, or have some similar change in the underlying circumstances for which we initially created your custom files. The cost of re-making your custom files, for such changes, is not included in ongoing services at all, and is subject to a per-incident charge depending on complexity (the exception is if you're not pleased with how we've created your custom files in the first place, in which case there's no charge for re-doing them more to your liking). Please consult with us for exact cost in each instance.

J. The SlideShow Demo

While it was developed to assist in marketing, the ServiceDesk *SlideShow* demo is nevertheless part of your provided program code, and is something you might find educational by way of introductory overview. One caveat is that, as designed, it's intended to be used in conjunction with a fully-developed set of operating files, which would show data in the various forms and windows as they are displayed. If you run the routine from your own package as newly installed, there will be no underlying data in these files, which will detract slightly from the demonstration's effectiveness.

To initiate the SlideShow, press **Ctrl-S** from any Callsheet. To pause it (or resume after pausing), press the **spacebar**. To terminate it, press Esc (you'll notice actual termination does not occur for at least several seconds after hitting Esc; please be patient).

Having read the foregoing discussions, you've now completed our broad and general overview of the ServiceDesk system. You should now have at least a vague understanding as to how the big pieces fit. With that accomplished (and assuming you are newly implementing ServiceDesk within your business), we urge you to skip ahead now to Chapter 13 ("A Beginner's Guide to Easy, Step-by-Step Implementation," page 231). The intervening chapters will be read when contextually applicable, as instructed from each of the Learning Steps described in that chapter.

Chapter 5

CALL MANAGEMENT

Hint: Read this chapter in conjunction with watching Video Tutorial Lesson # 2.

Callsheets are the absolute heart of the ServiceDesk system, and in particular of its Call Management system. These are the forms, arrayed four to a page, that constitute the primary screen in ServiceDesk. Each has facilities for recording all data that might possibly be relevant to incoming calls and service orders. Each has tools for managing those tasks as efficiently and easily as possible. And, besides their primary work of taking calls and forming the conduit for job-creation, Callsheets may also be used for inter-office communication, personal reminders, and so on. Think of them as little note pads (only in this case ones with great power), where you may instantly jot down anything that needs attention. And use them liberally, for there are an infinite number of Callsheets, all cheaper than paper.

A. Callsheet Navigation

Starting a new Callsheet is conceptually much the same as lifting one page from a note pad and starting on the next—except that, while with a note pad you might put unrelated notes on one page in order to conserve paper, in Callsheets there's no need. Start a new Callsheet as often as any new matter arises (such as a new incoming telephone call, for example).

To start using a new Callsheet, just move into the first one that's still unused (i.e., the first *vacant* one, which is always at the end of those already in-use). It's easy to move instantly to this position by pressing **Ctrl-PgDn** on your keyboard (this is the '*LastPage*' command). Of course, if you're already on the Callsheet page that includes this first vacant form, you can also move to it simply by clicking on it with your mouse. Or, you might use any of several other methods that have been created for moving around within the set of in-use Callsheets.

Most simply and obviously, you can move from one page of four Callsheets to the next by pressing your keyboard's **PgDn** key (note that ServiceDesk will not permit you to move beyond a page that still has any unused Callsheets on it). Similarly, you can move back to a preceding page by pressing the **PgUp** key. And, as noted, to move from one Callsheet to another within a page, you can simply click with your mouse. This method, however, requires moving your hand from the keyboard, which is not always efficient, so naturally a better means is provided. To move to the next Callsheet following your present using the keyboard only, press **Ctrl-Enter**. To similarly move to the Callsheet preceding, press **Ctrl-Backspace**.

While these methods are adequate for just moving from one place to the next, they're not always fast. Suppose you've got many pages of in-use Callsheets; you need to move a distance across them, and you don't want to stop at every station along the way. You need some means of *express* navigation, in other words. ServiceDesk provides three kinds of tools here, and you've already been introduced to part of the first: the 'LastPage' command (which moves you instantly to the last page of in-use Callsheets), executed by pressing Ctrl-End. Its counterpart is the 'FirstPage' command, executed by pressing **Ctrl-Home**.

Less obvious are the 'FormActiveUp' and 'FormActiveDn' commands. Their purpose is to provide a rapid, *search-type* access to those Callsheets that are presently active to your own desk (meaning there are tasks you should currently be doing in their regard). There may be several intervening Callsheets that are either in hibernate mode or active to someone else's desk, and you don't want to spend time searching through those to find the tasks that await you. The solution is to press **Alt-Backspace** to find the next preceding form that's active to your desk, or **Alt-Enter** to find the next such succeeding Callsheet.

Finally, there is a 'Specified Target, Callsheet Search' feature, which allows you to search for the specific Callsheet that bears any text string you might want to find. There are two ways to execute this search, both of which use the **Shift-F1** command. First, if you're in a Callsheet that already contains the string of text that you're interested in searching on, you can select that text, then press Shift-F1. The system will offer the selected text as your search target. You can hit Enter and proceed. If you're not already in a Callsheet with text that you want to search on, hit Shift-F1 anyway, and the system will prompt for entry of your target text. In either case, ServiceDesk will search forward, looking for the first Callsheet that contains text (anywhere within it) that matches your search target. On finding match, it will display it for you. You can resume search (i.e., looking for other matches in earlier Callsheets) by hitting Shift-F1 again. Once it reaches the front of the file (i.e., first Callsheet, the system will rotate around and continue searching from the end, back to the point at which your search began.

B. Desk Assignment and Status Options

On the right side of each Callsheet, you'll notice there are two sets of option buttons, one labeled 'ActiveDesk', the other 'Status'. These show which desk (i.e., which computer in your network) the Callsheet is presently assigned to, and whether the Callsheet's status is 'Current' (work needs done on it presently), 'Hibernate' (it's sleeping for a prescribed period), 'Delete' (this Callsheet is like a garbage piece of paper; you simply want to get rid of it), 'Job/Sale' (you've created a job/sale from this Callsheet), or 'Otherwise Done' (for some reason other than being used to create a 'job/sale', this Callsheet has similarly completed its purpose).

One detail you'll notice is that the last three status categories (*Delete*, *Job/Sale* and *Otherwise Done*) are framed somewhat separately within the list. The reason is simply to emphasize that it's by being moved into one of these three categories that the Callsheet is considered "done" (and it's only those that are in one of these categories that will be moved out of the current Callsheet workspace when, but not until, a *CallsheetArchive* routine is run). If the Callsheet remains in either 'Current' or 'Hibernate' status, by contrast, the supposition is that it's not "done," and it should (and will) not be moved out of the current Callsheet space, ever, so long as that remains the status.

Changing the selected option, when wanted, is done by left-clicking on the applicable button with your mouse, or by using the quick-key that is indicated for each of the possibilities. Indeed, you'll notice that the labels themselves indicate the key that, combined with Alt, will select them. Plus, you'll find that as you hold

down your mouse button while clicking on any of these, their labels will change to indicate the full quick-key combination that could be used. Thus, if you wish to switch a Callsheet that is presently in any other Status back into 'Current' mode, press **Alt-C**. If you want to mark one for 'deLetion,' press **Alt-L**, or to mark one for 'othrws Done,' press **Alt-D**. The other two status modes, 'Hibernate' and 'Job/Sale' are similarly selected, but more events necessarily occur upon their selection than a mere change in the Callsheet's status, so these will be discussed separately.

In regard to changing a Callsheet's 'ActiveDesk' assignment, the quick-key is **Alt-S** (as in "Switch desks"), or again, you can always use your mouse. The situation in regard to the Callsheet's display of potential desks is a little complicated, since ServiceDesk must work in an environment involving anywhere from one to several stations, and there are positions available for only two different desk-assignment buttons on the Callsheet. Essentially, if there's only one station, the second button will not be displayed (because, obviously, there's only one possible desk assignment). If there are two stations, one button will be labeled for the first desk, and the second for its counterpart (with each displaying the actual StationName assigned to it). In this case, quite naturally, you're allowed to switch to either desk by simply clicking on its corresponding button, or you can toggle between the two with each press of Alt-S.

If there are three or more stations, it becomes more complicated, since there's not space on the Callsheet for an infinite number of buttons corresponding to every desk a larger user might include in their system. In this case, the first button is labeled 'My Own' (i.e., my own desk), while the second may be labeled either 'Another's' (if the Callsheet is assigned to your own desk) or with the particular name of the other desk to which it is assigned (assuming it is, in fact, assigned to someone else's desk). Now, if the Callsheet in question is assigned to your desk and you want to switch it, the task is initiated by selecting the 'Another's' option (either click on it or hit Alt-S). At this point, ServiceDesk will display a list showing each of the other desks to whom it might be transferred.

Make your selection, again using either cursor keys or mouse (in the case of only two other options, i.e., three stations total, a single stroke of the up or down cursor key is sufficient for selection); at this point, the Desk Assignment will instantly change to the station selected. Conversely, if the Callsheet is presently assigned to someone else's desk and you want to switch it back to yours, there's no need for the list. Just hit Alt-S or click on the 'My Own' option.

C. Handling Supplementary Information

For most purposes, the Callsheet form in itself is quite complete. Right on its face, there are spaces for at least most the items of information that you'll need to take from a customer during the course of a call. However, during the extended course of call-processing and order-taking, you will occasionally need to record additional items of information. This section concerns those needs.

i. The MoreInfo Box

First and most importantly, there is the Callsheet's *MoreInfo* box.²⁵ Access it by clicking on any Callsheet's MoreInfo button, located in its lower-right corner (or press **Alt-M** on your keyboard).

²⁵ Another means of attaching added information to a Callsheet involves the *UnitInfo* system, described at page 213..

As you'll see upon loading it, the MoreInfo box contains two sections:

The first is labeled '*ExtraNotes*'. The purpose of this section is, if you have additional items of information, instruction (or whatever) that you want to have included on the work-order/invoice—and such additional information does not fall into any of the categories allowed for on the face of the Callsheet (or perhaps just wouldn't fit there)—you may include that information here. For example, perhaps there are special directions on finding the customer's home. Or maybe you as the boss have given the customer certain assurances about what the price will be, given certain conditions, and you want to communicate these to the tech right on the face of the ticket. Or perhaps you want to give the tech some detailed background regarding prior work history at the location. Whatever. This is the place for additional notes of such kind.²⁶

The second section in the MoreInfo box, labeled '*MoreNotes*', is rather different from the first. You'll notice, this is the section you are automatically placed into upon loading the MoreInfo box. And, ServiceDesk takes the liberty of beginning the entry, for you, with a short string of text indicating your own initials and the present date and time. Also note there is no intent for the contents of this section ever to print on your work-order/invoice. It's purpose is solely for internal documentation—as specifically related to elements of communication, or attempted communication, between you and the customer as connected with the Callsheet.

For example, suppose the Callsheet involves a simple message taken by the secretary indicating that so-and-so called to discuss a possible repair. You return the call, get an answering machine, and leave a message. You want to document having done so. Or, you've received a dispatch from an OEM or Home-Warranty client. All the order information goes into the Callsheet and the secretary calls the homeowner to schedule. She gets no answer, but wants document her effort. Or, a customer has called to complain about a technician's performance. She's going on and on describing details that you want to document. In any of these cases, the MoreNotes section is the ideal place to type-in the documenting info.

A suggestion here is to formulate a set of easy abbreviations that are understood by everyone in your office. You might use simple letters like "**Imor**", for example, to designate that you've 'left a message on the recorder', or "**Imwc**" for 'left message with child', or "**na**" for 'no answer', and so on. Thus, to document an attempt to connect with your customer, it becomes as simple as hitting **Alt-M**, then typing "**Imor**". Then simply hit Esc to exit from the form and save your entry.²⁷

You'll notice that a Callsheet's MoreInfo button is labeled "Empty" when the MoreInfo form contains no notes, and changes to "Yes" when notes have been added. Feel free to execute a Callsheet's MoreInfo button any time you wish to review or add more notes. You'll notice that, as a matter of course, ServiceDesk adds a new time and date stamp each time you enter the form—but don't worry, if you add no new notes after this inserted text (and do not edit existing notes), it will not be saved.

²⁶Please note that this box will not be functional until after you've setup the system for printing your invoices, and in doing so allowed a space for printing of these ExtraNotes upon your invoice (the space you allow, incidentally, being variable according to your own design). The reason is because not every user is likely to provide a space on their invoices for printing these notes. If there's no space on the invoice for the notes, obviously, it would make no sense for ServiceDesk to allow the user to create them. And, at any rate, ServiceDesk must know how much space has been allowed on the invoice in order to know how much text to allow within the form.

²⁷In earlier ServiceDesk versions we required a more complex action (other than simply hitting Esc) if you wanted your MoreInfo edits to be saved. The reason was because, each time you enter the form, ServiceDesk inserts that new Time/Date stamp into the text. We wanted the insertion to be automatic so that, without any further action or thought, you could simply begin typing new notes. The problem is that sometimes you go to the form simply to look at notes already there. In such instances, you don't want the automatically-added time and date text to be saved, so a means is needed for ServiceDesk to distinguish (as you exit from the form) whether to save the text in its altered form. Formerly, we forced you to consciously make the determination, and communicate it by using a different command depending on circumstances. This was dumb, for it made the process less intuitive, natural and simple than it should have been. Finally, we realized ServiceDesk can figure on its whether you, the user, have added any notes (or made edits) that should be saved. Accordingly, the process of leaving the form now requires nothing but a simple Esc in all instances.

ii. Telephone Number Notations

You'll notice that the Callsheet contains spaces for four different telephone numbers. *Typically*, we expect that the first telephone number in each of the two sections (CustomerInfo and LocationInfo) will be used for home telephone number, and the second for a business number. At least, this is what we do by convention within our office (while, of course, you may do however you please). Anyway, even when following this convention there is occasionally need to append a tiny note to one or more of the telephone numbers. Even if, for example, we're assuming that the right-hand number is for work, the question often remains whether (assuming it's a "Mr. and Mrs. . . ." kind of customer) it's the Mr.'s work that's referenced, or the Mrs.' (and it's awkward, when calling the work number, if you don't know which of the two to ask for). Also, there are times when you need to include an extension number. Other times you might want to indicate that it's a cell number listed, and so on.

For any of these kinds of needs, there are spaces on the Callsheet, *next to* each telephone number box, where you can add up to four characters of text—associated with that telephone number. But "wait," you say. You've looked, and can't see any such spaces there. You are correct. The spaces are *invisible*. One of our objectives has been to keep the Callsheet looking relatively simple, even while adding enhanced capability from time to time. Thus, when we added these spaces we elected to not have them show. Nevertheless, they are there, and are completely usable. If you click on any such area with your mouse, the cursor will locate there and you can begin typing. Or, from within any of the standard telephone number boxes, you can hit Tab and the system will immediately move you into it's adjoining Tel#Notation space.

iii. Attaching UnitInfo Sheets

A final method of dealing with supplemental information, in connection with a Callsheet, involves the UnitInfo system. This is a matter that's primarily discussed elsewhere (see page 189), for its application is much broader than a mere connection with Callsheets. In a nutshell, however, it's a system that allows you to create a specific record describing model number, serial number and similar kinds of information for a particular machine. This record may, in turn, be attached to either Callsheets, JobRecords or both. If, from a Callsheet, you have the need for such specific information in regard to a particular machine, this is another tool you should use.

57D.Methods for Auto-Insertion of Specialized Text

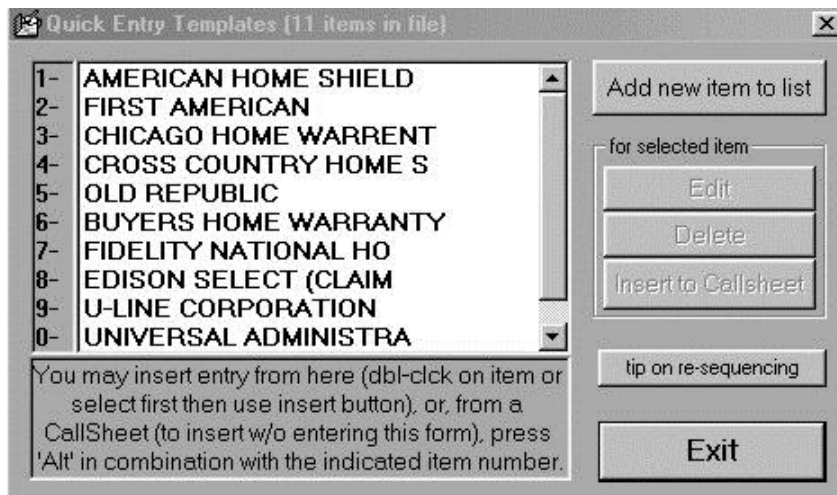
There are a plethora of tools designed to help you fill-in a Callsheet with minimum effort, maximum ease and incredible speed. In other sections (by way of example), we'll discuss how to insert a street name, grid reference, city name and zip code based on but a few keystrokes. We'll discuss how to insert the *full name,-address-and-telephone-number set* based on info from an existing customer's past job. And we'll discuss how to insert the Item-Type and Item-Make from a little drop-down list. In this section (by way of contrast), we'll discuss a selection of methods that are designed either to fill-in a Callsheet in its entirety, or at the least to insert a big block of previously prepared text. In fact, that last will be our first topic here.

i. Quick-Entry Templates

If you're like most servicers, you probably have a few clients for whom you do an extraordinary number of jobs. Perhaps you do work for home-warranty type or extended warranty companies, for example, or maybe it's a major manufacturer for whom you do new warranty work. Maybe it's a property management company that calls you, with some frequency, or any of several other possibilities.

Regardless of the particularities, in any situation such as this you're essentially going to be billing a third-party for work done at some location. Each time you write an order for this kind of third-party work, you'll want to put the paying-party's name, address and telephone number set in the "Customer Info" section of a Callsheet (i.e., in its *top* name and address section). You'll then want to put the homeowner (or tenant's) info in its "Location Info" section (i.e., the bottom section as provided for name and address). Please notice, if this is a paying party you're working for over and over again, it would be nice to be able to insert their full information set with minimal effort. That's where Quick-Entry Templates come in.

To create Quick-Entry templates, use what's called the *QuickEntries* form. It's accessed by pressing **Alt-Q** from any Callsheet, or via selection from the main menu.



The screenshot shows a window titled "Quick Entry Templates [11 items in file]". It contains a list of 10 items, each with a number and a company name: 1- AMERICAN HOME SHIELD, 2- FIRST AMERICAN, 3- CHICAGO HOME WARRENT, 4- CROSS COUNTRY HOME S, 5- OLD REPUBLIC, 6- BUYERS HOME WARRANTY, 7- FIDELITY NATIONAL HO, 8- EDISON SELECT (CLAIM, 9- U-LINE CORPORATION, 10- UNIVERSAL ADMINISTRA. To the right of the list are buttons: "Add new item to list", "for selected item" (with sub-buttons "Edit", "Delete", "Insert to Callsheet"), "tip on re-sequencing", and "Exit". A text box at the bottom left explains: "You may insert entry from here (dbl-click on item or select first then use insert button); or, from a CallSheet (to insert w/o entering this form), press 'Alt' in combination with the indicated item number."

On this form, you'll see the QuickEntries list (which will at first be empty). Click on the 'Add to List' button, following which you'll be presented with a form that includes text boxes identical to those in the Customer- and Location-Info sections of a Callsheet. Type data into these boxes in the same form and positions as you'll want to have it insert into every Callsheet/Invoice that pertains to the customer in question.

One important note in regard to formatting here. For any companies that use P.O. numbers as part of their service request (i.e., all home-warranty type companies), you should include space and labeling for that number as you setup the company's QuickEntry template. And, please note this setup must be in a particular location. Specifically, it must be in the top line of the form (i.e., the *CustomerName* line) toward the far right. The reason is because, during subsequent processing of the job, ServiceDesk will examine this particular line of text in search of a possible P.O.-type number. As the primary key of interpretation in this regard, it will look specifically for a pound sign (i.e., '#'). If it finds one, it will automatically interpret anything that follows as an actual P.O. (or equivalent purpose)-type Number.

Thus, if you're setting up QuickEntry text for American Home Shield, as an example, you'd want to have their actual name on the left side of the form's first line. Near the end of that same line, you'll want to put a label for where the unique P.O. number will be typed after insertion of this text to a Callsheet. Basically, you'll want to leave enough space after that pound sign for the particular quantity of characters the company

uses in its P.O. numbers.²⁸ In consequence, the first line of your AHS QuickEntry template (or that for any similar client) should look something like this:

AMERICAN HOME SHIELD	I.D. #
----------------------	--------

You might notice that in this illustration we've used the text "I.D. #" (rather than "P.O. #") to label the space we're setting aside for the client's job-identifying number. This is simply to illustrate that the particular terminology does not matter (and besides, this happens to be the terminology AHS has most recently adopted for their own use). It's the *pound sign*—and its location as the last character before the blank area that's left for the actual number—that's critical. The rest is purely optional.

In addition to providing space and labeling for the client's job-identifying number, you may also want to consider if the client requires you to include an indication of the homeowner's contract number. If so, you'll need to provide space and labeling for that as well. Basically, the same strategy is involved, except in this instance ServiceDesk will later look for such text in the *LocationName* line. In other words, whenever there is a need for ServiceDesk to pull and differentiate an indicated ContractNumber (as distinguished from other text), it will look (much as in the case of seeking a P.O.-type number) for a pound sign followed by *some* text. In this case it will simply be looking at a different line (the *LocationName* line), and interpreting text after a pound sign there as a ContractNumber. In consequence, your *LocationName* line of a client's QuickEntry template might look something like this:

CONT #

Note that again we've put in what is essentially our own labeling for a field and allowed space after for later insertion of a number. In this case, the left-hand portion of the line is left blank, for that's where (after the QuickEntry text is inserted to a Callsheet) we'll be typing in the homeowner's name.

Of course, many clients do not require that servicers regurgitate a homeowner's contract number, but do require collection of some deductible amount, which is a matter that for practical purposes is nice to have on each invoice. Thus, providing an indication (or space to insert) this information is another handy thing to place within a client's QuickEntry template. As it happens, we've found that clients requiring contract numbers typically do not require deductibles, and vice versa. Therefore, we've found it handy to place a deductible reference in the same place instead of a contract number, if the template is for a client that requires the one but not the other. If this client's deductible is always the same (or perhaps if it is even *almost* always the same), the reference can simply be placed there in its entirety, to be inserted to a Callsheet identically every time, as follows:

\$35 DEDUCTIBLE

If the deductible varied from job to job, on the other hand, you might want to prepare the template with labeling text that would prompt your operator (and provide the appropriate space) to insert the deductible amount, while leaving the actual number area blank (such as in the following):

\$ DEDUCTIBLE

²⁸ Actually, in our own service company we like to leave one extra space so that, as we're typing in the client's P.O. # after inserting the QuickEntry text, we have room enough to leave a space between the pound sign and the number (see invoice example on last page of this manual for an example). We think this *looks* nicer than if there is no such space between the two.

At any rate, you can see that in general the way you use the system is up to you. Just remember the one technical matter: that the system allows you to essentially create extra fields (in any Callsheet to which this text will later be inserted), one for a client's job-identifying number, another for a contract number. This is done by using a pound sign (i.e., "#") toward the right end of either the CustomerName or LocationName boxes. Whatever text is ultimately typed following such a pound sign in the CustomerName line (after the template-text is inserted to a Callsheet) will be interpreted by ServiceDesk as the client's P.O. number. Whatever follows a pound sign in the LocationName line (again, after the template-text is inserted to a Callsheet) will be interpreted as a Contract Number

For a full illustration of a completed QuickEntry template, please glance at the following:

Quick Entries (Editing)

AMERICAN HOME SHIELD

I.D. #

P.O. BOX 866

CARROLL, IOWA 51401 (\$300 AL)

800-326-4357

800-776-4663

Okay

\$35 DEDUCTIBLE

Cancel

HighVolumeClient' Abbreviation. This is the place to choose a two or three-character abbreviation, as applicable to this client, if you want it substituted for the full CustomerName in places such as the DispatchMap and SalesJournal.

AHS

This also results in non-inclusion in the normal DunningList, and other special treatment as typically preferred for clients you may invoice scores or hundreds of times each year. Leave blank to avoid such specialized treatment.

☐ Treat Sales to this client as Tax Exempt

☐ Substitute this name for Own company name in Narda form

Your Servicer ID Number (i.e., the number or code, if any, by which this client identifies you)

Your Account Number w/Distributor from whom you normally buy parts when servicing this client

Account Number of Distributor w/Mfg (if for this client you typically buy parts from that distributor)

Default amounts, if any, for S.Call and Labor amount

Type designation (if applicable)

American Home

The textual area here is filled in just as needed for insertion to our Callsheets when we're creating a ticket for American Home Shield. We've allowed just the right amount of space for their 8-digit job I.D. #s (in the space that you'll see at the end of the CustomerName line). Since they don't require that we provide homeowners' contract numbers, we've used the space where we might have setup for that to instead list the normal deductible amount. In addition, we've use a place where there was a little extra space (at the end of the CustomerCity line) to place in a little text reminding the technician of what our authorization limit is with this company. All of this provides a pretty perfect setup for what we want to have inserted to each Callsheet on which we write an order for this company. Of course, we've only discussed what's placed in about the top three-fifths of this form (where you see boxes echoing equivalent spaces in a Callsheet). At a glance, you can see the form obviously has spaces to fill-in a lot

more information. We'll now turn to discussing those.

First of all, please notice the box in the center of the form, the little one with a lot of labeling around it (labeling which begins with the words "HighVolumeClient Abbreviation"). That labeling says enough, perhaps, to make any re-explanation here somewhat redundant, but even so, we probably should expand a bit. The box is for your insertion of what we call the "HVC Abbreviation." The purpose of this abbreviation is to provide a small handle by which any such frequent client can be known within ServiceDesk. It is useful in many

contexts to have such a small handle, and useful as well for ServiceDesk to be able to distinguish this kind of a client (based solely on the on the fact that you've provided such an abbreviation) from others.

In the DispatchMap, for example, it makes sense to have a standard abbreviation for each of your very frequent customers, so that an American Home Shield job to the Smith residence, for example, can be rendered as "AHS-Smith," rather than as "Ameri-Smith". Similarly, it is better for completed sales to be rendered in such fashion. And it's important for ServiceDesk to discriminate in contexts such as making a standard DunningList (where presumably you will not want to include this type of customer), or in compiling your CstmrDbaseIndex (where you don't need thousands of entries under the specific name of a single OEM, Home-Warranty, property-manager or similar type of client), or when it conducts it's automatic search to verify, when you're creating a job, that you've not inadvertently overlooked a pending Callsheet order (see page 73). Your own part in creating this distinction is simple. In regard to any customer you want ServiceDesk to recognize in this set-apart manner, simply include a two or three-character abbreviation as part of that client's QuickEntry Template, in the space provided here. It's that simple.

Immediately below the center area as provided for designating the client's HVC abbreviation, you'll see a couple of items that can be checked off or not. They're pretty self-explanatory. Check as appropriate.

Moving on down in the form, you'll see places for inserting information as applicable to a client for whom you do warranty service (such as your Servicer ID #, default labor amounts, and so on). These are used by ServiceDesk when later filling in a NARDA for you. Basically, when filling-in a NARDA ServiceDesk will look at the CustomerName as it exists in the underlying JobRecord, compare that text to that which exists in the equivalent field of each QuickEntry template, and when it finds a match figure to itself: "Aha, I can take the Servicer ID # from this Template (along with other appropriate items), and fill them items into appropriate places in the NARDA."

Finally, there's one more little box in this form we should explain. It's in the bottom-right corner and is labeled "*Type Designation (if applicable)*". The purpose here is connected to automated dispatch (whether via the emailed-dispatch-insertion system or when using the WebBasedDispatchEnabler in conjunction with ServiceBench). Basically, if an automated system needs to insert a service order from one of your high-volume clients to a Callsheet, it makes sense that it should first insert the particular QuickEntry Template that you've setup for that client. But how is the system to know which of the templates you've made is for, say, Whirlpool Warranty, and which is for Frigidaire Contract Service, and so on? The solution is in that little bottom-right box. That's where you tell ServiceDesk, when you've made a template that is in fact for Whirlpool Warranty, that that's the template it is. Then, when the WebBasedDispatchEnabler is doing inserting such a service request to a Callsheet, it knows that's the template to insert first.

After you have completed and saved the template for any given client and exited the Edit form (click on its 'Okay' button), you should now see it as one of the items in the QuickEntries list. Notice that for the first 10 items in your list, there are single-digit numbers arrayed along the left side. This is to indicate that, in regard to these particular items in the list, there is a QuickKey method of insertion that does not depend on loading the QuickEntries form. Basically, from any Callsheet you can simply press the Alt button on your keyboard, combined with the number indicated for that entry. Thus, if American Home Shield template were the first in my company's own QuickEntries list (in fact, it is), I could insert the AHS QuickEntry text from any Callsheet simply by pressing **Alt-1** (or Alt-2 for the next template in the list, and so on).

Alternatively, if I can't remember the particular number to use for a given client in conjunction with this Alt-N method, or if I'm wanting to insert for a client beyond the first ten (meaning they're beyond the range for which the Alt-N method is available), I can simply bring up the QuickEntries form (press Alt-Q) and make my

selection from there (either double-click on the item, or select it then press the indicated 'Insert Entry' command button).

Naturally, any time you wish to change a client's QuickEntry template, that easy to do also. It's all self-explanatory from the form.

One final tip: Immediately after you've inserted QuickEntry text into a Callsheet (i.e., you're typing in an order as connected to a HighVolumeClient), you'll find your cursor is left at the beginning of the CustomerName line. If it's the type of order that requires a P.O. Number, hit the 'End' button on your keyboard. This will take your cursor to the space just after the pound sign, where in your template you've made space for that number. Thus, just after hitting that 'End' button you can begin typing the number. Similarly, if you've made space at the end of the LocationName line for a Contract number, you can tab down to that box and again hit the 'End' button to position your cursor in the proper place to begin typing that. Once it's typed, hit the 'Home' button to move the cursor back to the front of the line, where you can begin typing the homeowner's name (note that if you did this last in the reverse order, i.e., typing the name first, you'd upset the space as provided for the contract number).

ii. Job-Info From Dispatched Emails

While we don't want to give the impression ServiceDesk is designed particularly for servicers who do work for American Home Shield, receive dispatches through ServiceBench, or otherwise process dispatches as received from large, institutional vendors (it's not), it's nevertheless true that a good many servicers do such work. To assist these, we've created some very specific systems.

In particular, we've created some extremely advanced systems as encompassed in our Web- and Email-BasedDispatchEnabler utilities (see page 103), but these are separate programs, optionally available for an added price. Prior to creating these, we'd already created somewhat less advanced, less fully automated systems that are fully integral to ServiceDesk itself. If you don't want to purchase the more advanced utilities, you might still want to use these somewhat less advanced tools.

In regard specifically to processing dispatches as received via email from AHS or other companies, it may have occurred to you that, within each such email, you already possess—in electronic text format—virtually all the information that needs to go into a Callsheet. What a pity it would be to already have this information in such form, and yet nevertheless need to manually re-type it into appropriate spaces of a Callsheet. We'd not want you to be forced into this, and so created a system by which you can easily take that text, from an emailed dispatch, and instantly insert each element into appropriate boxes of a Callsheet.

The first step in using this system is to copy the applicable email text into your *Windows clipboard*. As explained elsewhere, the Windows clipboard is simply an unseen space that can temporarily hold any item of text, graphic, or whatever that you want to place into it. You can copy things into there, then 'paste' or 'insert' them back out into other locations. It's very handy.

To copy an applicable email dispatch into the Windows clipboard, just go the applicable email and press **Ctrl-A** on your keyboard. This is the windows command to 'select All', and will cause the entire email contents to be highlighted. Now press **Ctrl-C**. This is the Windows 'Copy' command, and will copy the text that you just selected into the Windows clipboard.

Now that you've got the email dispatch text in your Windows clipboard, the next step is to go to a fresh Callsheet, and from there simply press **Alt-I**. This is the *ServiceDesk* command to 'Insert email contents' In

result of the action, ServiceDesk will essentially read through the text that it finds in the Windows clipboard, looking for key words that tell it where to expect the particular text that should be inserted into various boxes of your Callsheet. Within an instant, it should find each such item of text, and insert them into appropriate boxes of the Callsheet for you.

Pretty easy.

Of course, since ServiceDesk doesn't have true intelligence, there may be a few items of doctoring to do with the inserted text in your Callsheet. ServiceDesk doesn't know how to distinguish a first from last name when there's no comma, for example, so you may need to determine which of the names provided is a last name (AHS dispatches are typically in standard first-name(s)-then last format), and put that name at the front of the line with a comma following (again, use standard Windows *cut and paste* techniques to make this easier). And, you may need to backspace from the end of the address-line to bring up your ServiceDesk StreetList and do an insertion with grid reference, and so on (really, the needs should be obvious).

As mentioned, we now have a more advanced system that *completely* automates things (see page 103), but if you don't want to get it, the above can be an excellent solution.

iii. Job-Dispatches From ServiceBench

Much as there's an older system designed to work with *emailed* dispatches, there's also an older, totally-within-ServiceDesk system designed to work with dispatches as received through ServiceBench (for those particular servicers that do such work).²⁹

If you are such a servicer, you may know you can go to their website and download a file that contains your pending dispatch information. Obviously, you could read the information out of this file, and manually type each new job-item into a Callsheet, thus initiating each job as received. However, that would be a little work. Since the info is already in digital form, it makes much more sense to let ServiceDesk do the work.

For this purpose, all you must do from within ServiceDesk (after you've downloaded the dispatch file from ServiceBench) is, from within a Callsheet, press **Ctrl-Shift/I** on your keyboard. The system will then prompt you to identify the dispatch file in question, and once done will instantly pull each job-item from it, inserting all the relevant data into correct places of Callsheets (then, of course, it's up to you to process further).

Again, we've now created separate utilities that *completely* automate these processes (see page 103), but if you want to take advantage of just what's available within ServiceDesk itself, the above should do nicely.

iv. Message or Service-Order Info From Your Answering Service

We've found that most service companies *don't*, in fact, use a live answering service, but there are many advantages in doing so. Among these is the little known fact that you can arrange with such services to handle calls in a manner such that most callers will never realize they're speaking with an answering service (assuming instead that they're talking with office personnel). And, working in that fashion, you can in fact have

²⁹ If you do not know, ServiceBench is a web-based company that specializes in providing informational-process links between manufacturers of products and their independent servicers. Originally, their domain was primarily concerned with claims administration (as involved in OEM warranty service), but more recently they've become involved in the dispatch end of work as well. That's where the topic of this section comes in.

an answering service take service orders for you, even scheduling the calls. In this manner, you can avoid losing many potential jobs—that would otherwise go to a competitor who gives the caller the satisfaction of speaking immediately with a human, and getting an actual appointment scheduled.

At any rate, if you use (or should decide to use) a live answering service, one of the associated tasks is getting your messages (including those regarding service orders taken) from them. ServiceDesk provides a means for this to all be done electronically, with each message/service order inserting itself for you into appropriate spaces on a Callsheet. It makes the process very convenient. The process of setting up for this is described in the Appendix, beginning at page 278.

E. Creating Addresses with Auto-StreetFind

As you begin typing in a street address (regardless of whether in the Customer- or Location-Info section of a Callsheet), ServiceDesk is watching. It's waiting for you to finish typing the number portion of the address, and begin typing the street name (technically, it figures you're on the street name when you've finished typing the first "word" of text, as in the "**123-A**" portion of "**123-A WALNUT LN**", add a space, then continue typing).³⁰ As you do so, ServiceDesk watches with each keystroke to see if there are matching entries in your StreetList. As it finds any, it will immediately display them in a drop-down list. As the user, you should continue typing as many characters as are needed for the street name you are seeking to appear, then select that item for entry.

Actually, there are three ways of indicating your street selection, and you should use whichever is easiest. First, you may left-click with your mouse on the desired street. Second, you may use your keyboard cursor controls to move into the list and onto the desired street, then hit Enter for your selection. Finally, if you've typed enough characters to make the desired street come to the top in the list, simply hit Enter: ServiceDesk will assume you want the top item and insert it for you.

Aside from these alternatives for *selecting* a street, there are also options for specifying the *type* of insertion you prefer. Most obviously (and as the default method upon selection), ServiceDesk will insert simply the street name, its map grid reference, and the city name into appropriate spaces of your Callsheet. This format is obviously appropriate if it's simply a service location you're typing in, and you do not anticipate any subsequent need to mail a bill or other items.

Of course in many instances it will be a third-party payer's address that's being typed, in which case there's no need for a map grid reference, yet you do need full state and zip info for subsequently mailing of the bill. To designate this type of entry, use the right mouse button instead of the left when clicking to designate the street, or if using keyboard method make it a shift-Enter rather than Enter alone.

You might notice that, while your customized StreetList includes only abbreviations to each city, it will be the full name that ServiceDesk inserts for you. This conversion is based on a list of 'CityNames and

³⁰ Pay attention to the fact that ServiceDesk expects the street name to begin as the *second word* in an address line. This is important, for if you were to type something like "**251 A MAPLEWOOD LN**," you could not expect ServiceDesk to find Maplewood Ln, for it would in fact be looking for (and not finding) a street supposedly called "A MAPLEWOOD LN." In the case of an address of this type, use "**251-A MAPLEWOOD LN**" instead. The way ServiceDesk see the street name as beginning at the second word position in the line, just as it needs to.

Abbreviations' that we created specifically for your area. A copy of this list is provided in the Appendix (page 323). You may find it useful in deciphering the abbreviations.

In a perfect world, this would be all the discussion that's needed regarding use of the StreetList. However, on occasion you're likely to find that some things, at least, are imperfect. You may not find the street you are seeking, for example. Or you may find the city name that's assigned is erroneous. To deal with such issues, and to understand in greater depth some of the underlying mechanics, please (as need arises) consult the technical discussion that begins in the Appendix on page 287.

F. Using the ItemLocate and Auto-Dial Features

When you have a caller on the phone and are scheduling an appointment, you'll want to know where they're located, especially as compared to other jobs in your lineup. Just right-click³¹ on the address line. Your DispatchMap will immediately display and show the location, flagged in red. Hit Esc to return back to your Callsheet. (The same feature may be used from the JobsCurrent form, from whence you may be scheduling a return call after having ordered parts in.)

When you want to dial a number that is listed in any of a Callsheet's four TelephoneNumber boxes, just right-click on the number. You'll quickly see a form displaying the number, and should hear your modem go off-hook and sound the appropriate dialing tones.³² After your modem has dialed, go ahead and pickup your telephone's handset (or press your telephone's speaker button) on the appropriate telephone line. Now that your telephone is connected, press any key on your computer for ServiceDesk to hang-up, leaving you ready to conduct an unimpeded conversation.

This same feature may be used from the context of your DispatchMap (simply right-click on a job's list representation). Also, much as in a Callsheet, it may be used from any Archived Callsheet context, or from the JobsCurrent form. The DispatchMap application is particularly useful if you're attempting to locate a technician at some job site that's listed on his present schedule.

You may note that when this feature is used in conjunction with the CstmrDbase search (see page 209), you end up with a kind of built-in telephone directory. Suppose, for example, you did a job for an "Andrea Ward" at some time in the past, and you presently want to dial her. Just hit F12 to bring up your CstmrDbase search feature (see page 67), then type in "**WARD, A . . .**". At this point, you'll see a list pop up showing all jobs performed for her. Click on the most recent, and note her telephone number(s) displayed in the appropriate locations on the JobsArchived form. Just right-click in the appropriate location and, almost instantly, you're connected—all with just a few keystrokes.

There's still another use of the AutoDialer which may seem kind of silly at first, but is potentially handy when you need it. Do you ever want to dial one of those alpha-stated numbers (like "1-800-1-CALL US")? It's quite annoying to search for each letter on your telephone's key pad. We personally got fed up with this annoyance, and so created a solution. If you left-click on the label section corresponding with either the

³¹ If the simple right-click causes any trouble in this context, see the discussion at page 328.

³² There are some potential complications ServiceDesk must contend with when deciding precisely how a number should be dialed. One question, for example, is whether a '1' prefix should be placed before the number. Another is whether an area code should be included for numbers even within your own zone. Because the answers vary, depending on telephone company policy in your region, we've provided two items, in the net-wide section of the Settings form, by which you can provide the needed indications. See page 247.

Customer-Info or Location-Info section of any Callsheet, the auto-dialer will load with no number in it. At this point, you can simply type in the number wanted, but using letters if preferred. The AutoDialer will automatically translate letters into corresponding numbers, and dial for you. It's a tiny problem, but a convenience you might occasionally enjoy nonetheless.

ServiceDesk will look for your modem, by the way, at whatever CommPort is specified (by you) from the Settings form. On most installations, this will be CommPort 2.

G. Using the Customer Database From a Callsheet

One of your options in the 'local' portion of the Settings form is whether you want the *Auto-CstmrDbase-Search* feature enabled or not. When this feature is on, ServiceDesk will conduct a search of the CstmrDbase Index, in each of eight different Callsheet fields, as you are typing-in characters. These are the Customer and Location *Name* fields, the Customer and Location *Address* fields, and the home and business *Telephone* fields in both Customer and Location sections. For obvious reasons, ServiceDesk will not begin the search until enough characters have been typed to make it meaningful.

When ServiceDesk finds a match or matches between your entered text and items in the CstmrDbase, it will expand the Callsheet and display a list referencing each match. This will go on as you continue typing-in your entry, and on any reasonably-powerful computer (300 megahertz or above) ServiceDesk can easily maintain your pace, instantly updating the list with each of your subsequent keystrokes.

The CstmrDbase is based, simply, on the entire history of all jobs written and managed through ServiceDesk. More specifically, it includes the already-indexed portions of the *JobsCurrent* file and all of the *JobsArchived* file. By using an index set that refers to both files, the system is able to conduct complete searches in the tiniest fraction of a second, even after accumulating a history consisting of many tens of thousands of jobs.³³

Since references in the list may refer to either current or past jobs, there is a distinction in the manner each item is displayed, to let you know which category of job it references. Basically, if items are from the *JobsCurrent* file, their list references will be shown with a row of four asterisks before and after the customer's address. If from the *JobsArchived* file, the references will not include such asterisks. Thus, you can distinguish in an instant between references to jobs still pending and those from the past.

If you see an item in the displayed list that you'd like to view more fully (the list itself includes only name, address segment, and telephone number), you may either left-click on it with your mouse, or press **F1**. At this point ServiceDesk will display a shortened version of the *JobsArchived* form, with the referenced item loaded into it (if you used F1, it shows the first item in the list). Thus you can instantly see all pertinent details regarding the selected job, regardless of whether it's current or one that was completed years ago. In many cases, you'll want to look at more such items as you talk with a customer on the line. It's easy. Once a single

³³Because it references jobs already created within ServiceDesk, you'll obviously not get any reaction through this system when you are first testing ServiceDesk. Indeed, please don't expect to begin getting any benefit (or slightest CustomerDbase reaction) until after you've begun operating (and have, in fact, operated for a while) under Learning Mode 3 or above (see page 280). Even then, you'll have to be sure you've appropriately followed directions in setting up the systems that create the needed underlying indexes, and so on. For a detailed description of the system and precisely what is involved in this regard, see page 307.

list item is referenced and displayed, use your cursor keys to move up or down in the list and display adjoining items. Thus you can view all the jobs in a particular customer's history, in seconds.

In many cases, rather than wanting to review past history, you'll be more interested in using that past name, address and telephone number simply to fill-in your present Callsheet, and without having to type or ask your old customer for such info again. In this case, you may either right-click on an item in the list, or again press F1 to move into the list, then cursor down to the item and press Enter. With either action, ServiceDesk will insert into your Callsheet the same full name, address and telephone number set that were used on the previous job. You may also simply press Enter after you've selected a job for display, and ServiceDesk will similarly insert its customer data-set into your Callsheet. Alternatively, if you wish only to enter the name and telephone numbers from the CstmrDbase item (i.e., you just need the name and telephone number to remind you to check on something then return the person's call, or something similar), you may modify any of the above actions by holding down the Shift key. This will cause an insertion with address and city omitted. Also, you can do an insertion of the previous info set in "Recall" format by doing Ctrl-Right/Click.

In any event, each of these insertion methods provide a virtually instant means by which you can use a former name, address and telephone set and get them instantly into a new Callsheet—with something so close to zero effort as is almost magical (to help you remember all the various commands available in a Callsheet, there's an instantly accessible command summary; see page 76). Plus, you should be aware the system is doing even other things for you at such time, behind the scenes. For example, as you insert a customer's info set, the system simultaneously checks in your Accounts Receivable file to see if any amounts are outstanding from the same customer. If so, it brings it to the operator's attention, and offers to place a note on the invoice advising the technician to collect as he makes the visit.

While it does not require any appreciable time for ServiceDesk to conduct its search of the CstmrDbase, substantial resources are needed for your display system to expand and contract while accommodating a list that must appear or disappear in rapid response to your keystrokes (depending on whether matching entries are found with each new resulting strike). If your system is not particularly powerful and you type quickly, there may be enough delay to be annoying. If so, you may want go to the Settings form (use the 'File' section of the main menu or press Ctrl-F1) and disable the Auto-CstmrDbase-Search feature.

Whether your auto-CstmrDbsSearch feature is turned on or off, incidentally, you can still conduct the same searches, any time you want, and on the same fields. Simply press **F1** while your cursor is located anywhere in such a field on a Callsheet. Indeed, within our office (even though we always have the auto-search feature on), we find this last trick eminently useful. The reason is because it may occur to you *after* the data has been typed in (and the then-displayed list is now gone) that you want to again conduct a search on one of the fields. You could at such point use the F12-based search feature (see following paragraph), but that would require re-typing some of the same target text that, in this instance, already exists in your Callsheet. Or you might backspace into some of that already-existing target on your Callsheet, and by such action re-invoke an auto-search (assuming such feature is turned on), but then you'd have to re-fix whatever editing you did. In fact, it's far easier, whenever you already have suitable target text in any of a Callsheet's search-able fields (and especially if your cursor is already in that same box), to just hit F1. That invokes the same search as though you were typing in the field, but without actually doing so.

All this discussion, incidentally, is directed toward using the CstmrDbase from within a Callsheet (since using Callsheets is the larger topic of this chapter). You should know, however, that if you are not otherwise typing a customer's info into a Callsheet (or don't already have it there), there's a much easier way to conduct a search than by this method. Simply press **F12** to bring up the TechInterface form (as discussed more fully at page 209). From this context, it will automatically load itself in a format that's immediately ready to conduct a

CstmrDbase search for you. Your task is simply to begin typing characters from the name, address or telephone number that would be found in the record you're looking for. Immediately, you'll see a list popping into view (similar to that which is produced from within a Callsheet) that shows all matches. When you see an entry you're interested in (again as if in a Callsheet), just click on it for full display (or again you may simply press F1 to move into the list, then use the cursor keys to move up and down within the records offered). Experienced users will typically find themselves using this feature many times each day.³⁴

H. Callsheets Hibernating, and the Overdue Alarm

As you may have noticed, Callsheets not only store information, they also serve as a kind of "to do" note, like slips of paper you've placed on your desk to remind of particular things needing done (or like a faxed dispatch that sits on your desk reminding that a customer needs called for scheduling). As you may also have noticed, ServiceDesk helps you know when a Callsheet needs work done by following a simple convention. When work does need done, and needs done by you (i.e., the Callsheet's in 'Current' status and assigned to your desk), ServiceDesk keeps it illuminated at your desk. When work does not presently need done, on the other hand (i.e., the Callsheet's in any other mode), ServiceDesk keeps the Callsheet dim. Thus, when you create a job from a Callsheet (meaning its particular duty is done), or transfer responsibility on it to another desk, the Callsheet naturally goes dim, telling you at a glance that you presently have no work to do on it. It's much as if that piece of paper on your desk is taken away: you don't have to see it there anymore, nor feel nervous about whether there's any work for you to do in its connection.

This is great for those items that you can, in such manner, actually dispose of, but obviously, there are many items that you might have dealt with only in terms of immediate need, even while knowing the matter needs to be re-visited in an hour, a day, or whatever. Or maybe it's something you haven't taken care of at all, but it doesn't require tending to right away, and you intend to put it off as long as you gracefully can. In either case, you don't want an annoying piece of paper sitting on the desk bugging you during hours or days when you simply don't intend to address it. Nor do you want to lose track of the fact that, ultimately, you must.

The solution, when that piece of paper is a Callsheet, is to simply put it to sleep for whatever period of time you intend to ignore it. This is what we call Callsheet *Hibernation*, and essentially, it involves making a Callsheet go dim (so that you can in a sense sleep in its regard) for a specified period of time. To put any Callsheet to sleep, either click on its hibernate button or press **Alt-H**. This will bring up the *Hibernate* form, from which you can easily select the period of rest. Note that, as in most other forms which offer a single set of options, here you can either click on your selection with the mouse or move over it using your cursor keys, then press Enter for your selection. You can also specify a wake-up time that is not provided in the direct options, by simply typing it in.

You may notice that each of the sleep-period options have an accompanying "alarm period." This relates to still another feature that helps you know which Callsheets need attention, or, more specifically, which ones are *past-due* for attention. This feature is what we call the "Overdue Alarm." Basically, there's a built-in sentry in ServiceDesk that's continually monitoring all active Callsheets (i.e., ones in 'Current' status) to see that adequate attention is paid to them. If any go past some specified time without activity, ServiceDesk will begin beeping, and showing a notice in the title bar at top, to alert you to the neglect.

³⁴ Still another context is to conduct a search from text already existing within a JobRecord as displayed from within the JobsCurrent form, as described at page 120.

It is, essentially, this specified time period that's being described by the Hibernate form's indicated 'alarm periods.' Essentially, ServiceDesk will allow the 'alarm period' stated, after the sleep period ends and a Callsheet "wakes up," before sounding an alarm that the Callsheet is overdue for attention (assuming no attention has been paid to it during that period). Logically, there are longer alarm periods following longer sleep periods, and vice versa.

Of course, the alarm period that follows a Callsheet's awakening from hibernation is only one attribute of the Callsheet Sentry. For Callsheets in general (i.e., one's not subject to a specific post-awakening alarm period), the period is two hours. If, therefore, any Callsheet that's not in a specific post-awakening alarm period does not receive attention (i.e., work being done on it) over a space of two hours, ServiceDesk will begin sounding the alarm in its connection.

Of course, the beeping of an alarm can be annoying, and you may not have time immediately to satisfy its call. If this happens, you can silence the alarm by pressing Ctrl/Shift-LeftCursor (this combination is used because the keys are next to each other and are easy to press). Each press gives you one minute of silence. If you wish, you can hold the keys down and allow the Windows auto-repeat feature to run up several minutes of reprieve-time. By holding down for three or four seconds, you'll get an hour or more of silence (depending on the repeat rate you've set from Windows). Alternatively, if you do not like this alarm feature, you can turn it off entirely from within the Settings form.

Eventually, of course (and regardless), you should heed the alarm and do something to service your anxious Callsheets. ServiceDesk is easy in this regard. If you make any saved change in a Callsheet, ServiceDesk figures you must have done your duty for now, and lets the matter rest for another two hours. After that, if there have been no saved changes in the interim (and you didn't put the Callsheet to sleep for longer, or in any manner put it to final rest by creating a job from it, or something similar) it will inform you, by beeping again, to indicate that the item is again past due for attention.

It may be useful to know, at this point (and as explained in more detail later), that ServiceDesk saves your Callsheet edits, when there have been any, immediately upon any action that moves the control focus to another form. Thus, if you've just changed one or more characters in any of a Callsheet's text boxes, and you hit F5 to display the DispatchMap (or take an action to display any other form, even moving to another Callsheet), ServiceDesk will immediately save your Callsheet in the form it was left. ServiceDesk also does immediate saves when you change a Callsheet's status or desk assignment. Thus, it will immediately save if you transfer its assignment to another desk, or if you change its status from 'Current' to 'Otherwise Done,' or anything similar.

Regardless of how done, it follows that you can satisfy what might be called the Callsheet Sentry by invoking any Callsheet save. In fact, you can even cheat if wanted. Suppose, for example, that at the present moment you have some Callsheet that's beeping you. You don't want to give it any serious service, right now, and yet don't want to put it back to sleep either. Simply make it your active Callsheet, then hit Esc. This will invoke an immediate save, and ServiceDesk will be fooled (for another two hours, at least) into thinking you've paid it some attention.

I. Scheduling From a Callsheet

There are two contexts in which you may be making appointments with your customers. First, is when you're originally scheduling the job, generally at the same time your customer first places his or her order. Second, is any subsequent scheduling, such as if the first appointment was canceled and a new one is needed, for example, of if the technician found on the first visit that he needed to order a part, and therefore rescheduling was required after the part arrived, etc. It's important to conceptually distinguish between these contexts, for just as the actual contexts are different, the mode of scheduling from within ServiceDesk also varies, depending on which context is involved.

Specifically, and for the first context, if you're making a first-for-the-job appointment with your customer, it follows you're probably still at the point where you're typing job-initiating information into the Callsheet. Thus, there is as yet no JobRecord for the job. To this point, it's represented only by the Callsheet upon which you are typing various kinds of information. Obviously, the first appointment is one such item, so naturally there's a place in the Callsheet for you to enter a short notation describing it. Logically, it's the box labeled 'Date and Time'. Of course, the simple fact of entering an appointment within this box does not, in and of itself, create an entry into your ServiceDesk ScheduleList (this is a separate file where ServiceDesk keeps track of each appointment). For that to occur, you must *create* a job from the Callsheet (see next section). As part of the process when you do so, ServiceDesk will read the notation in the Callsheet's 'Date and Time' box, and create a corresponding entry in the ScheduleList for you.

The second context (this one for making *subsequent* appointments) occurs downstream, as it were, perhaps some number of days after you've gone through the process just described in regard to creating the job and its initial appointment from a Callsheet. At subsequent points, obviously, the originating Callsheet has long since done its job, and so has been retired to the Callsheet Archive; thus, it is no longer a candidate tool for the process. Instead, at this point scheduling is done from the job's JobRecord, using the JobsCurrent form (or, though it's comparatively clumsy, you can do it by making entries directly from the ScheduleList form itself). Regardless, this chapter is about Callsheets, so at this point we'll not further discuss how scheduling is done for appointments *after* the job's been created (see discussion beginning at page 108 for such information).

Instead, at this point we'll simply describe a few details regarding entry of an appointment notation in the 'Date and Time' box of your Callsheet.³⁵

The first thing you'll notice, as you enter the 'Date and Time' box, is that a little calendar pops up above it (we call this the 'DatePicker').³⁶ You can use either your cursor keys (then hit Enter) or click with your mouse to select the day of the appointment. Upon doing so, you'll see that the system enters a notation in the 'Date and Time' box regarding the day in question, then immediately another box pops up, offering a selection

³⁵ Issues related to scheduling are discussed at several points in this manual, as contextually relevant. This discussion, for example, concerns the basic processes by which you may place an appointment notation into the 'Date & Time' box of a Callsheet. Later there's an entire chapter devoted to matters of Scheduling and Dispatch (Chapter 6); in particular there, consider a discussion beginning on page 111 that addresses methods by which you may resolve a few outside-of-the-norm issues in your appointment notations. And there's a discussion (as already referenced in the text) on creating appointments in the context of an already existing job, from its JobRecord (as opposed to doing so while *initiating* a job from a Callsheet, as discussed here); that begins on page 123. Finally, if you're interested in the underlying technical details by which ServiceDesk interprets appointment notations (and what the requirements strictly are, in terms of format), there's an available discussion in the Technical section of the Appendix, beginning at page 307.

³⁶ If you wish ever to *force* an appearance of the DatePicker or TimeSelector controls (i.e., say you're already in the Callsheet's 'Date & Time' box, the DatePicker appeared automatically, but you didn't want it at the moment so hit Esc to get rid of it, then a few moments later you want it to reappear), you can simply hit **F1** or **right-click** within the Callsheet's 'Date & Time' box.

of time-frames (or appointment windows) you may select from. Again, you may either click with your mouse or use the cursor keys and hit Enter to make your selection. Alternatively (especially if none of the offered time-frames fit your immediate need), you may simply type-in the time or time-frame for which you are scheduling the customer. Regardless, you'll see the process is eminently easy.

Of course, there's "easy" and "easier." In most cases of scheduling, you will have used the Callsheet's ItemLocate feature (see page 44) to view the job's location from your DispatchMap. It's likely *while there*, moreover (viewing various dynamics in your existing schedule) that you will have agreed with your customer on a suitable day and time for the appointment. Since that is the context where you're deciding, why not indicate your intent then and there—and allow ServiceDesk to enter an appropriate notation in the Callsheet for you? In fact, this is almost too easy. When in the DispatchMap having arrived via an ItemLocate request, and when viewing the day for which you want to schedule the person, simply **left-click**³⁷ on the item's graphic reference, as flagged in red. In response, you'll immediately see a combination entry/list box. You may either select a time-frame as offered in the list or type-in a different time or time-frame as wanted.³⁸ The system will assume that the day displayed within the map is the day wanted, and create an appropriate notation in the Callsheet for you. Wala! That's all there is to it.

Always bear in mind, however, that an appointment notation within the Callsheet is not the same as an entry in the ScheduleList. In and of itself, the former (a mere notation in the Callsheet) has no operative effect. It becomes operative only when you create a job from the Callsheet, at which point ServiceDesk copies that appointment notation (along with info referencing the job itself, which tech is assigned, and so on) into the ScheduleList for you. There, *within the ScheduleList*, an entry as thus created *does* have operative effect (meaning the job will be displayed in your DispatchMap, and so on).

In summary, please remember these points. For an appointment that's made at the same time you're creating a job from its initiating Callsheet, simply place an appropriate notation in the Callsheet's 'Date and Time' box. You can place it there using helpful tools that are attached to the box itself (which will automatically appear at appropriate moments), or by signifying your appointment from the ItemLocate context in the DispatchMap, or even by typing manually. Regardless, when you then create a job from that Callsheet, an appropriate (and operative) appointment entry will be inserted to the ScheduleList for you.

J. Creating a Job/Sale

The purpose of the Callsheets, primarily, is to take down information from incoming callers, document the process, serve as a work-space for follow-through, and so on. Of course, you'll hope that a significant percentage of your callers are placing orders for service. Or perhaps you have someone at your sales counter buying a part. In either case, the relevant information first enters ServiceDesk through a Callsheet. These are the portal, you might say, for initiating every job or counter sale.³⁹

³⁷ If you want to select a day for the appointment that is so far removed from the present as to make *paging* to that day (using your keyboard's PgDn key) somewhat inefficient (suppose, for example, you want to make an appointment for five weeks hence), you may use the calendar function from within the DispatchMap. Just **right-click** on the on the job's red-flagged locational reference, rather than left-clicking (or, you can press 'C' on your keyboard), then select the day as wanted from the little calendar that appears.

³⁸ If you don't like the set of time-frames that are offered in this list, incidentally, you can easily make a different list. To do so, see the instructions provided at note 67 on page 108.

³⁹ Since most of our clients are more heavily involved in performing service calls than they are in counter sales, we'll more often refer, in this context, only to the former. However, you can essentially think of every counter sale as kind of a mini-job. Aside from the fact that a counter sale involves no appointments or job history, it can otherwise be handled in exactly the same way as a job.

So what distinguishes a mere Callsheet (say one on which you've only taken and returned someone's call) from one that's used for an actual job or sale creation?

The secret lies in what you *do* with the Callsheet. If you wish to use it (and the information you've already entered upon it) to create an actual job or immediate sale, your task is simply to click on the Callsheet's option button that, appropriately, is labeled 'Job/Sale'. Or you can press **Alt-J**.

In either case, you'll immediately be presented with ServiceDesk's *Create Job/Sale* form. Though small, this form offers several options, most of which are self-explanatory. Typically, you'll want to leave most everything in its default state, and simply hit Enter or click on the indicated 'Okay' button. Now, ServiceDesk will do several things, all in the blink of an eye. These tasks are:

- (1) it prints an invoice for the job or sale; (2) it creates a JobRecord under which, assuming it's a job rather than mere counter sale, all further work and progress on the job will be managed and tracked; (3) if the job involves an already on-the-Callsheet-scheduled appointment, it enters that appointment into your ScheduleList; and (4) the originating Callsheet's status is changed to show a job/sale has been created from it and that its task is thus done (accordingly, it goes dim and is ready for movement into the Callsheet Archive).

Of course, there are several details you may wish to know about.

For example, as the Create Job/Sale form displays, you'll notice you're immediately placed into a box labeled 'Assigned Tech.' The reason for this box: if you know at the time who you want as the technician for the job (or simply have a pretty good idea), you might as well make the assignment presently (just enter his initials). While you can wait and make the assignment later from your DispatchMap, it's nice to have it done from the start (assignments can be re-juggled later, regardless). Also, you'll notice there are options, immediately to the right, marked 'Tentative' or 'Definite' (it defaults to the former). The purpose: if the job's a recall, special request from a customer, or for any other reason definitely needs to go to a particular tech, you can so specify here (you can click on the 'Definite' option with your mouse, but it's even faster, after entering the tech's initials, to press your keyboard's right cursor key then the down key, as this will highlight and select the 'Definite' option without any movement of your hands from the keyboard).

If you've enabled the option in ServiceDesk that allows you to separate your sales among various company divisions or departments (see page 172), you'll also notice there's a box in which you must select the category to which the job or sale will belong (absent enabling this option, you'll see no such box, and needn't worry about this subject). Just click on the appropriate department title, or tab over to the box and use cursor keys to highlight the appropriate department, then press Enter to select.

At the top of the Create Job/Sale form, you'll also notice a box with several options marked 'Action Type.' Basically, the default option is to both print an invoice and create a job. However, you can instead specify that you want to create a job without also printing an invoice (handy if you're dispatching the job by telephone to a technician in the field, and he's going to hand-print one from his location, so no additional physical invoice needs printed from your office). Or, you can just print information from the Callsheet onto paper (creating kind of a pseudo invoice, sans genuine InvoiceNumber) without creating a job or sale at all. Finally, if a job/sale was already created from the Callsheet, yet its status somehow got changed to indicate otherwise, you can simply specify that you want its status changed back. Just click on whichever option you prefer (usually it should be the first, default option).

You'll also notice that the Create Job/Sale form specifies the InvoiceNumber ServiceDesk will assign to the job/sale in question. If you've not yet specified a beginning number from the Settings form, for this

purpose, you'll be urged at this point to do so. ServiceDesk tracks each of the numbers used, and will assign the next number, in sequence, to each subsequent job/sale. It is possible, if you wish, to change the number assigned in any specific instance (this does not change the number ServiceDesk tracks as the next one properly assigned). However, there should seldom (if ever) be any such need. In general, we caution against changing the number proposed if in fact you're creating a job, for serious difficulties can arise if out-of-sequence numbers are assigned to items entered in the JobsCurrent file.⁴⁰ Similarly, you'll note ServiceDesk displays the CreationDate that will be printed to your invoice. You can also change this, should the odd need arise.

Bear in mind, in regard to InvoiceNumbers, ServiceDesk prints them to the invoice for you. Thus, there is no need to have them advance-stamped to the invoice by your printing company. This saves money, and avoids the necessity of having to verify, with each invoice printing event, that the invoice in your printer is stamped to the same number as ServiceDesk is prepared to print. In the future, please be sure to order your invoices with a blank space for the number, but no number actually stamped in.

You'll find, both as you move from a Callsheet to the Create Job/Sale form and as you execute the form's 'Okay' button, ServiceDesk checks several elements to make sure all is in proper order for creating a job (at least everything it's capable of checking, such as that an appointment is in proper format, for example). A particularly helpful check, done in the background so you're not even aware of it, is to make sure you are not printing a new job with the same P.O. Number (typically involving dispatches from a home-warranty type of client) as was already used on a previous order. If ServiceDesk finds you're re-using such a number, it will bring it to your attention and suggest you reconsider your course.

Another useful check is that ServiceDesk examines your still pending Callsheets to make sure you're not printing from a new one, when in fact you should be printing from an earlier one. Specifically, we've noted that occasionally there will be a home-warranty type of job, on which we already have a Callsheet in connection with which we've been trying to schedule the customer. Eventually, the customer calls back to schedule, but doesn't mention they're doing so under that pending home-warranty dispatch. Formerly, without realizing the connection, we'd sometimes write-up such a person on a new Callsheet, assuming they were a cash customer (while in fact there was still that earlier, home-warranty dispatch pending). This additional check, by ServiceDesk, now warns before such a mistake occurs.⁴¹

In regard to actually printing your invoice, please bear in mind that nearly every servicer has his or her own, preferred invoice layout. In creating ServiceDesk, we've not wanted to make you, as the user, bend to

⁴⁰The problem is that, from many contexts, ServiceDesk locates items in the JobsCurrent file on the basis of InvoiceNumber sequence (this enables it to find items rapidly). If there are any items out-of-sequence, ServiceDesk may disastrously fail to locate them (much as you would fail if looking for page 59 in a book, when it happened to be placed between pages 97 and 98). Even so, there may on rare occasion be circumstances where you'd want to specify an out-of-sequence number. Suppose, for example, a JobRecord is erroneously deleted from the JobsCurrent file, and you need to get it back there to process it to completion. You might re-create it, with the same invoice number as was originally assigned, via the process we're here warning against. There'd be little problem if you kept the resulting JobRecord (out-of-sequence in terms of its number) within the JobsCurrent file only so long as necessary (hopefully not more than a few minutes) to restore its history and other details, quickly process it through to completion, then move it to the JobsArchived file (by running the JobsCurrent forms's Archive routine). On the other hand, suppose you want to create an invoice without any actual, accompanying JobRecord, and for some particular, out-of-sequence invoice number (and perhaps an odd date too). Simply put all the information into a Callsheet as *though* you were creating job. Then, click Alt-J and, from the JobCreation form, specify the parameters (i.e., invoice number, etc.) wanted. And, in this case, be sure to use the option indicating that you want to "print-only," and do not want to create a job. With this procedure, you needn't worry about an out-of-sequence number, because no JobRecord is being created.

⁴¹This feature points up another reason why it's important to assure that each of your large clients (i.e., one's who *frequently* dispatch work to you) are properly designated, in the QuickEntries form, with an HVC-Abbreviation. When this feature conducts its automatic search within Callsheets (to warn if the presently-being-printed one has a substantial duplicate), it knows to disregard any matches that involve a HVC-designated name, address or telephone number. If you have multiple Callsheet entries pending for any such client, yet have *not* properly designated a HVC-Abbreviation for them, ServiceDesk will treat each such entry as a match, and *needlessly* warn accordingly. Thus, it's important (for this reason, among many others) that the designation is properly made.

any particular design in this regard. Thus, we've setup the system so that ServiceDesk will print job-initiating work-order/invoice information onto whatever format you specify. There are instructions in the Appendix that describe how to setup for your own particular layout (see page 269). Of course, you *can* simply continue in using the default format that's provided for you (primarily we provide it just to give you an easy and quick start), but for most operations we don't think it will be most ideal. The biggest reason is because that default format is designed to print both *form image* and job-text onto previously blank paper, and we think generally it's much more efficient to use invoices where the form-image has already been printed-in for you by a local printing company.

In regard to the *process* of creating the job and printing its invoice, there's one more matter to note. One of our clients wanted to equip their technicians with a supplementary document that would outline the history of past jobs at the same location. It seemed like a good idea, so we've provided a feature to make it convenient (the printed information will consist of the history of prior jobs, data that you'll accumulate in ServiceDesk regardless). We mention it here because, if this is something you want to do, it probably makes sense to print these histories at the same time you print each new, job-initiating invoice.

The recommended procedure for this purpose is as follows. After printing the invoice, press F12 (thus bringing up the TechInterface form in CstmrDbase search mode, see page 209), then type the first several characters of the address in question. If matches come up, you can click on one, view it, and view adjoining ones simply by using the up or down cursor keys. If there's history you deem significant enough to send with your technician, press 'P' or 'Alt-P' to print it. ServiceDesk will remind you to be sure you've shifted to plain (i.e., non-invoice) paper, and on your consent will print the history of each item thus requested.

A final matter concerns the ability to re-print an invoice after the job-creating process has already been run. Perhaps, for example, your printer jammed, or had the wrong kind of paper in it, or suffered some similar problem when the process was first run—so you didn't get a good paper invoice, but have in fact created the job (i.e., a unique InvoiceNumber was pulled and used, a JobRecord was created, etc.). Or, perhaps you dispatched a job to one tech, and he's got the physical invoice, yet calls in that he can't make it. Thus, you need a new copy to give to a different tech. Regardless, do not ever attempt print an existing job's invoice, once the job has been created, by running through the job-creating process again from its originating Callsheet. Instead, any new efforts to print the invoice should be made from its now existing record in the JobsCurrent form (press F7). There is a utility there provided for the purpose. Simply bring up the record wanted, then click on the command button labeled 'Print Copy' or press Alt-P. Ironically, the correct method is also the easy one.

Otherwise, if you do it the wrong way, you'll essentially be creating a new, duplicate of the original job (i.e., a new and different JobRecord, InvoiceNumber, etc.). We've found by experience that neophytes tend to make this mistake unless warned, so (you may find if you personally make such a mistake) ServiceDesk does it's best to steer the errant user away from such a course.

K. Miscellaneous Callsheet Utilities

There are several tools to aid in the manipulation of text on a Callsheet . First, you can use any of the normal Windows editing tools, which are as follows:

Cut	Ctrl-X	removes highlighted text into the Windows clipboard ⁴²
Copy	Ctrl-C	copies highlighted text into the Windows clipboard
Paste	Ctrl-V	inserts clipboard contents at your cursor location
Undo	Ctrl-Z	restores text, in your current box, to its pre-edit form

Suppose, for example, you want to move a particular section of text from one place to another within your Callsheet (or even from one form to another). Just highlight the characters using your mouse, then press Ctrl-X, at which moment you'll see them disappear (don't worry, they're now in the Windows clipboard). Now, move your cursor to the place you want to insert and press Ctrl-V, at which point you'll see the characters pop into their new location. The other Windows tools work in a similar fashion, though each with a somewhat different result as described above.

Besides the basic Windows tools, ServiceDesk provides several of its own tricks, that are specifically tailored to your needs in a Callsheet. These are:

Undo	Alt-U	restores the entire Callsheet to content that existed at last save
Erase	Alt-E	erases all text from a Callsheet
FlipFlop	Alt-F	moves the LocationInfo text into the CustomerInfo section, and vice versa
Telephone FlipFlop	Alt-T	moves the telephone number of the box you're in into the other box of the same section, and vice versa
Originate	Alt-O	resets a Callsheet's origin info to the present desk, date and time

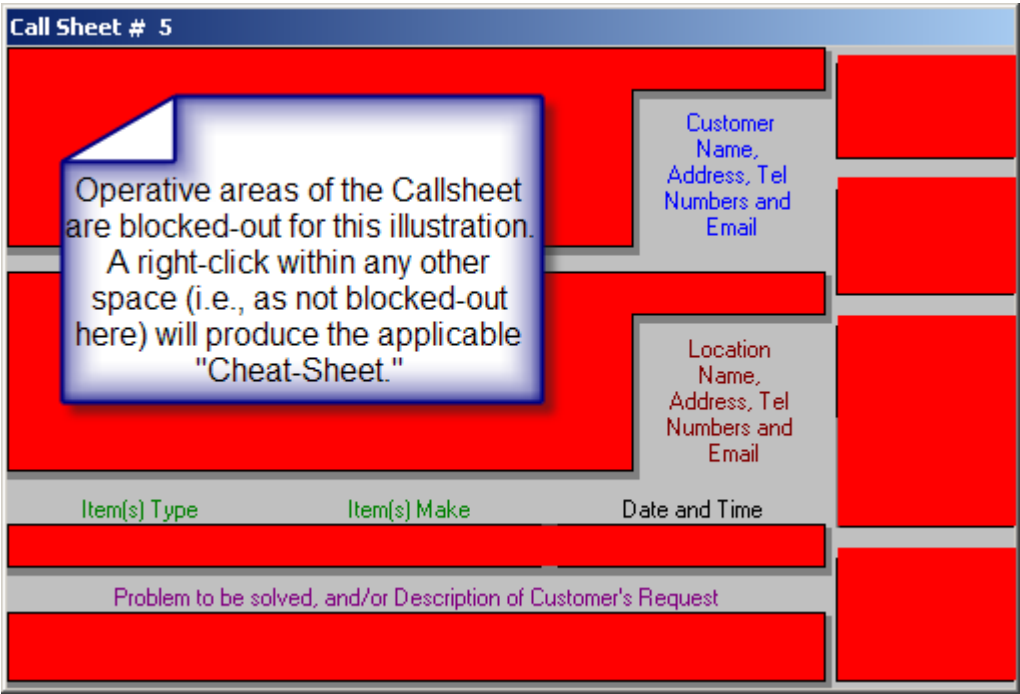
To illustrate some possible uses for these tools, suppose that a landlady is calling to order service at her rental. She's just given you an address which, as she gave it, you thought was for billing, so you dutifully typed it into the CustomerInfo section. Then, she continues, "and my billing address is . . ."—revealing you were mistaken about her earlier intent. Now you've got the location address already typed where the billing address needs to be, and in fact, she's presently dictating the latter to you. Where can you type it? No problem, just type the billing info into the vacant location space you're already in, and when done hit Alt-F, immediately reversing the two into their correct locations. Telephone FlipFlop can be used for an almost identical purpose (only in the context, obviously, of a home telephone versus business number).

As you're reading this, we reckon you might be feeling that all these different commands are a lot to remember. Couldn't we have designed it, instead, so there are obvious buttons right on the face of the form for each purpose—so you don't have to remember? The answer is, there's only so much you can present visually in a limited space, without crowding out other important stuff, and without making the whole thing appear too busy and complex. As in everything else in life, it's best when there's balance—in this case between what's visual and mouse-click accessible versus increased utility via other actions that are just too many to be presentable in such fashion.

⁴²In case you do not know, the *clipboard* is an invisible space, within Windows, where you may temporarily put text or images, and retrieve them later (in the same or different context) if wanted. Whatever you put there will stay, and be available from *any* Windows context, until you either replace it with something else or terminate your Windows session.

But there is a very handy aid. In other contexts, we'll tell you about a very nice command summary, available on the MainMenu. This summarizes each of the various keyboard commands, in a very nicely organized hierarchy, making it easy for you to review while learning, or to consult simply to refresh your memory later on. Within this command summary, there's a very detailed section that relates to Callsheet commands. If wanting to remind yourself of the command for some Callsheet function, you *could* click on Command Summary from the MainMenu, then choose Callsheet Controls, and you'd see this section. However, that's a tiny bit of work.

Instead, if you simply **right-click** in any otherwise un-operative space of a Callsheet, the system will instantly present the appropriate section from the command summary for you. Think of the resulting show as your "cheat sheet."



As you'll see later, the same kind of thing is available from several other contexts.

L. Callsheet Numbering, Saves, Updates, and Name Interpretation

In addition to these utilities, there are some helpful things to know about how ServiceDesk manages a few elements of housekeeping.

First, you'll notice all the Callsheets, existing on your system at any particular time, are sequentially numbered. The purpose of this numbering is to provide you with sign posts, so to speak, so you can easily tell how many you have at a given time, and where you are within that set. It's also so that when you want to draw a coworker's attention to a particular Callsheet, you can easily reference it via that number (i.e., I might say to my secretary Brenda: "Mrs. Zelipsky on Callsheet 23 needs to be scheduled). The thing to remember is that this sequential numbering, of working-area Callsheets, is nothing more than that. The numbers are not, by any

means, a permanent ID assignment for such Callsheets—as are, by contrast, the InvoiceNumbers that are assigned to actual jobs.

Second, when making changes in a Callsheet, it may sometimes be useful to know they are not continuously recorded (i.e., "saved") to your hard drive. However, 'saves' happen with some frequency, as specifically triggered by particular events, including: your own action in moving from an edited Callsheet (i.e., into another form, including any other Callsheet), your action in switching a Callsheet from one desk to another, changing its status, and so on. Of course the work is done in the background, and you needn't think about it. And please be assured these automatic 'saves' are frequent enough that, even if you had a system crash, you'd be unlikely to lose anything more than what you were typing in the preceding few seconds.

Moreover, even if you've done nothing inadvertent to trigger a Callsheet save (in terms of moving to another form, etc.), ServiceDesk automatically invokes an unseen, 30-second timer after any editing activity (i.e., typing in a Callsheet, deleting text, etc.). If the timer counts down to zero and there has not otherwise been a save, ServiceDesk invokes one on its own.

And further still, if wanted you can purposely invoke an immediate save, in any Callsheet that's had changes, simply by hitting Esc. As you do so, you'll notice that the normally teal-colored background (the area between the Callsheets) momentarily turns to a blue color. This is confirmation that your Callsheet is saving. As a matter of fact, this momentary change in color occurs during any Callsheet-save event, whether triggered by the manual Esc or automatically.

Something you'll quickly notice if running with multiple stations is that ServiceDesk is designed to allow the use of only one set of active Callsheets network-wide. In other words, the entire set of active, not-yet-archived Callsheets is the same for every station in the network, so from any desk the same Callsheet information is available and displayed. The difference between each desk is that if mine is the desk that a particular 'Current' status Callsheet happens to be assigned to, then it is on my desk alone that such a Callsheet will be illuminated, while on all others, it will remain dimmed (and those others will also be locked out from editing my Callsheet). Thus, while everyone has the same information, it's obvious to each which tasks they are immediately responsible for.

Since all stations constantly display the same set of Callsheets, you might wonder how is it when one station makes a change that others are updated (i.e., when one station edits an existing Callsheet, adds a new one, changes status, etc.). In particular, you might wonder what happens when two stations are both in the same first-vacant-Callsheet position, typing in their own separate info to create their own independent new Callsheets. In such a circumstance, both stations are essentially claiming the same position, on the same Callsheet page, for their own new Callsheets. How is this handled? Don't worry. ServiceDesk takes care of it, and does not require that you pay such matters any attention.

However, and even so, in the event you encounter situations like this, it may help if you understand a few quirks in operation, in terms of how ServiceDesk is juggling each user's demands. Basically, it follows these rules.

- a. If your station is not on the same Callsheet page as the one on which another station has just saved a change, no reaction is needed (the updated information will be received, in a matter of normal course, whenever you move to that page).
- b. If your station is on the same Callsheet page as the one on which another station has just saved a change, your station will immediately read the newly saved Callsheet information, and display it.

- c. If both you and another station are simultaneously making new entries in the same first-vacant-Callsheet position (which by definition must be on the same page), the station which saves first will claim that position for its new Callsheet. The other station's current position will be moved, together with whatever text has been entered, into the next first-vacant-Callsheet position.

The result of the above protocols⁴³ is that, from your perspective as user, there's virtually nothing to worry about. The only time anything seems even unusual, within ServiceDesk,⁴⁴ is in the case described for situation 'c.' In this case, you may be merrily typing information into a new Callsheet when, suddenly, because another station has just saved their own new Callsheet in the same position, you find yourself being unwillingly transported to the next vacant spot. The event could seem disconcerting if you did not understand what was happening. However, with the explanation here provided, it should not cause alarm.⁴⁵

Occasionally, you may enter information into a Callsheet that you want to immediately display to other stations, even while you have no purpose for otherwise engaging in an action that would create a save. Again, the Callsheet-Esc method comes in handy. Just press that single key from your Callsheet, thereby inducing an immediate save, and thus causing instant updates to other computers.

Another matter of note concerns ServiceDesk's treatment of text in a Callsheet's two name fields (i.e., CustomerName and LocationName). The underlying reality here is that, as a user, you might be entering either a business or person's name—and the treatment in certain contexts needs to be different, depending on which is involved. If it's a person's name, for example, you'll want only the last name applied to short-references in the DispatchMap, to the "Dear So and So" in future billing reminders, and so on, while if it's a business name, the entire name will (in most cases at least) be used as a unit. Thus, ServiceDesk needs a means to distinguish between the two situations, and since it still lacks full human intelligence, you need to provide a little assistance. All you need to do is follow this convention: any time it's a person's name, use the format of LastName-comma-space-FirstName (e.g., "BOYLES, JOHN"). Finding the two words separated by a comma, ServiceDesk will figure it's dealing with a person's name, and will separate out the last name as that. In the absence of this format (i.e., no comma in the line), ServiceDesk figures it's dealing with a business name, and treats it accordingly (see note beginning at page 293).

As still another matter regarding your typing of info into Callsheets, you might note that ServiceDesk automatically engages the caps lock on your keyboard whenever you enter a Callsheet. The reason is because there is little point here in slowing down your typing by requiring the use of both upper and lower

⁴³ Another protocol concerns updating to another station's Archive event. This is discussed separately (see page 86).

⁴⁴ Okay, we'll admit it: there is something that may sometimes seem unusual *outside* of ServiceDesk. Specifically, if you have ServiceDesk running in the background, and are simultaneously working in a different application while someone else works from another station within their Callsheets, you may find that sometimes ServiceDesk pops suddenly back into the foreground on your screen. This can be annoying. It happens because, as part of updating a Callsheet, within your own ServiceDesk application, to changes made by another user, it is sometimes necessary for ServiceDesk to change the particular control, upon a Callsheet, that has the "focus." Because of limitations in the Visual Basic programming environment that ServiceDesk is written within, it's impossible to so change the focus without *grabbing it* (the overall "focus," that is), from any other application that may be running. That's what makes ServiceDesk pop to the front. It's stupid, but we've not been able to find any way around the limitation. At least, however, we can suggest one or both of two accommodations if this should happen to you. First, if you switch the page of Callsheets that you have displayed at your desk to the very first page (use Ctrl-PgUp to get there instantly), it's much less likely for the event to occur. The reason is because, generally, the first page of Callsheets consists of old stuff on which another user is unlikely to be doing much work. Second, if you simply *minimize* your running ServiceDesk application to the Windows Taskbar, the situation will not arise regardless.

⁴⁵ Even so, there have been occasions when I was taking my time in a new Callsheet servicing some customer, while my secretary was taking calls and adding her own new Callsheets in rapid-fire succession, with each such event bumping me continually from one location to the next—to the point of some annoyance. One day, I finally realized that to prevent this, I needed only to do the easy Callsheet-Esc trick, thus saving my Callsheet instantly, even if only in its present, not-yet complete form. Thus, my fixed spot is now secured, to be bumped no more, while I continue collecting information (or working out an appointment time, etc.) from my customer.

case. The expected convention, therefore, is that unless there's a specific reason to do otherwise, all your typing in a Callsheet will be in upper case. This simplifies the typing, and provides for a nice uniformity of output. You'll find the invoices look just fine in such format too.

Oh, and incidentally, in regard to the four telephone number fields that are provided in each Callsheet (two in the Customer-Info section and two in the Location-Info section): it's our convention that home telephone number is placed in the left box and business telephone number in the right. You can certainly do otherwise (and attach appropriate notations to indicate pager numbers, etc.) but we find as a basic convention, this works well.

M. Archiving Callsheets

Like pages out of a note pad, individual Callsheets will eventually serve their purpose (actually, many will fulfill their purpose almost instantly, as you take the info for and schedule a service call, then immediately create a job from it). Naturally, you don't want your workspace cluttered with Callsheets that, if they were paper, you'd be ready to throw into the waste basket (or perhaps into a stack of completed call-records). So obviously, you need some means to readily cleanse your Callsheet workspace of all but items that will remind you of work that still needs done.

To accomplish this, you may either select the '*Archive Callsheets*' option from the main menu or press **Alt-A** from the Callsheets themselves (as is, in fact, suggested by the menu option, in case you forget). The archiving process looks at each of the Callsheets in your current workspace. Any that are marked 'Job/Sale' or 'Otherwise Done' are moved into the CallsheetArchive, a separate file where they remain available for later search and review should the need arise (if you've ever had the need to go over old call-records, you can imagine how this might be useful). Those marked 'Delete,' obviously, are simply erased. All others remain in the current work area.

You can run this archive routine as often as you feel the need to clear your workspace—probably at least once a day, perhaps several times, depending on the volume of your incoming calls and personal preference. Some operators like to wait and see how many Callsheets have accumulated after a full day's work, then they archive. Others like to keep the workspace as uncluttered as possible, and so archive frequently throughout the day.⁴⁶

You might note, incidentally, that there are several other forms in ServiceDesk that occasionally require an archiving "flush" of non-current data from their active workspace; these are discussed in their own sections. You might also note there is an Auto-Archive feature that, if turned on, will automatically run archive routines on all of these, once each night (see page 209).

⁴⁶ Bear in mind that, by removing completed items, the Archive routine changes the overall set of Callsheets that are displayed on all desks, network-wide. Obviously ServiceDesk must force an immediate update on all the other stations whenever such an event occurs—which invites difficulties if at the same time any station is actively engaged in editing one. Because of the archive event that Callsheet may suddenly be found in a different place, in terms of pagination, than it formerly occupied (preceding completed Callsheets having been removed), and there is no way for ServiceDesk to make an advance prediction as to where. For this reason ServiceDesk cannot preserve a secondary station's unsaved Callsheet edits when another station runs the Archive routine. It *will* display a warning to the secondary user explaining that another station has archived, cautioning that unsaved edits are about to be lost, and allowing a one-time chance to write that information down separately (to be re-entered after the update). But even so, as a courtesy to other users, it's a good idea to make sure none of them are in the midst of making unsaved Callsheet edits at the time you run an Archive routine.

If you ever want to view or conduct searches on (or just browse among) your Archived Callsheets, it's very easy. Just press **Ctrl-F2** (or click on the corresponding MainMenu command button). This resource can be quite handy. You might want to lookup information on a Callsheet that was archived some time ago. Searches are easy. Just press 'S' or click on the indicated command button, then type your target (any leading portion is adequate), and hit Enter. To resume the same search after the first or any subsequent target is found, just hit 'S' and press Enter again (the target specified remains in memory). Press Esc to return to your current Callsheets.

Chapter 6

SCHEDULE AND DISPATCH MANAGEMENT

Aside from the Callsheets themselves, your most favored venue in ServiceDesk will undoubtedly be at what might be called the dispatcher's "Command Central," a place where you can see and manipulate every important detail of every day's schedule, for all your technicians, and at a mere glance.

Everything in regard to dispatch revolves around the set of jobs you are scheduling for a given day. This list of scheduled appointments, then, is the one item of varying information you must constantly reckon with, and it is in regard to how items in this list fit, with respect to the matrix of time, geography and the peculiarities of your own technical staff, that your every dispatching decision must focus.

That, essentially, is the subject of this chapter—in other words, how to manage the set of scheduled appointments that, within ServiceDesk, we simply call the "ScheduleList." But to clarify, we'll not here deal with everything related to the subject. In fact, while we'll discuss the various means for managing the list *once it's established*, we'll not discuss the particular *processes* by which the list's entries are first created and inserted into it. Instead, those discussions occur contextually in chapters that otherwise describe the processes of which they are a part. Specifically, for appointment-making in conjunction with creating *new* jobs, you'll find an applicable discussion in the preceding chapter on call management (page 70). For similar process but in conjunction with management of *existing* jobs, there's an applicable discussion in the following chapter on job management (page 108). Plus there's a relevant technical discussion in the appendix (page 283). We mention it because you should know these three other discussions, while containing information that might arguably fall under this chapter's umbrella, are in fact located elsewhere.

Hint: If using the video tutorials, please read this chapter in conjunction with having watched Lesson # 3.

A. The DispatchMap

ServiceDesk provides two different means for viewing and manipulating your ScheduleList. The first (to be discussed in this section) is intensely *graphic*, showing each item from the schedule in its precise relationship of time and geography, and allowing manipulation of each item as needed from such a standpoint. The second (to be discussed in the following section) is *textual* instead, and allows the manipulation, obviously, of textual details. These will now be discussed, in such order.

i. General Elements and Considerations in Map Design

You may note, as you briefly examine your DispatchMap, it's is customized to show the boundaries or your own unique service area. Yet it does not include a lot of geographic detail. Depending on what seemed suitable to the designer in your case, it may consist of nothing more than red lines representing the major highways, or black lines showing border areas, blue lines showing coastlines and major lakes, textual notations for the larger cities, and a simple red dot for your service location. There is a reason for this economy. A dispatcher needs an instant and broad overview of the entire service area, showing in one glance the relative location and times of all jobs scheduled for a given day. He or she does not, on the other hand, need a detailed display of city streets. Nor could such a detailed display be shown (with today's technology, at least) while also providing that all-important, broad overview. Naturally, ServiceDesk provides the former.

One detail you may notice (depending on your own map) are lines defining odd-shaped cutouts or divisions between otherwise open areas within your map. In those service territories that encompass mountains, rivers, or other large zones with no roadways across, it will sometimes happen that housing developments lie geographically close along either side of such obstructions (i.e., a very short distance apart as the crow flies), but that driving between such locations involves much distance because of the need to go around the intervening obstruction. By representing these non-traversable areas on your map, ServiceDesk allows you to see in an instant that two otherwise close locations may not, in terms of driving, be close at all. More specifically, you can see that your technician must drive around the indicated obstruction to get from one location to the other.

Where possible, your entire map will be included on a single screen position, or at least on as few screen positions as feasible. Where there are multiple screen positions, you may move between them (this is called "panning") using your cursor keys.⁴⁷ If your service area is comparatively tall as compared to wide, you may have two or three successive screen positions arranged along a vertical axis. If your territory is rather more wide than tall, it may be split into multiple screen positions arranged along a horizontal axis (or perhaps both). In either case, the successive positions will be arranged with substantial overlap, so you should never have a location that cannot be viewed without substantial territory all around. Plus, you can use the map's 'Overview' mode (press and hold the **spacebar**) to view the entire map, reduced as necessary to fit into one screen. This mode is particularly helpful for getting an instant look at how all your jobs layout for a day, though there is less detail in the information provided (versus full-scale mode), and you cannot edit in this mode.⁴⁸

In some cases we've encountered service territories that are so large as to make it possibly less than obvious (while panning in full-scale mode) which portion you are viewing at a given moment. Thus, we've added a feature that allows you to see a miniaturized inset of your map, which internally indicates the portion that's shown on your main display. This can help improve orientation. To use the feature, you must turn it on from within the Settings form (accessed from the main menu or by pressing Ctrl-F1). If you find its presence annoying, turn it back off. Another selectable feature provides range lines, showing distances from your office in (typically) five mile increments. This makes it easy for you to determine, in a glance, how distant any

⁴⁷For a couple of panning shortcuts, see the two paragraphs that describe how to use the 'End' and 'Home' keys on your keyboard, page 94.

⁴⁸Typically, if your company has a large number of technicians, we'll make the map bigger, so there's better resolution within each screen and less clutter for so many jobs, but, on the down side, it requires more panning to view your entire territory. If you've got just one or two techs, we've probably rendered the map somewhat more shrunken, so a greater portion of the entirety can be seen on one screen. Of course, the actual size of your territory is a factor too. Ideally, we'd like to have a one-to-one relationship between the invisible grid positions on the screen (where job references locate graphically) and the actual grid sections within the pages of your map book. For those clients with small territories, we're able to do this. More typically, however, we're forced to go with a ratio (for large territories), sometimes as high as four-to-one (meaning there may be, for example, as many as four contiguous map grid positions (or actually streets, falling within those positions from your paper map) that will show, within your on-screen map, in identical displayed positions). Simply stated, a larger on-screen map suffers less from this loss of resolution, but is harder to see in one glance too.

particular job is from your headquarters (at least as the crow flies). A final selectable feature (again, turn it on or off from the Settings form) causes ServiceDesk to display mileage estimates for each tech's route and for your fleet as a whole, for any given day's schedule as currently setup. We added this in the hope of helping our own secretaries pay more attention to how efficiently they setup the technicians' routes. It provides them with a score, you might say, with which to judge their success or not.

You may be interested to know that, in sizing your on-screen map, we've struggled between the competing objectives of: (1) providing as much detail and resolution in terms of job location as possible; and (2) fitting enough of the relevant territory into a single screen so as to allow effective overview without excessive panning. To a very great extent, balancing these objectives is a matter of judgment.⁴⁹ In this regard, you should know our design decisions (the product of which you now possess) are not set in stone. If you'd like your on-screen map more condensed (so you can view more of it within a single screen), or enlarged (for better, more close-up resolution), it takes only about an hour of work for us to make the conversion. We'll be happy to do it for you free of charge, so long as you make the request within a reasonable time after your purchase date. Other small changes may be made as well. Just ask.

ii. How your map displays Schedule and Related Information

In terms of displaying information from your schedule, you'll find your map shows each of your scheduled appointments in two context: first, as a *graphically* correct representation within the geographic boundaries of the map (each within a tiny rectangle, and with connecting "route-lines" drawn between any of multiples for an assigned tech),⁵⁰ and second, as an item (typically within the screen's peripheral area) within a set of segmented quadrants, one for each tech, showing in *list* format the jobs that are assigned to each. Obviously, these two representations are rendered differently. The graphic representation, because it must serve a purpose similar to that small flagged pin on a wall map, cannot take up much space. For this reason it is printed in the smallest feasible type, and includes only the scheduled customer's name (or snippet thereof) and time scheduled. In the list section, where there's more available space, the type is larger, the customer's name is rendered more fully, and the telephone number and city are also displayed. In either case, where it's a job with separate billing and service locations, a snippet of both names will be displayed, separated by a slash, and in the case of a 'HighVolume'-designated client (see page 60), ServiceDesk will use the abbreviation (thus, an American Home Shield repair at the "Jones" residence would be rendered as "AHS/Jones").

Actually (and at the risk of boring those for whom this is not applicable), we should mention that in some cases (depending simply on what we thought would be most advantageous as we engaged in the design process) we've elected to put all the techs job listings in a section together (rather than tucked into available spaces here and there around the map). Where this is the case, we've likewise made it possible to move to and from this list area with the press of a single key: it's the 'End' key (no, there's no relation between the key's label and this particular use; it was just a handy, available key for the purpose). Press it once to move

⁴⁹Your map's panning positions were designed with the assumption that your monitor is set at the standard resolution of 1024 by 768. However, if you are using a higher resolution, you'll notice that upon displaying the DispatchMap ServiceDesk expands to take appropriate advantage of the larger screen space. This is nice, because it's helpful to see more of your DispatchMap in a single screen view. However (and especially if your resolution is *significantly* higher than 1024 by 768), you may find that the panning positions are not at all optimum. If so, they can easily be modified. The positions are defined via a single line of text in your custom .MAP file. If you're a tinkering type, you can open your file, find the line, and can probably deduce how to modify, to preference. If you want to modify and need some help, just call. .

⁵⁰For a description of how the DispatchMap chooses precisely where to display each geographic reference (and of moderate positional inaccuracies that cannot be avoided), see the technical discussion in the Appendix, page 317.

into the tech's list area (where so setup), and again to move back to whatever space you were previously viewing within the map.

Somewhat similarly (but available as a feature on *every* setup), we've made it possible for you to move instantly from anywhere on the map back to the home location, with the press of a single key, and in this case the name of the key is rather more fitting: logically, it's the '**Home**' key. Press it once to move instantly to the "home" position, and again to move back into whatever viewing space you were viewing previously (this is most useful, obviously, if you have a large dispatch map with many different pan positions).

There are several ways to view your Map. The most simple is to press **F5**. This produces a direct, unadorned display of your map. Alternatives are to use the ItemLocate feature from either a Callsheet or JobsCurrent form, where you right-click on any address within its appropriate text box. In this case your map will display with the selected location flagged in red. The purpose is to show you the relative location of a job you are attempting to schedule.

When first displayed, your map will always show the *present* day's schedule. It's obvious, however, that you might want to see how a job you're considering fits in with jobs scheduled for tomorrow, or perhaps the day after. To view those subsequent days, press PgDn. To return back to earlier days, press PgUp. You can also use the FirstPage/LastPage QuickKeys here. Thus, if you have a date several days hence displayed, and wish to return instantly to the present, press Ctrl-PgUp. Or, if you're looking at some date in the past and wish to return instantly to the present, press Ctrl-PgDn.⁵¹ As still another method of selecting the day displayed, you may simply press '**C**' on your keyboard (for '**C**alendar') and the system will show a small calendar on which you may rapidly select any date wanted.

If you have several technicians, each with full routes simultaneously displayed on your DispatchMap, you may sometimes find that multiple routes end up criss-crossing repeatedly over the same area (depending, of course, on how you've set the routes up). When this is the case, you may find it's difficult (in spite of the distinguishing route-colors) to visually discern one technician's route from another. If this happens, we have an easy solution: we call it the ShowOneTechsRouteOnly mode. To invoke this mode of display, simply **right-click** on the technician's name (in the *list* section, above his list of jobs) whose route you wish to exclusively see. Route lines for the other tech's will then disappear, making the route for this particular one perfectly clear. Hit **Esc** to go back to normal display. (As another visual aid, remember you can press and hold the spacebar, on your keyboard, to shrink the entire display into one screen, for a broad overview of the whole setup in one easy glance.)

It often happens, when you're looking at a particular job's appointment within the DispatchMap, that you want to know more about the job itself. Perhaps you're considering changing the appointment's assignment to a different technician, for example, and are wondering whether the nature of the job is such as to make this practical. Or, maybe you want to look at a description of what needs done in order to estimate how much time will likely be needed. Whatever the impetus, we've provided a very handy means for you to instantly view the JobRecord of any job whose appointment is shown in the DispatchMap. Simply do a **Ctrl/Right-Click**, on either its graphic or list representation. In response, ServiceDesk will immediately display the item's JobRecord, allowing you instant grasp of every detail concerning it (we call this the '*ShowJob*' method). Please do not forget this feature.⁵² It's very, very handy.⁵³

⁵¹ Occasionally you may find results in this context are inconsistent. For an explanation, see the last few paragraphs in the technical Appendix discussion, beginning at page 317.

⁵² Don't worry, on the other hand, about remembering the *particular command* as needed for this or any other feature. As mentioned elsewhere, there's a *contextual command summary* in the DispatchMap (much as in many other forms), which provides a handy "cheat sheet" for whenever you need it (just right-click in any empty space to of the DispatchMap to see its *summary*). You should not need to

A similar feature allows you to instantly view (and/or edit, if wanted) any appointment's reference in full-text format from the ScheduleList form (remember, both of these forms display and manipulate the same, identical list of appointments, it's just that one does it more graphically and the other in a purely textual format). Simply do a **Shift/Right-Click**, from the DispatchMap, on either the appointment's graphic or list representation. This will take you immediately to the item's reference in the ScheduleList form, saving you from having to separately hit F6 to bring up the form, then having to locate the item, there, that you're interested in (we call this the '*ShowText*' method). This is much easier when, from the DispatchMap, you've decided that you want to do some kind of editing, to an appointment reference, that's not available from directly within the DispatchMap. Of course, many kinds of such manipulation regarding appointments *are* available from directly within the DispatchMap itself, and that's the subject to which we now turn.

A final information-conveying element within the map is a feature we call the *Appointment Density Graph*. Stated simply, the purpose of this tool is to visually convey an instant conceptual grasp of how heavily loaded you are, appointment-wise, throughout each hour of the day. Thus, for example, if a quick view of the graph shows you're already heavily loaded for the morning hours, but that you're light in the afternoon, you'll know without further analysis that it may be wise to urge a caller toward scheduling for the afternoon.

To view the Appointment Density Graph, simply press '**D**' on your keyboard while the DispatchMap is displayed. You'll see this tool appear in the screen's lower right-hand corner. Press '**D**' again (or hit Esc) to hide the display. As you'll see, it's a fairly typical graph, but configured to show the overall appointment load for each hour of the day, with the vertical axis indicating number of appointments involved during each hour.

In this regard, the system understands that if you have an appointment scheduled for, say, between 9 and 12, and if your average job takes, say, one hour, that appointment is not likely to consume all of each of the three hours involved in the time frame. In fact, it's likely to take *only about one* of the three hours, but across some (presently) indeterminate span within the range. On the other hand, if you have three 9 to 12 appointments, each expected to consume about one hour, you can figure that, between the three, they're likely to jointly consume all of the three hours, for one man, within that time range.

Anyway, it's with this dynamic in mind that the system is configured to spread out the time-value for any appointment (at present, it assumes one hour; though we may make this subject to user specification at a later date) across the time-range involved – so that it reflects a realistic load-indication on the graph. The result is that if you've got *four* technicians, say, you'll want to shoot for a corresponding appointment-load (of "4" on the vertical axis) across the hours of the day. If you look at the graph and see areas that are significantly above such a level, you'll know you're already extra-tight for those hours. If you see areas that are significantly below, you'll know there's room during such hours for more jobs.

While those are the basic *information-conveying* utilities within the DispatchMap, there's another, separate utility that some firms will want to use for the purpose of facilitating their scheduling activity. We call it the *ZoneScheduler* system. It's especially suited to companies that are having jobs dispatched to them through *ServiceBench* (a web-based, third-party firm that provides information-link services between manufacturers and servicers), or for any other operation that is of such a large size (typically twelve trucks or more) that the standard DispatchMap methods of assessing availability are less than convenient.

even try memorizing these. Just use the "cheat sheet" whenever you need to be reminded. Without any effort at all, you'll soon find yourself remembering without such help.

⁵³To further illustrate, suppose you just did this, to look at the actual JobRecord as applicable to a DispatchMap-displayed appointment. Now you're looking at that JobRecord, and find yourself interested in knowing what jobs you previously did for the same customer. Just hit F1 from this JobRecord, and now you'll see a list of such previous jobs. Now suppose you want to view not just the list but an actual previous job. Hit F1 again (or alternatively use the mouse) and there it is. By a sequence of but *four* simple key-press/click events, you've gone from looking a current appointment in the DispatchMap to reviewing past jobs from the same customer. Wow!

Because the ZoneScheduler system is one that we believe only a small minority of our clients will be using, we've placed the full description, and instructions concerning its usage, in a separate document. You'll find it on your installation CD or in your c:\sd folder under file name "**ZoneSchedulerInstructions.PDF**."

iii. Working in your DispatchMap to make Operational Changes to your Schedule

It's obviously necessary, as you view your map and see how a day's jobs are geographically distributed, to decide which jobs will be dispatched to which technicians. Obviously, your decision on each needs to be communicated in some manner to ServiceDesk —so it can register the assignment. This is accomplished by **left-clicking** on the job's *graphic* representation. In response ServiceDesk will display a little box that shows each tech in your roster. If you've held down the mouse button after clicking it on the job in question, you can simply drag the pointer until the wanted tech is highlighted, then release the button; instantly, that tech is selected. If you did a full click on the job (i.e., pressed the button then released it), you can simply click again on the wanted tech.

That's it, for assigning any job to any given tech within ServiceDesk. It's just that simple. And, naturally, you can re-assign (using identical means) whenever wanted.

Of course, outside circumstance may slightly complicate this picture. As mentioned in another context, for example (see page 72), there are jobs that definitely must, for one reason or another, be done by a particular technician, while others may be done by whichever technician is most convenient for scheduling. You obviously need some manner as you're considering assignments within the map to distinguish between these two situations. ServiceDesk signifies the distinction as follows. 'Definite' assignments are displayed entirely in the color-code for the assigned technician, in both graphic and list representations; for 'tentative' assignments, the second line of the graphic representation remains red, as does the city abbreviation within its list representation. Based on this obvious signification, you can see in an instant, as you're juggling various assignment possibilities in your list of jobs, which ones can properly be switched between various techs, and which cannot.

Of course, it sometimes happens that it's only upon viewing the map that you learn (or realize) that a job should have been designated 'Definite,' but was not (or vice versa). Sometimes, indeed, you may have a job that is properly labeled 'Definite,' but you may be forced by circumstances (e.g., the assigned technician called in sick) to change its assignment regardless. In either case, there is the obvious need to change its assignment status from within the map. To do so, do a **Ctrl-click** on the job's graphic representation. In this case, multiple such clicks will rotate you back and forth between the 'Definite' and 'Tentative' assignment status.

Besides re-assigning, it often happens that you'll want to re-sequence jobs. As you'll soon learn, there's a simple drag-and-drop method for doing this from the *list* section of the DispatchMap, but it can also be done (if somewhat less intuitively) from any job's graphic reference. Just right-click on the reference and you'll see a little mini-list showing each of the assigned tech's assigned jobs for that day. Simply click again within that list at the new position where you want the job moved to, in terms of sequence.

You may notice there are functions for both right and left mouse buttons when clicked on a job's *graphic* representation. Similarly, there are also functions taking advantage of each button when used to click on its *list* representation.

Suppose, for example, you've just talked to a tech at the job and want to call him right back, or you're calling his jobs in the effort to locate him, or you're arranging jobs and want to call a particular customer to see if they don't mind you shifting their time a bit. In any such case, **right-click** on the job's list representation, and ServiceDesk will instantly dial the number for you. It may seem like a small thing, but such convenience feels substantial when the need arises. As a manager, you'll never again have to ask your secretary for such numbers. As a secretary, you can impress your boss by having a relevant customer on the phone almost instantly.

Still another function in the list sections of the map concerns an alternative method for either job re-sequencing⁵⁴ or re-assigning (note we've already discussed methods for doing this from a job's *graphic* representation). If you're working in the *list* section, this is actually much more simple. Basically, you can use your mouse (left button please) to simply drag an appointment reference, either into a different sequential position under the same tech's list or into another tech's list altogether.⁵⁵

Besides these various methods for managing appointments once made, remember that the DispatchMap also serves as a facilitator for getting appointments into your ScheduleList, in the first place. This is via On-Map scheduling, as *initiated* through use of the ItemLocate feature from either a CallSheet or JobRecord (see pages 71 and page 108). As you'll notice, in each case the appointment is still added to the ScheduleList, ultimately, from a context other than the DispatchMap (when later creating a job from the CallSheet to which the DispatchMap's ItemLocate feature has inserted the appointment, in one case, or taking you directly to the ScheduleList, with most of the entry's text already filled-in, in the other).

As a matter of incidental interest, you should know that any time an appointment is added to the ScheduleList (or even a modification saved), the entire list is automatically sorted into a logical sequence based on dates and times. If from the DispatchMap you want ever to manually invoke this process (suppose you've place a number of jobs out of proper sequence and want in one simple operation to get them back, for example), simply press **Alt-S**.

iv. The Check-Off / Status System

To be maximally useful, your DispatchMap must visually convey not only each of your appointments, their times, locations and the techs to whom they are assigned; it must also be willing to indicate the status of each such job. In other words, it should be able to show whether a job has actually been dispatched to the tech, whether the tech has arrived at the job and whether he's finished there, etc. It should even indicate whether the expected PostVisitReport (see page 110) has yet been filed. All these needs are handily met via visual indicators on the DispatchMap itself.

The most critical such need is for you to keep track of which jobs you've actually dispatched to the assigned tech (i.e., you've conveyed the information to him in some manner, so he's knows it's a job he's supposed to do), and which you have not. To understand how this works, suppose yours is an office where the technicians come into the office each morning, and each receives a stack of tickets (or invoices), which constitutes his then pending set of assignments. As you give any particular technician his stack, hit F5 to bring

⁵⁴In regard to re-sequencing, please bear in mind that, unless set otherwise, the system will automatically sequence appointments according to ordinary time values (i.e., an 8-11 will be placed ahead of a 9-12). For this reason, manual sequencing becomes relevant only if you disable the default sorting, or if you're changing the sequence of jobs that have an equal time value. In regard to disabling the automatic sort/sequencing function, there are a few alternatives. For a description, see page 112

⁵⁵You'll likely note while dragging that the dragged item tends momentarily to erase portions of the display over which it is dragged. This is normal for how the system is setup. It could easily be designed otherwise, but only at the expense of creating a less snappy display of the map image. With a choice between snappy display versus momentarily erasing some of the image, we opt for the former.

up the ServiceDesk DispatchMap. Go to the TechList area, and look at the list of jobs for the technician involved. Confirm that the first item in the list is the same as the first invoice in the stack you're about to hand him. Now, using your mouse, do a **Shift/Left-Click** on that listing. This will cause a *check mark* to appear just to the left of the listing, which is the visual confirmation that item has actually been given to the tech. Proceed to the next invoice and do the same.

When you're done, you will have handed the technician a stack of invoices that conforms perfectly with the list under his name in the DispatchMap, and all items in that particular list will have a check mark next to them—the visual, at-a-glance confirmation (for anyone in your office who happens to check) that these particular jobs have indeed been given to the technician.

If you use any of the automated methods for dispatching instead, by the way (see section following), the system will automatically volunteer to do this checking off for you as part of the automated sequence. If you add one or two jobs to a technician's schedule during the day, the absence of a check mark next to that appointment's reference (until you've actually conveyed the information to him and checked it off) will serve as a visual reminder that the task still needs to be done.

All in all, this *checking off* of the fact that a job has actually been given to the involved technician is very important (unless, of course, yours is a one-man shop). We highly encourage you to use it.

And then there's more—that you may optionally want to use.

If, in addition to keeping track of whether each of a day's jobs have actually been dispatched to their assigned techs, you also want to keep track, throughout the day, of where each tech is in his assigned route, we also have facility for that.

Specifically, if you're having your technicians call-in as they arrive and/or depart from each job, you can do specific *check-offs* in the DispatchMap to indicate each such event. When a technician calls in his arrival at a job, just locate its reference under his name in the DispatchMap, and do a mouse **Ctrl/Left-Click** on it. This will cause a double-right arrow to appear, in place of the check mark that had formerly been there to indicate the job was dispatched. Based on this symbol, anyone in your office can glance within the DispatchMap and instantly determine where that technician is at the moment. When he calls in later to indicate he's done and leaving, do an **Alt/Left-Click**⁵⁶ on the same reference. This will change the check-off symbol to a double-down arrow, signifying his completion there. And so on.

Status of Check Off	Required Keyboard/Mouse Action	Symbol That's Displayed
Item has been dispatched	Shift/Left-Click	√
Tech has arrived	Ctrl/Left-Click	⇒
Tech has finished	Alt/Left-Click	⇓

When changing these statuses in a technician's *List* area, incidentally, you'll likely notice that an appointment's *graphic* representation also changes—as a means of providing further, at-a-glance indication of where the tech is in his set of jobs. Specifically, when he's arrived at a job and hasn't yet departed (or rather,

⁵⁶ Bear in mind that all these kinds of commands are available in convenient reminder format from within the DispatchMap itself. Just **right-click** in any empty space there and you'll instantly see the applicable ContextualCommandSummary. It's nicely categorized, to help you easily find the command/action you're looking for.

when you've so checked it), that job's graphic representation will have a bright yellow background (signifying that's where he is). After you've checked off his completion, it simply has a large 'X' across to signify that fact.

There is another pair of status indications that are not typically done manually, though provision is provided should the odd need arise.

Status of Check Off	Required Keyboard/Mouse Action	Symbol That's Displayed
PostVisitReport has been completed, JobNotFinished	Ctrl-Alt/Left-Click	⊗
PostVisitReport has been completed, JobFinished	Ctrl-Alt/Left-Click, again	♥

Normally, your appointments will be placed into PostVisitReportCompleted status (Job Finished or not) as an automatic consequence of *doing* the report (see page 110)—and not from any separate effort you'd make independently in the DispatchMap. We mention it here solely to give you a full understanding of these different statuses (in a job's graphic representation, incidentally, this status is indicated with grayed-out background, darker gray if the job is finished).

There is another benefit, incidentally, in logging your technicians' arrivals and/or departures (note that you could elect to do only of the two, or none at all)—besides providing at-a-glance indication from the DispatchMap of where they are in their day's work. ServiceDesk logs the times of each such event for you (based simply on the computer's internal clock), which means when you're later doing PostVisitReports, each job's start and/or end times are already there, and so don't need to be manually entered. It's a nice convenience, and provides a little added benefit if you're going to the work of having your technicians regularly call in.

In the best possible world (*if* you've got the office personnel to accommodate it), you'll be using the *immediate call-in method* (see page 112) not only to document your technician's arrivals, but also to take their PostVisitReports. In this case there's still another benefit, stemming from another feature: ServiceDesk's **Arrival-On-Time Sentry**. This feature continually monitors each technician's roster of jobs, comparing the times scheduled to where he is in his route. If and when it appears he's threatening to be late on any job, that job's reference will start *flashing*. The purpose, obviously, is to attract the attention of an office person, who can then take remedial steps (calling the tech to see when he expects to arrive, calling the customer to bring them up to date, possibly reassigning the job to another tech, etc.). The whole idea is to bring the matter to your attention *before* your customer is upset or disappointed. To make this element work, you do need to have the Immediate Call-In option turned on.

v. Automated Dispatching: A Cornucopia of Methods

Aside from tasks such as assigning or reassigning a job, changing status or sequence, scheduling, keeping track of where the tech is, and similar hands-on functions within the DispatchMap, there's another set of features that are concerned, primarily, with *communicating* dispatched job information to your technicians. As mentioned elsewhere, we know it's the habit of *most* servicers to give each of their technicians a stack of assigned work-order/invoices each morning. This stack essentially constitutes the technician's job list. If any additions or other changes occur during the day, most servicers will simply call the technician (via telephone or two-way radio), and communicate such matters orally. But, while this may describe the more common

situation, we know that many other servicers work differently. With ServiceDesk, we handily accommodate a plethora of other methodologies.

In general, these methods are all designed to address the situation where you want to dispatch jobs to technicians who are not physically in the office. Hence, the category of functionality is sometimes referred to as “remote dispatch.”

Some Distinctions

To begin this discussion, we want you to notice a few distinctions. First, please note the difference between “*batch-dispatching*” a whole-day’s roster of jobs to a technician, versus dispatching jobs “*individually*.” Second, please notice that, regardless of which such method you use, the process will be initiated from the technician’s *list* area in the DispatchMap (as distinguished from any job’s *locational* reference there). Third, please notice you can transmit dispatch information via your computer’s internal fax, via printing first to hard copy (which, presumably, may then be faxed to the technician using an external fax), via direct link to separate alpha-numeric paging software, or via email.

Regardless, if you’re wanting to *batch* dispatch (defined as sending information regarding all of a technician’s appointments, for a given day, that are not already checked off as having been dispatched), to initiate the process you need simply **click on the technician’s name** within the DispatchMap. At this point you’ll be presented with a selection of options regarding the method of transmission you want to use. Simply choose the method and proceed.

If wanting to dispatch *individual* jobs, by contrast (which might arise, for example, if you added-on a job later in the day, after having already batch-dispatched the majority of a tech’s work), the process is initiated by doing a **Ctrl-Alt/Rt-Click⁵⁷** on the job’s *list representation* (as found under the tech’s name, in other words). Again, the system will present you with a list options, asking you to choose the method of transmission you prefer. Simply make your selection and proceed.

Various Methods

In regard to the various methods, most are self-explanatory. You can try each and review the result, to decide which will work best for any of your particular situations. In regard to any printed output, please note that you can either print locally and then put the paper in your fax machine as a method of transmitting to the technician, or, when presented with the printer-selection box, you can select your machine’s internal fax, and fax directly—and thus avoid getting involved with paper at all.⁵⁸

In specific regard to the option for *AlphaNumeric*, we should explain a few details regarding the underlying mechanics. If you’re doing alpha-numeric paging, presumably you must have some kind of software package via which the process is managed. Most of these have a method of interfacing with other programs (such as ServiceDesk, for example) via which the other program can request that a particular page (having particular text contents, etc.) be sent out. The typical method is that the paging program looks periodically (such as once every minute, for example) within a particular folder, on the hard drive, to see if any files, properly formatted for paging instructions, are there. For each such file that it finds, it creates and sends

⁵⁷ Please remember that this command, like all the others in the DispatchMap, is conveniently listed in the *contextual command summary* (just right-click in any empty space of the DispatchMap to bring this up).

⁵⁸ If, incidentally, you want to see what the product of such a process will look like when received at the other end, you can experiment by faxing to your own physical fax machine—assuming that, like most offices, you have one of these that’s separate from your computer’s internal fax/modem.

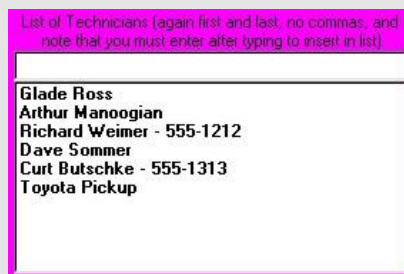
a page accordingly, then deletes the file. Thus, the task for any other program (again, such as ServiceDesk) is simply to create the particular kind of file, and in the particular place, that the paging software will look for. That's what ServiceDesk will do for you when you select the option under consideration here—so that within a minute or so of having done so, your paging software should take over and duly transmit the relevant job information to the assigned technician's pager.⁵⁹

If you're not using paging software that has this capability, it's likely that at least it provides you with a context where you can type a message that you want to have sent to your technician. ServiceDesk helps you here as well, with an option to copy dispatch information into the Windows Clipboard (if you'll recall, the Clipboard is simply an invisible place that can temporarily hold any text that's placed into it). Thus, if you want to dispatch via a paging system where you're required to type in text, all you need to do is select the "Copy into Clipboard" option in regard to the job you're wanting to dispatch. At such point ServiceDesk will invisibly place the dispatch information into your Windows Clipboard. Now all you have to do is go to the place in your paging system where normally you'd type a message. Now just press **Ctrl-V** on your keyboard (the universal Windows command for *Insert*). Wala! The information should appear and you can send your fax.

Another option is to send the dispatch information directly via email. For this capability to work, your computer must be equipped with an internally-installed email client that is *MAPI-compliant* (such as Outlook, Outlook Express or Eudora). It won't work with a web-based account such as Hotmail. At any rate, when you select the email option ServiceDesk will immediately locate any properly installed MAPI-compliant email account as is defaulted on the local machine from which it's running, and use it to email the dispatch information to your tech. It's virtually a one-click deal, and very convenient.⁶⁰

As a final matter directly concerning remote dispatch, we want to mention that some companies have wanted to advance their operations toward a system that we'll here call "*On-Demand Dispatch*." What this

⁵⁹There is a little work you'll have to do to set it up. In particular, if your paging software is going to know the appropriate pager telephone number at which to send the page, it's going to have to find the number within the request file that ServiceDesk creates. This means you'll have to first inform ServiceDesk of the pager number in use by that technician—so that it can then do its job. To do this, go to the Settings form (**Ctrl-F1**). In the "List of Technicians" found there, locate a particular technician's name that you'll be paging. Click on it so as to edit his name line. Now, add text to the line such as to include the appropriate paging telephone number as the fourth "word" within that line (each "word" is defined as any string of text, separated from others by a space; the first and second such "words" in the name line will be interpreted by ServiceDesk as the tech's first and last names, the third as his "focus group," if any, and the fourth as his paging telephone number). If you're not using the focus group option (see page 325), you may elect to use something simple and innocuous, for the third "word" (such as a hyphen, for example) as follows,



where, as you can see, it's the third and fifth technicians that we've setup with paging telephone numbers.

The last little task you'll need to worry about, in regard to getting your ServiceDesk-initiated dispatches properly fed into your system of paging software, is selecting the folder (and probably file-name extension, too) where that paging software will look for its paging data. You'll need to consult your paging software manual for this information. Then, when initiating the page from within ServiceDesk, simply specify the file-name and location as indicated. When done once, the system will remember on each subsequent page so that you'll not have to bother with this specification again.

⁶⁰As in the case of automated alpha-numeric paging, this too requires some setup (in fact, very similar setup). In this case the system needs to know what email address to use for the particular tech. We could have rigged so that you're prompted to type in the address each time, but that would be too much for you. Much better for you to indicate the address once, and be done with it. The place to do it is the same as in regard to a paging number (see last footnote), except make any email address the *fifth* "word" in the line of text that includes the tech's name (again, for any unused preceding words, you can simply use hyphens).

refers to, basically, is a system where only one job is dispatched to each technician at a time. When any technician finishes a job, he informs the office, which in turn determines which job would next suit the overall needs situation best, and dispatches that one to him. Thus, the office remains in much more active management of the overall schedule situation, and is probably much more able to offer near immediate response to customers calling in for service. We mention the concept here, in particular, because you can see that if this is the kind of thing you wanted to do, ServiceDesk would be very ideally suited for it.

On-Demand Invoice-Printing

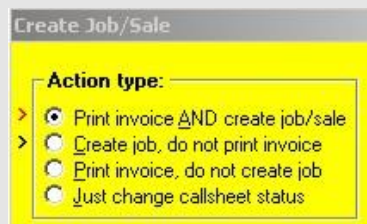
Having just introduced the term/concept of “on-demand dispatch,” we’re going to do our best to confuse by discussing a different *on-demand* concept. This one stems from the fact that we’ve had a number of clients that did not want to print invoices for each job as each job is created (which, you’ve probably gathered by now, is the *typical* ServiceDesk-related method). Instead, they’ve wanted to schedule the jobs, create the JobRecords, and so on, but leave it with no actual ticket printed until immediately before giving the technician his set of jobs (at which time, they finally print the tickets (aka invoices), so they can give the tech the stack that’s involved in his assignments for the day.

If you think about it, it’s obvious this is not properly a subject of “remote dispatch,” but we’re nevertheless discussing it at conclusion of this section because the method for batch-printing a stack of invoices as applicable to a tech’s route, in such circumstances, is accessed from precisely the same context as the various methods for remote dispatch (we say it’s not “remote,” because if you’re giving the tech a stack of invoices, he must be physically present in your office). In fact, it’s just another item in the list of options.

A few notes about using this method:

1. You’ll want to change the default in the *Create Job/Sale* form to the second option, rather than the first (i.e., so the automatically offered default from that context will be to create the job but not print the invoice, instead of doing both).⁶¹
2. You may be like one of our clients and take the on-demand invoice-printing concept to its logical conclusion—not only are you not keeping printed tickets on-hand in connection with new jobs pending their hand-off to a technician; in addition, you further refrain from hanging onto tickets he’s returned with for the sake of giving it back to him when he returns to a job for second or third visits. Instead, you’ll also print fresh tickets on such pre-existing jobs when they’re part of your tech’s roster (thus he has a fresh ticket each time).

⁶¹To change the default, first display the little yellow *Create Job/Sale* by clicking on the Job/Sale button in any Callsheet (to avoid being bothered by messages as may occur when the Callsheet is not properly setup for creating a JobRecord, you might want to go to a fresh new Callsheet and simply type a character or two in the top line, then click on *Job/Sale*). With the form displayed, you’ll see the relevant section as follows:



The screenshot shows a yellow dialog box titled "Create Job/Sale". Inside, there is a section labeled "Action type:" with four radio button options, each preceded by a right-pointing chevron (>). The first option, "Print invoice AND create job/sale", has a red dot in its radio button, indicating it is the current default. The other three options—"Create job, do not print invoice", "Print invoice, do not create job", and "Just change callsheet status"—have white radio buttons.

Just click on the little ‘>’ symbol that corresponds with the option you want to have selected for you by default. When it changes to red, you know that’s the new default.

3. If you've taken step 2, you should additionally invoke the solution we've created to deal with the fact that, when it's not the first visit you're sending the tech out on, the fresh invoice that you're giving him should not look like an up-front, no-work-has-yet-been-done kind of ticket. Instead, it should show the work that's already been done, parts ordered and used, etc. (please contrast this less common scenario with what's more typical, where managers send technicians back with the same up-front-printed invoice that he originally used, and on which he's hand-written this kind of work-done info). We've accommodated this need by creating an option which makes it so, when you specify printing of a tech's invoices from the DispatchMap, the system checks to see, on each involved job, if there's been earlier visits. If so, it will use a FinishedForm for any such particular invoice (in this context you might think of it as an "interim" form), rather than the standard, up-front variety.⁶²

If you don't want to hang onto and manage invoices prior to immediately before handing them to your techs as they head out on their work, this system can work very well. Really, it's a matter of preference whether you want to do it this way or use more traditional methods. As with anything else, there are pros and cons both ways.

vi. Other Within-Map Utilities

Some of our clients have wanted to charge variable service call rates, with the actual amount varying as a function of the customer's distance from headquarters. One of the difficulties in making such a system work, obviously, is in predicting the distance of a caller quickly, while they are on the telephone, and also quickly calculating what the service call amount should then be, based on whatever formula you've setup for the purpose. We've added a feature that makes this very easy. For future reference, we refer to this as the system for *Miles-Determined S.Call Rates*.

As the first element in this system, you may notice that whenever you do an Item-Locate operation to the DispatchMap (i.e., by right-clicking on a customer's address line from either a Callsheet or JobRecord), the system will indicate estimated road mileage to the location in the title bar at the top of your DispatchMap. This will always occur, regardless of whether you want to use Miles-Determined S.Call Rates or not.

As the second element, if you want to have the system calculate an actual service call amount for you, based on some formula and the estimated distance, you must provide the formula. To do this, you simply need to create a file/document that contains three numbers: the maximum number of miles that you wish to have included in the base service call amount, the actual dollar amount for that base, and the incremental amount that you want to charge for each additional mile beyond the base allowance. These three numbers, essentially, should be created as the text within a simple document (using WordPad or similar program), each on its own line, and the result saved (in 'Text Only' format) to a file called '**MileageRates**', to be located within the **lsd\netdata** folder on whichever drive is being used as your FileServer.

In other words, if you were wanting to include a distance of 15 miles in a basic service call rate of \$60, plus charge an additional \$1.50 for each mile beyond that distance, you'd want to create a document with this text:

⁶²To setup this option, simply bring up the DispatchMap's contextual command summary (right-click in any empty space of the form). Then, in the *Dispatch Methods* section of the summary, click on *Set default for FinishedForms in lieu of new Invoice*. At that point, you can follow the prompts.

15
60
1.50

Then be sure to save under the expected file name, expected location, and in 'Text Only' format. When ServiceDesk starts up, it looks for this file. Upon finding it, it reads the data, and whenever you do an ItemLocate to the DispatchMap, besides showing you the estimated distance from headquarters in the title bar at top, it will also show you the Miles-Determined S.Call Rate that results from applying your provided formula values.⁶³

That's *one* miscellaneous utility. Another involves the fact that, at the end of each day, you'll probably want to print a hard-copy list of all the day's jobs. It's particularly useful, indeed, to take this list home so you can, if needed, locate a tech working on late jobs (or confirm that someone who calls in on an apparently missed appointment was indeed on the list). To print such a list, press **Alt-P** from within the map. This will print a list for the day presently displayed.

In conjunction with this *entire*-schedule printing feature, there is still another option that a very few people might want to use (it was requested by one of our clients, so we added it). Instead of printing the schedule onto paper, you can instead print it to a *file*. From there you can import the data into a spreadsheet program (or merge it into a form-letter type setup within an word processing program) to make whatever separate use of the information that you may want (the particular client for whom we made the feature likes to print a daily route sheet, for his techs, onto which they are required to account for mileage to and from every job, time in transit, and similar details). If you have use for the feature, you'll notice this 'Print to File' option is offered on the SelectPrinter form when in the context of printing a whole-day schedule.

In regard to the particular keyboard commands or mouse-sequences that are required for all of these functions, please remember that there is a very nice command summary that's part of the MainMenu, designed as a resource via which you can easily remind yourself of the particular method that's needed to access any feature. It can be accessed directly from within the MainMenu (naturally, under 'Command Summary') or, if you're already in the DispatchMap, just do a **right-click** in any empty space on the map. The appropriate section of the MainMenu's Command Summary will pop into view for you, thus providing instant reminders of whatever particular action you were attempting to remember. This is another "Cheat-Sheet."

Bear in mind there are many scheduling tricks that cannot be done from directly within the DispatchMap because they require direct textual editing, as available only from within the ScheduleList form itself. These tricks (among other items) are discussed in the next section.

⁶³You may also, optionally, setup to have ServiceDesk print this Miles-Determined S.Call amount into your invoice. You simply have to setup a field for it as part of the process in creating the file that instructs ServiceDesk in the format wanted for your invoice printout (see page 291). When using this feature, bear in mind that ServiceDesk realizes not all invoices should have a service call amount printed in (or, rather, we as the programmers realize that). Thus, before printing the service call amount into an invoice, ServiceDesk checks to see if: (a) there's a valid appointment in the 'Date & Time' box (of applicable Callsheet or JobRecord); and (b) there's a grid reference attached to the address. These two checks help to distinguish from the situation where, say, you're simply creating a ticket for an across-the-counter parts sale, and so would not want to have any service call amount printed in for you.

B. The ScheduleList Form

While your DispatchMap excels in presenting your list of scheduled jobs in a format that's rich in both temporal and spatial meaning, there remains the need to occasionally deal with your list in a more ordinary (i.e., textual) format. This is the purpose of the ScheduleList form, from whence you may view each of your scheduled appointments in pure textual manner, edit or change them, add or remove items, and perform many other tasks. It's the utility that's provided, quite simply, for those kinds of purposes.

To view the ScheduleList form, either press **F6** or click on the corresponding command button in the main menu.⁶⁴ As you'll note upon viewing it, the ScheduleList form is configured to display up to 25 items on a single viewed page. If there are more than 25 items in your present schedule, they'll simply fall to subsequent pages of display. To move between these pages, use your PgUp and PgDn keys. To go instantly to either the first or last page, use Ctrl-PgUp and Ctrl-PgDn (the same convention as is followed in the Callsheets, and in several other ServiceDesk contexts). The caption at the top of the form indicates how many pages there are in your list, and which page you are viewing.

i. Adding new appointments to the ScheduleList

Most commonly, your visits to the ScheduleList form will be for the purpose of adding new items to the list itself (i.e., new appointments). Bear in mind in this regard that, for any *new* job, the appointment entry will be made for you (assuming an appropriate notation exists in the job-initiating Callsheet) at the time you create the job. Thus, in those circumstances there is positively no need for you to visit this form; everything is done behind the scenes, leaving you with little or no need to take notice (at most, if you're new to ServiceDesk and want to *verify* such behind-the-scenes work, you may, after creating a job that includes a scheduled appointment, hit F6 to bring up the ScheduleList form and verify that, sure enough, an appropriate entry has been placed in the list for you).

When creating any *subsequent* appointment for a job, however (i.e., after the job's initial creation from the Callsheet), entry of that appointment must ultimately be made from this form. Of course, as you have or will read in other contexts, the process is most easily *initiated* from a item's JobRecord, as viewed from within the JobsCurrent form (see page 108). From there, you may either ItemLocate to the DispatchMap and engage a scheduling process from there (via the On-Map scheduling feature), or more directly use the JobsCurrent form's own 'Scheduling' option button.

In either case, you'll immediately be directed to the ScheduleList form, which is where the actual entry of the appointment (assuming it's after the job's JobRecord was created) must ultimately take place. Specifically, the system will have placed the form into Add-Entry mode, meaning that it has displayed editing boxes in an appropriate position within the list for a new entry. And it will have already inserted, within the

⁶⁴ All the function key commands, each of which executes a different form, work regardless of what form you may already be in. Thus, you may display a succession of ServiceDesk forms (simply by successively hitting their respective call keys), and may view and use them all without ever returning to the main Callsheet interface (i.e., four Callsheets and the MainMenu). Additionally, in this regard, you should know that if you press **F1** from any context other than the main Callsheet interface, (i.e., you're in some form other than the four Callsheets or MainMenu), ServiceDesk will return you back to the main Callsheet interface, without unloading the form or forms you have displayed. This may be useful when you need to go back into your Callsheets to handle an incoming call, and do not want to interrupt work you're presently doing in another form. This is a new feature, however, and you should be warned that the results when you return back to the form you were working in have not yet been fully tested. You should be aware also that as you keep more and more forms loaded simultaneously, you'll be creating a greater load on your system's memory, which may result in system errors. As a general rule, therefore, it's a better practice to exit a form and be returned to the main interface by default, rather than by layering that main interface over the top of a non-closed applications.

editing boxes, all the appointment-relevant text that could be garnered from the initiating context. In other words, it will have inserted invoice number for the job, name notation, telephone number, grid reference, and so on. In the case of On-Map scheduling, it will also have filled-in the 'Date & Time' box (for that information was already garnered from the On-Map scheduling context). In any case, it will leave it to you to manually fill-in any boxes that remain empty (or, in the case of the 'Assigned-Tech' and 'Definite or Tentative' boxes, leave them blank if wanted).⁶⁵

You'll notice that, in regard to filling-in the remaining boxes, pop-up forms and lists are provided to assist you.⁶⁶ If still needing to fill-in the 'Date & Time' box, for example, you'll find the same DatePicker form pops into position much as in the Callsheet context, and a list of time-frames to select from as well.⁶⁷ Plus, there's an additional list that pops down to help you select an assigned tech, and so on. In general, you can probably input anything that's needed in a matter of 2 or 3 seconds. Then, to save the entry (i.e., input it into the list of scheduled jobs), you must simply hit **Enter**. This is important. If you hit Esc instead, the system will decide you must have decided not to proceed with the entry, and will simply back you out, while saving nothing. Assuming that you hit Enter, it will instead add the entry immediately into your list, sort the list (i.e., put it into the proper position time-wise), then take you back either to the DispatchMap (if you got to here via On-Map scheduling) or to the JobsCurrent form (if you got here using its more direct 'Scheduling' option button). In either case, you should be able to instantly verify addition of the appointment in such context (i.e., in the DispatchMap you should see an ordinary schedule notation rather than the ItemLocate flag that was formerly there; in the JobsCurrent form you should see a new entry in the item's history describing the scheduling act that you just performed).

Again, bear in mind that appointments that exist when the job is first created (and that are described with an appropriate notation in the Callsheet's 'Date & Time' box) are added to the ScheduleList for you, when you engage in the job-creation process. All subsequent appointments must be entered from within the ScheduleList form itself—subject to the caveat that it's usually most convenient to do so by initiating the process from a job's JobRecord as displayed in the JobsCurrent form. Either you may Item-Locate the job to the DispatchMap (and invoke On-Map scheduling from there) or use the JobsCurrent form's more direct 'Scheduling' option button. In either case, you'll be taken from such context to the ScheduleList form, where it's then your job to complete the process. Of course, if you want, for any reason (as will be needed, for example, when you're in Learning Mode 2 and no JobRecords yet exist), it's also possible to *begin* the process from directly within the ScheduleList form (it has a 'New Item' button for the purpose) but in that case you'd have to manually fill-in every item of information, which is considerably less convenient.

⁶⁵To be somewhat more specific in regard to these editing boxes, there are eight arranged in a horizontal row, each representing a separate field (or item of appointment information) in the ScheduleList. The first is for the job's invoice number, the second for the customer's name, the third for the location telephone number, the fourth for location's city initials, the fifth for the scheduled time or time-frame (following the same format as prescribed for use from Callsheets, see page 77), the sixth for the initials of the assigned technician, the seventh for a "D" or "T" to indicate whether the assignment is Definite or Tentative (see page 80), and the last for the job's map coordinates (in the same format as was created automatically when you entered the street name from a Callsheet, [see](#) page 71).

⁶⁶There are specific requirements in regard to how an appointment notation must be formatted if it is to be properly accepted and interpreted by ServiceDesk. For a discussion of these requirements, and of certain liberties that are permitted in regard to other text you may want to insert in the 'Date & Time' box, see the discussion beginning at page 307.

⁶⁷If you don't like the set of time-frames that are offered in this list, you can easily make a different list. To do so, simply create a file that contains a list of the time offerings you do want (make sure your line-item offerings conform to expected ServiceDesk format, as outlined in the Appendix, p. 319). You can do this using Notepad (a utility in Windows), WordPad (another such utility), or almost any other text editing software. From any such program, begin a new document and type-in the list you want (i.e., type one time-offering as wanted, hit Enter for the next line and type the next, etc.). When done, save the file (be sure it's in 'text-only' format) as '`\\sd\\netdata\\TimeFrms.Lst`' on the "server" drive as applicable to your ServiceDesk setup. Seeing that such a file is now present, ServiceDesk will load the list *you've created* in lieu of its default list.

ii. Changing or Canceling Appointments

Besides adding new, post-job-creation appointments to your ScheduleList, other important functions within this form are to *change* existing appointments or *cancel* them (events that are often precipitated because of requests from the customer). Please note that any such actions must ultimately be done from *within* the ScheduleList form (much as after-the-job-has-been-created appointments are also added from here). However (and again to continue the parallel, the actions may be *initiated* from other contexts).

To provide the fullest possible picture, let's suppose you want to change or cancel an appointment. The most direct (albeit not usually the easiest) method would be to hit F6 for the ScheduleList form, then within its listings locate the appointment item in question. Then just **left-click** on the item if wanting to edit (after which its text will be enclosed in editing boxes, ready for the purpose) or **right-click** if wanting to delete (after which a message box will ask for confirmation of your intent).

That's plenty straightforward, and is probably all that would be needed for a very small service operation. If your own operation is a little larger, however, it would likely prove a little burdensome to locate the particular appointment—among many scores of others—as displayed within the ScheduleList. For this reason, we've provided a couple express means to locate a particular appointment of interest, among all the others.

The means of choice would depend on the context that most convenient when you decide that you wish to edit or delete an item.

In particular, if you're in the DispatchMap, looking there at an appointment that you want in some manner to change (in particular, in a manner that's not directly available from within the DispatchMap itself), all you need do is a **Shift/Left-Click** on the item in question (either the list or graphic reference, it doesn't matter). We call this the *QuickLink*-to-ScheduleList operation. It will immediately open the ScheduleList form for you, locate the reference that pertains to the item in question, and enclose it in editing boxes, ready for any changes you may want to make within its text. All you need do now is make those changes, then hit **Enter** on your keyboard to save and return to the DispatchMap. If, on the other hand, you wanted to delete that particular appointment item, you'll need to hit **Esc** from your keyboard to remove the editing boxes, then **right-click** on the item line, in order to initiate its deletion.

If, on the other hand, it's more convenient to simply locate the JobRecord that pertains to an appointment that's being cancelled or changed, that's an alternative path you may in such instances prefer. In this case simply locate the JobRecord that pertains to the appointment in question (in the typical scenario, in other words, you'd press **F7** to bring up the JobsCurrent form, then probably '**N**' to initiate a quick name search, then type the first few characters of the name, then hit Enter on your keyboard to locate the job). Once it's located, you can simply select its 'Scheduling' option (click on the item, click on the appointment box, or hit Alt-S on your keyboard). In response (and much as when you've done a QuickLink-to-ScheduleList from the DispatchMap), the system will open the ScheduleList form, and locate any particular appointment, as attached to the job, as may be pending. ⁶⁸

⁶⁸ Note that we did virtually the same thing when wanting to *create* a new appointment entry in connection with a pending job (see page 107). The action here is much the same, the only difference being that when as you initiate the action, if the system locates a pending appointment on the job, it will first offer you the option of editing or deleting it, rather offering only the option of adding a new appointment.

In any case, after you've changed or deleted an appointment, the system will automatically add a note into the JobHistory reflecting that fact. Thus, all significant actions are appropriately documented, and with virtually no added effort on your part.

iii. Making Entries Regarding Times When Techs are Not Available

While those are the more basic and common functions to be employed in the ScheduleList form, there are some others you'll likely want to use from time to time. The most important involves use of the ScheduleList not merely as a means for tracking each of your job-performing appointments, but to document other items of significance in your schedule as well. In particular, it's critical for all dispatching personnel to know if there are particular times when one or more of your technical staff are unavailable for scheduling. What you want, moreover, is for them to be able to see notations reflecting these realities from right within the DispatchMap. This can be done quite easily, simply by creating entries in the ScheduleList, for any given day that a technician will be unavailable (or even partially unavailable).

What's primarily different about these entries is, instead of having an actual customer's name in the customer name box, they'll instead have a special phrase that has meaning, as such, to ServiceDesk. Specifically, the phrases are any of the following:

'Unvbl',
'Unvbl until' and
'Unvbl after'

You'll see these phrases in a drop-down select box, when you place your cursor in the box where you could otherwise type a customer's name.

The general notion is that if a technician is taking the morning off on a particular day, and is not expected to arrive for work until 1:00 pm, then you'd make an entry—for that particular day of unavailability—placing the above-described *'Unvbl until'* statement into what is normally the 'CstmrName' editing box. Then, in the 'Dt & Tm' box, you'd place the applicable date and time (much as you would for any other appointment), but in this case, rather than describing an actual appointment, you're describing the time that you expect your technician to arrive for work on the day in question (after being unavailable for a portion thereof). Of course, you'll want to put his initials into the 'TechAssgnd' box, so that it's clear to whom the entry refers.

It's the same general notion if your tech needs to *finish work early* on a given day, but here you'd use the *'Unvbl after'* statement. And here (again in the 'Dt & Tm' box), you'd indicate particular the date and time *after* which he'll be off.

If, on the other hand, your technician is going to be off for an entire day, you'll want to use the naked *'Unvbl'* statement, and simply indicate the day (no time applicable) in the 'Dt & Tm' box. If he's taking several days off, you'll need to make a succession of separate such entries, one for each of those days.

In result of this kind of effort, a user in the DispatchMap will see, under an appropriate technician's name, something along the lines of **"Unvbl until 12:00"** for an appropriately applicable day (or simply **"Unvbl,"** or whatever the case may be). Thus, everyone who glances at the map has instant access to this important information.⁶⁹

⁶⁹In a somewhat similar vein (i.e., using the ScheduleList for purposes other than referencing an actual customer appointment), I have sometimes had my secretary insert entries to indicate that I have an appointment to interview a prospective job candidate, or to go see the dentist, or whatever. You can make almost any kind of entry, pertaining to a particular date and time and any person who's included

iv. Other ScheduleList Tricks

There are a number of special situations that bear mentioning. One arises, for example, when you have a primary technician assigned to a job, and another that's assigned to go help him. In this case, you'll need a separate ScheduleList entry for each of the involved techs, so each will have a route and list showing the job. However, only one technician needs to be responsible for reporting on the job. There's a simple means to help ServiceDesk distinguish this situation (so it knows better than to pester, for example, for reports in connection with ScheduleList entries that only involve a *helping* technician). In place of the customer's name, on such "helping tech" type entries, place the key word "**Hlpng**" followed by the primary tech's initials (i.e., "Hlpng DS"). Be sure the entry otherwise includes the job's invoice number, an appropriate appointment reference, grid reference, and so on—as with any other ScheduleList entry. Finding that basic key word in the CustomerName field, ServiceDesk will know to treat the entry differently in appropriate circumstances.

As another situation, you might sometimes find there's a job scheduled that you know (perhaps because he's told you) will take the technician an extraordinary amount of time. It's nice to be reminded of such expectations when you look at your DispatchMap. To accomplish this, consider appending the respective customer's name in the appointment reference (or the date & time reference, or both), in such a case, with asterisks to indicate the expected, longer-than-normal completion time. Indeed, you might use varying quantities of symbols, each representing 30 minutes or an hour of expected work time (depending on the convention you adopt in your office). Thus, if you expected that a job was going to take 2.5 hours (and figure normal jobs average one hour), you might add three asterisks. Then, anyone who looked at the map would know, at a glance, that this job was expected to take approximately that much time. To illustrate the effectiveness of this, take a glance at the following string of appointment references:

12/4 WED 9-10 ****
12/4 WED 1-4
12/4 WED 2-5
12/4 WED 3-6
12/4 WED 4-7

You might imagine how, once you get used to how the asterisks are used, it becomes very intuitive to glance at a string of references like this and realize, without even thinking about it, that the first appointment is expected to take about 3 hours (one hour normal, plus an added 30 minutes for each asterisk), while the others are all expected to be about average. It's really surprisingly effective, and yet very simple (ServiceDesk in itself, by the way, does not pay any attention to such symbols).

Another matter concerns appointments that are essentially for a *whole* day. In other words, you've not committed with the customer to any particular time or time-frame, within a day, but just for a day at large. Perhaps, for example, you've scheduled for days ahead and it's not your practice to commit to a particular time until the morning in question. Or maybe the customer will be gone all day and is leaving a key under the mat, so any time during the day is simply fine. Regardless, ServiceDesk will recognize a whole-day appointment if, so far as the time-portion of an appointment reference is concerned, rather than a time or time-frame it encounters a simple exclamation symbol (i.e., a notation such as '**12 THUR !**'). In this regard you'll notice this

in your TechList, as wanted. An important factor to notice, in such regard, is that in any of these cases you'll be leaving the field blank, within the ScheduleList, that would normally include a job's InvoiceNumber. That's obviously because, for these kinds of references, no job InvoiceNumber is applicable. And this is important, for ServiceDesk uses that fact (no InvoiceNumber in the reference) to deduce that it's not an actual job appointment that the reference refers to. Thus, these references (ones with no InvoiceNumber) are never added into the count of jobs, and there's no effort to graphically display them within the DispatchMap either. We mention this so you'll know, if you ever make a reference and expect it to display graphically without the InvoiceNumber, it simply will not.

is what's inserted for you if, from the TimeSelector box in any of its contexts, you choose the 'Any-Tm' listing (i.e., the listing is an abbreviation for "Any time").

We mentioned in connection with describing the DispatchMap that jobs can be manually re-sequenced, there, by the simple act of clicking on an item within the Tech's list and dragging it up or down. A similar function is available from within the ScheduleList, though we think it unlikely you'll have the urge to use it (its presence here is vestigial, having been created *before* we added the more convenient method from within the DispatchMap). If you want it, however, it functions in a slightly different manner. Simply click on the item you want to move, which of course causes it to be enclosed in editing boxes. Then use the up or down cursor keys to move it within the list as wanted.

In regard to the general matter of re-sequencing jobs, we realize there may be occasion when, rather than wanting to change the relative sequence between say two '9-12s' (or any other jobs that are equal in time value), you instead want a '9-12' *moved ahead of an '8-11'* (or some other such unnatural sequence). ServiceDesk will allow you to do this, no problem—except that during its next auto-sort process (which happens every time a ScheduleList entry is saved), it will move any such unnaturally sequenced jobs right back to logical order. Yet you may want the '9-12' sequenced before the '8-11' for good reason: perhaps because it fits the tech's route better, for example (and you figure he'll get to the '8-11' in plenty of time regardless). So, how can you prevent the auto-sort process from putting the jobs back around after you've *unnaturally* sequenced them?

There are two methods by which you may deal with this (since this subject is a little arcane, if you want to skip the next two paragraphs for now and return only when needed, we encourage you to do so).

First, you may subtract or add one hour to the sort-value⁷⁰ of a stated time or time-frame. In the case above, for example, you could simply put a '-' sign after the '9-12' notation (i.e., in the ScheduleList entry for that appointment make it read "**13 FRI 10-1 -**") or a plus sign after the '8-11' reference (i.e., in the ScheduleList entry for that appointment make it read "**13 FRI 9-12 +**"). In consequence of either alteration the Sorting routine will see the two appointments as time-equivalent, and thus will not change them from whatever sequence you've manually placed them in (bear in mind that the plus or minus sign is expected as a *fourth word* in the text, meaning there must be a space between it and the time designation or ServiceDesk will not perceive it as such).

As a second option, you may take advantage of a matter that is incidental to the whole-day scheduling feature (see five paragraphs back). It's simply logical, in regard to jobs where you may show up at any time, that sometimes you'll want to sequence them toward the front of a tech's schedule, sometimes toward the end, and so on. Yet those jobs have no inherent time-value for auto-sorting, or if they did it would logically have to be mid-day. But if a mid-day value were used, they'd often get moved (by the auto-sort process) out of an early or late sequence where you might have preferred to put them. For this reason, the auto-sort feature is programmed to leave whole-day schedule items (i.e., those with a '!' in place of any time reference) in whatever relative sequence it finds them. You may take advantage of this even when dealing with an appointment notation that includes a time reference (and yet that you want to sequence unnaturally). Simply place that exclamation symbol as a *fourth word* in the appointment reference (i.e., do it something like "**13 FRI**

⁷⁰ It may help to understand how ServiceDesk attaches a sorting value to different date and time references. Essentially, where your appointment encompasses a designated space between times, such as 9-12 for example, it computes the average between the two, and uses this as the value for sorting. Thus, 9-12 (averaged to 10:30) would be listed before the single time of 11:00, and after the single time of 10:00. Similarly, the time-frame of 9-11 would be listed before 9-12, while 10-12 would be sorted to list after.

9-12 !").⁷¹ The disadvantage of this (over using *plus* or *minus* symbols to merely alter the item's sort-value) is that an item thus marked will not be auto-sorted at all. These items, in other words, are exclusively subject to manual sequencing.

Finally, please be aware of a very handy tool. For many of the tricks discussed in this and the preceding two sections, you obviously need to first locate the relevant appointment entry within the ScheduleList (in order to delete an entry, for example, edit it for certain purposes, etc.). Most often, you'll realize the need for this while looking at an appointment from *within the DispatchMap*. You could at such point hit F6 to bring up the ScheduleList form, then peruse through its listings until finding the one you're interested in, then left-click on it for editing. As a shortcut, however, it's much easier to simply do a **Shift-RtClick** on the item's list representation within the DispatchMap. This immediately loads the ScheduleList, locates the item in question for you, and encloses it within editing boxes—thus making any within-the-ScheduleList work that needs done very easy.

Here in the ScheduleList form, by the way, there is once again a "Cheat-Sheet," based on the MainMenu's Command Summary, much as is contextually available from within the Callsheets and the DispatchMap. To access it, just **right-click** within any otherwise un-operative space.

The above is, by the way, the general rule for producing a contextual Cheat-Sheet (i.e., to right-click within any otherwise un-operative space of an applicable form). As an aside here, the particular forms that have Cheat-Sheets are Callsheets, the DispatchMap and ScheduleList form (each of these already discussed), plus the PartsProcess form, and Inventory Control and PartsProcess forms – six forms total. These are the particular forms where we've needed to embed actions to afford greater user power. As with other matters, our suggestion is that you make no overt effort to memorize the particular commands as involved in the Cheat-Sheets. Instead, just remember the particular six forms that have them, and use the Cheat-Sheets in each as needed. Eventually (and through use), you'll find yourself accidentally remembering the commands you most use, and no longer needing the Cheat-Sheets except for unusual purposes.

v. Matters of Housekeeping

As one workday expires and you begin another on the following morning, you'll find, naturally, that your ScheduleList still contains entries reflecting the preceding day's schedule. Yet, that day is past, and you no longer need its entries there. They clutter the list, moreover, making the entries you are interested in somewhat less accessible. Of course, this effect is compounded with each succeeding day, so obviously, you need some means to instantly clear out items that pertain to past days' appointments. For this purpose click on the form's 'Archive' button (or press **Alt-R**). In response, ServiceDesk will remove past appointments out of the ScheduleList and at the same time transfer them into your ScheduleArchive, which you may access for later search and review using the ScheduleArchive form (accessed by pressing **Alt-F5**) or by simply paging into past days from your DispatchMap. Thus, your current list will be kept relatively short and small, containing only items that are *currently* scheduled.

Prior to Version 3.8, this item of housekeeping was never automatic. You were required to take the initiative to do it yourself (even though, if you failed to do so, you'd find the system pestered you with urging reminders). Now there is an Auto-Archive feature which, if turned on, will perform this and other archive functions automatically each night (see page 209). If you happen to not be using this feature, you should

⁷¹Actually, as of Version 4.0.87 there's no longer a need to do this via manual text editing (as described in the text). In that version we introduced a feature that allows you to do this via click-action from within the DispatchMap. Just do a **Shift/Left-Click** on any appointment's *graphic* reference there to toggle on or off this per-item block against auto-sorting by ServiceDesk.

remember to manually invoke the Schedule-Archive process first thing each morning. That way the previous day's appointments are immediately cleared from the current list, and there's little chance for it to become cluttered with old stuff.⁷²

This is particularly important in view of still another function in the ScheduleList form. One of our concerns when using the PostVisitReporting system (see page 110), is to assure that a report is properly filed on each dispatched appointment. The ScheduleList form helps to accomplish this purpose. As part of the Archiving process, it makes secondary copies of all past scheduled items. These are kept in the ScheduleList, though in a category not typically displayed, while originals are sent into the archive.⁷³ The ScheduleList holds these secondary copies in a format that is displayed only when you select the ScheduleList form's *'Waiting for Tech's Report'* or *'All Items in File'* display options (thus, the copies do not clutter your ordinary work). Each copy is maintained here until a proper PostVisitReport is filed on the job, at which point ServiceDesk finally removes it from the list.⁷⁴

The result, operationally, is that you can instantly verify at any moment if there are any jobs that were formerly dispatched to any of your technicians on which a PostVisitReport has not been properly made. Just hit F6 to view your ScheduleList form, then 'W' to select the 'Waiting for Tech's Report' display option. As you'll see, you can specify at this point whether you want to check with regard to jobs dispatched to a particular tech, or in regard to all your tech's in general. Obedient to your command, ServiceDesk will immediately display the information requested.⁷⁵

It's a good idea, we've found, to make this check in respect to each technician before giving him a new day's work. This way you can verify and demand that he's reported on all his past dispatches before giving him any new ones.

C. The Zone-Scheduler System

At this point we're through with the *main* discussion about scheduling and dispatch. You'll want to read further, into this section, only if you have specialized needs. In particular, if you want to coordinate with other companies in a manner that allows them to schedule and dispatch appointments on your behalf (such as through ServiceBench, for example), or if yours is simply a very high volume operation (i.e., 10 trucks or more), you'll likely find information here that's helpful. Otherwise, you might just want to skip this section.

The Zone-Scheduler system came about, initially, because Whirlpool Corporation wanted to schedule jobs on behalf of its independent servicers. In other words, a consumer calls into Whirlpool's National Call

⁷²Because of the algorithm that's used in deducing which actual calendar date each appointment notation refers to (see page 77), you'll find that if you go more than five days without running the ScheduleList's 'Archive' routine—a lapse that would constitute gross neglect, to say the least—items that exceed such distance in the past will not be cleared. If you're slothful and allow this to happen, be sure to clear those items manually (i.e., use the item-delete function by **right-clicking** on each item you want to get rid of).

⁷³The exception is if, in place of the customer's name, the appointment reference has the keyword 'Hlpng' (as in **'Hlpng DS'**, see page 111). This type of entry is simply dropped after its time is past.

⁷⁴Or until it's at least five days old, at which time (at least if you have not resisted running a ScheduleList Archive event) the system removes it regardless, while informing you of the fact and urging that you should have assured the item was removed through the more proper process of filing a PostVisitReport.

⁷⁵If you've selected the 'All Items in File' display option, items that are from past days and still awaiting a PostVisitReport will display in a *medium gray* color. Items from the past that have been reported on (and are thus awaiting deletion) or that have simply been marked for deletion, are displayed in a *very light gray* color.

Center with a problem on their new appliance. Instead of simply giving the consumer the telephone number of some independent servicers they could call, or taking the consumer's information and themselves calling a local servicer (who'd then call the consumer to schedule), Whirlpool figured it would be a much higher level of consumer service if their call-taker could do the whole thing: take the complaint and, correspondingly, and go ahead and schedule the call on the intended independent servicers behalf.

To make that system work, Whirlpool went to work with ServiceBench (essentially, a web-based information processing intermediary). They setup a system whereby each independent servicer can divide his territory into any number of "Zones," and simply indicate the quantity of jobs that may be scheduled for each zone on any given day (or for a particular segment of day, if preferred). As "slots" get filled up from the servicer's end, he goes to the ServiceBench website to indicate the decreased availability, and thus Whirlpool's National Call Center constantly knows for which days the servicer is available for scheduling within any given zone.

It's a smart system, and other manufacturers are gradually joining (as of June 03 we understand Frigidaire is also on the system; we're not sure who is or is not from other industries). We also understand that General Electric has setup a similar process in connection with Key Prestige (the information processor they use), and that certain consumer electronics companies have setup similar capabilities through NESDA. It's definitely the wave of the future.

This obviously means your service management system needs a means of working with the issue of "Zone Availability." We've worked this out rather wonderfully, and it's integral to ServiceDesk. Because, however, we figure it's a capability that a minority of our clients will use, the operation is described in a separate document (you'll find it either in your c:\sd folder or on your ServiceDesk installation CD) entitled "ZoneLoadInstructions.Pdf." If you have a need in this area, simply read and follow the instructions in that document.

As a final note, even if you're not setup to take dispatches from one of these kinds of operations, if you're running 10 trucks or more, you may find the Zone-Scheduler system is a somewhat more efficient means of allocating your jobs than is the DispatchMap.

D. The Web- and Email-Based Dispatch Enablers

Besides creating the above-discussed ZoneScheduler as a system to assist in the coordination of scheduling by other companies (or more effectively accommodate extremely large volumes of appointments internally), we've created a couple of related utilities (each installed as small, stand-alone programs) that are quite revolutionary in terms of the integration provided.

For context, consider that if you have outside companies that are dispensing dispatch requests to you (whether pre-scheduled on your behalf or not), there ought to be a more efficient means of receiving the dispatch information—rather than by having someone from the institution call your office with the request. In fact, for a few years many institutional dispatchers (such as manufacturers seeking OEM warranty service, extended warranty companies, home warranty companies, and so on) have been employing methods such as fax or email for the purpose. But even here, it's less efficient than it should be, for though the information arrives at your office with reasonable ease, there's still a terrible inefficiency since a human being must *read*

the received text, and then re-type it into your own software system. That's kind of stupid, frankly, when much better can be done.

With ServiceDesk, we do indeed offer much better. Specifically, we have two stand-alone utilities that facilitate the dispatch process in an *automatic* fashion. Say that Whirlpool or American Home Shield want to dispatch a job to you? Do you have to answer your phone? Do you have to check your fax? Do you have to check your email? Do you have to log into a website and check? Not when equipped with these ServiceDesk utilities. You don't have to do anything of the sort. Instead, within moments of the when Whirlpool or AHS generates a dispatch, you'll see all the relevant information pop into a new Callsheet on your ServiceDesk screen. No effort required on your part. No reading of the information from elsewhere and re-typing it into the system. It's all done for you.

We have two different utilities in this regard; one is called the *Web-based Dispatch Enabler*, the other the *Email-based Dispatch Enabler*. Each is structured to work as their names imply. Specifically, some institutional vendors of dispatch requests (such as AHS) have structured their systems to dispatch via email. Others, such as ServiceBench (and the manufacturers that work through ServiceBench) have setup to do so via the Internet. For whichever type, you'd simply install and run the utility that corresponds. With that utility running, you'll get the dispatches automatically.

And that's not all!

As mentioned when discussing the ZoneScheduler system (see preceding section), some of these institutional vendors want to do more than simply leave it to you to call the homeowner and schedule in response to the dispatch request. Some would prefer to do the scheduling for you, up front. But of course, it's not practical for them to do so this unless they know when you're available for scheduling. For this reason (as also mentioned in that previous discussion), some operations (including especially, as of June 2003, ServiceBench and P.C. Richard) have a means by which you can keep them informed of how many "slots" you have available for scheduling. For this reason, besides *receiving* dispatches, these two utilities also have the means to *automatically* inform such institutions of your scheduling availability.

In regard to ServiceBench and the WebBasedDispatchEnabler, it's all done via the one, unitary utility. In regard to the EmailBasedDispatchEnabler, it's done via a separate utility, which is nevertheless packaged with the main one. Regardless, either system will look at your ZonePlanner to see how many jobs you've allocated for each zone and/or time segment, compare such allocations to what's presently scheduled, and transmit the resulting number of available slots to the relevant institution. It will do this virtually in real-time, so the institution is constantly informed of precisely how many slots are remaining as available to them for each day, each zone and each time segment as you've allotted.

Aha, we do live in a modern age!

Each of these utilities is available, essentially as an *accessory* to ServiceDesk (a more complete discussion of the WebBasedDispatchEnabler is in the Appendix at page 318). Just contact us for current pricing if you're interested.

Chapter 7

JOB MANAGEMENT

Already, before even fully concluding our discussion on ServiceDesk's call-taking and dispatch functions, we've ventured almost unintentionally into a new area of concern: tracking and managing jobs after dispatch. ServiceDesk would be a poor servant, indeed, if it did not have systems (at least as elaborate as those designed to help you secure jobs) to also help you assure their proper and efficient completion. These tools are the subject to which we now turn.

Hint: If using the video tutorials, please read this chapter in conjunction with having watched Lesson # 4.

A. The Work-In-Progress System

The first step in tracking job performance occurs the very moment you invoke the Job/Sale process from Callsheet. At this instant (assuming standard defaults, at least), ServiceDesk adds a new record to your JobsCurrent file. This record contains, as a start, a copy of all the information from the initiating Callsheet, along with a history section that has, as its first entry, a notation regarding the date and time the record was just created, and by whom. All this happens in the background, creating no cause for notice while you're going through the process. This JobRecord becomes the basis, however, for almost all of the tracking and managing of job performance that follows.

i. The JobsCurrent Form

Your primary window into the contents of your JobsCurrent file (i.e., the file containing all current JobRecords) is the JobsCurrent form—accessed by pressing **F7** or by clicking on the corresponding MainMenu command button. As the form displays you'll see that, unlike many forms (which show several records at once), this one is designed to display just one record at a time, each constituting a full page of display. Thus you're able to see a great deal about the history and status of any particular job, on its own unique page.

Of course, it's impossible to view even one job until it's selected for display, so several locating features are provided. As you'll see upon viewing the form, you're initially shown the file's last page (i.e., the most recently-created JobRecord), and can instantly return to that page (assuming you've navigated elsewhere) by using the standard, **Ctrl-PgDn** command. Of course, you can similarly jump to the form's first page (i.e., your oldest still-pending JobRecord) using the standard, **Ctrl-PgUp** command. Upon selecting any

particular record, you can use the **PgUp** or **PgDn** keys to display adjoining ones, or just browse. It's handy, for example, when you're in a Callsheet and want to check the record of an item that was among the last few printed, to simply hit F7, then hit PgUp a few times until you see the record you're looking for.

Of course while useful, browsing is often less than ideal (many users will have more than a hundred records in their JobsCurrent file at any moment). So more pointed search methods are provided, allowing you to find an individual JobRecord based on its InvoiceNumber, CustomerName, Address, or even P.O. Number (useful for those home-warranty type jobs that involve them).⁷⁶ Just click on or use the indicated QuickKey to access the search option you want, and ServiceDesk will prompt you to enter the search target (or as few leading characters thereof as you wish). At your bidding, ServiceDesk will then search in the actual file, in the order from most recent record to most ancient, and will display the first match found. If this was not the item you were looking for, you can hit Enter and the search will resume. Once a record is loaded and displayed, you can view and/or edit it, just like in any other context.

In regard to conducting these JobsCurrent form-based searches, bear in mind you are only looking within the JobsCurrent file, and in fact there are many other ways of searching within your job records. If interested in *archived* JobRecords for example, you might instead conduct searches from within the JobsArchived form (see page 120). Or, if interested in both categories at once, it might be preferable to use the CstmrDbase system to perform an instantaneous search (see page 66 for a non-technical summary of that system's *methods*, or page 283 for an in-depth description of *how it works*), though it does not offer some of the more targeted search methods that are available from within these two forms (such as searching under a specific P.O. number for example).

In specific regard to CstmrDbase-type searches, there is a very handy feature that's available from directly within the JobsCurrent form (and this is, incidentally, not obvious on its face). Occasionally you may be looking at an existing job (or, rather, at it's JobRecord within the JobsCurrent form) and be curious about what other jobs have been done for the same customer (or at the same location) in the past. Certainly, you *could* easily press F12 (for the dedicated CstmrDbase search utility, see page 209) and begin typing in a search target by way of name, address or telephone number. However, those very targets already exist in the JobRecord you're already looking at. Why should you have to bother re-typing them? Instead, if you simply hit **F1** while any job record is displayed in the JobsCurrent form, the system will instantly do a search on each of its search-able fields (i.e., customer and location names, customer and location addresses, customer and location telephone numbers). Or if you've put your cursor *into* a particular field and hit F1, it will search on that field only. You can then click on displayed list items to view the corresponding past job (or hit F1 again while the list is displayed) as in any other context.

Also in regard to searches, you should be aware of a still another utility that we call the *JobsPerusal* form. To understand its function, suppose you're wanting to review all of your current jobs that are in 'Working to Schedule' status (perhaps it's the time of day that you should be calling all such people in the effort to

⁷⁶You should remember that, to interpret a "P.O. Number" as *such*, ServiceDesk must find it in the place and format where it "expects" such a number to appear. You'll notice in this regard that a Callsheet (and the JobsCurrent form's corresponding layout) does not have a dedicated, set-aside textbox for this purpose. This is because, by design, we're trying to keep Callsheets simple (a purpose that would be defied if we forever multiplied their textboxes for each and every conceivable purpose). Instead, for a few specific purposes we've written code into ServiceDesk that allows it to semi-intelligently pull some items of data from otherwise common text. A job's "P.O. number" is one such potential item. On every job, ServiceDesk looks toward the right side of the CustomerName box for a particular sequence of characters. If found, it interprets the associated characters as a P.O. Number and allows for separate treatment as such.

Specifically, there are two bases under which ServiceDesk will interpret a string of characters, found there, as a P.O. Number. First, if the *last word* (or group of characters) is all numeric (i.e., all numbers and no letters). Or second, if the last word (whether all-numeric or not) is preceded by a pound (i.e., "#") sign. For further explication (particularly in regard to setting-up a QuickEntry template that allows for this), see page 64. For an illustration of a service ticket as printed with such a setup, see the exhibit, last page of this manual.

schedule with them, and so need a way of looking simply and solely at that particular set of jobs). That's the kind of purpose that the JobsPerusal form fulfills. To access this form, press **Shift-F7** on your keyboard (you may notice that we've tried to group all the various "Jobs" forms around the F7 key). Once you've displayed it, you'll find that use is pretty self-explanatory. Basically, you'll just select the status category of JobsRecords you want to review. The form will then inform you of how many there are in that category, and allow you to scroll through each, one by one.

In regard to how each actual JobRecord is displayed, you'll notice that besides information that was copied from the initiating Callsheet, there is a section specifying the job's 'status' (i.e., it may presently be '*Dispatched to Tech*', '*Waiting for Part*', or in any of several other categories). These status settings are all done automatically on the basis of information gathered by ServiceDesk in other contexts, so there's no need for you to worry about keeping the settings up-to-date. They simply provide information to you (one detail you'll notice is that the '*Recorded to SIsJml*' and '*Marked for Deletion*' status categories are somewhat set apart within the list; this is to emphasize that it's only a record that's in one of these two categories (unless certain Learning Mode options are set otherwise, see footnote 119 at page 166) that will be moved out of the JobsCurrent file when an ArchiveJobs routine is run).

Even more informative is the JobsCurrent form's History section, which documents every important event in a job's progress toward completion, whether concerning its initial creation and dispatch, a tech's time on the job, diagnosis and repair work there, ordering of parts, items used from stock, money's collected, re-scheduling efforts after ordering parts, eventual completion, etc. With minor exception, these entries are also automatic, created by ServiceDesk with no additional effort on your part, again in consequence of actions performed in other contexts. The power of this utility is enormous. At any moment you can instantly know the entire history and status of any job. No more need to go running for paper records, or pleading here and there to find someone in your company who knows what's happening. It's all there for you, ready for instant and sensible display with practically no effort.

Besides *checking* on a job's History, you may want to make your own, additional notes in it. Or you may want to edit notes already there. No problem. If merely editing, the procedure is obvious. If adding a *new* note, first click on the '*add to History*' button (or press 'H'). This will insert a time and date stamp for your note, and position you to begin typing. This is particularly useful when, for example, you're wanting to document having called the customer in an effort to re-schedule after having ordered in a part. In such a case you can add notes much as in a Callsheet's MoreInfo section, and again the use of abbreviations (like "Imor" for "left-message-on-recorder") will make it even easier.

Besides being used to review jobs, update historical events, and so on, the JobsCurrent form provides utilities for several kinds of *actions* that may be involved in furthering a job's performance. If you're simply wanting to call the customer on such a job, for example, you can auto-dial from here, just as from within a Callsheet or several other contexts (just right-click⁷⁷ on any or the form's telephone number fields). If wanting to schedule an appointment for the job, you can use a couple of different methods, as specifically discussed in the next section (it's a sufficiently involved topic to warrant its own discussion). If you want to create a new service ticket for the job (perhaps the original was misprinted, mangled, lost, etc.), a formal claims presentation document, a work history or even export an Ascii file describing the job, there are options on the form for all such possibilities.

Finally, if you want to attach a UnitInfo sheet (see page 180), enter the receipt of monies collected (see page 156), or create a non-stock part order (see page 122), these events may also be initiated from the

⁷⁷ If the simple right-click causes any trouble in this context, see the discussion at page 328.

form—but in these cases please bear in mind that most typically such actions will instead be initiated via the PostVisitReport process (see page 110). Essentially, they are available from within the JobsCurrent form for those particular occasions where, because of how circumstances arose, it simply did not fit (or perhaps someone failed) to do them more normally.⁷⁸

The last major function in the JobsCurrent form involves, as in so many other forms that manage major files, an archiving process. As jobs are completed and sales reported thereon, it's obvious they are no longer work-in-progress, and there's no longer a need for their records to be in the file where we keep track of in-progress items (i.e., the JobsCurrent file). At least once a week, therefore, it's important to run the JobsCurrent form's *Archive* routine. Either you may invoke the procedure manually (click on the designated button or use its QuickKey), or you may use of the Auto-Archive feature (see page 209) to have it done for you—either automatically on a nightly basis or as part of an umbrella process invoked manually each morning (see page 221). In either case, ServiceDesk will run an internally-elaborate routine that involves, most obviously, moving those records that pertain to completed⁷⁹ jobs out of the JobsCurrent file and into the JobsArchived file. It will also destroy any records you have marked for 'Delete' status, thus leaving your JobsCurrent file cleansed of all but jobs that, at the time the routine is run, genuinely represent jobs still in-progress.⁸⁰

ii. Scheduling on Existing Jobs

Previously (see page 70), we discussed the process of scheduling as it occurs in the context where we're typing a job's initial order information into a Callsheet (essentially, we use any of several methods to create a notation, in the Callsheet's 'Date & Time' box, describing our appointment, then ServiceDesk inserts an entry into the ScheduleList for us as the job itself is created). Now we want to discuss the *other* context, which occurs whenever we're scheduling on a job that *already* exists (such as, for example, when wanting to make a new appointment after ordering in a part that was diagnosed on an already-created job's first trip).

As you might expect, this process is initiated from the JobsCurrent form (the topic just discussed), with the relevant JobRecord (obviously) first loaded into it.

Upon first examining a JobRecord in this context, you may notice this form's *own* 'Date & Time' box and logically think it reasonable to simply change the appointment notation there. However, this would have no beneficial effect. Operationally, the information in this box has *no effect* elsewhere in the system. To start with, in fact, the information there is a mere historical artifact of the text that existed in the corresponding box of the originating Callsheet, at the time the job was created. Even at the Callsheet, moreover, that box's

⁷⁸ As an example, suppose a technician is on the job and has diagnosed need for a non-stocking part. The customer is very anxious, so the technician calls the office and asks you to order the part immediately. In a more normal situation, the order-request would be initiated via a PostVisitReport (see page 125), but that process won't be completed until tomorrow, when the technician returns with all the required information (including what time he finished the visit, etc.). You need to enter the request now (i.e., get it into the ServiceDesk PartsProcess system [see page 138], so you can manage the ordering, checking-in and so on through it). So in this instance, rather than depending on the normal via-PostVisitReport method for creating the request, it's completely appropriate to use the method provided directly on the JobsCurrent form itself (i.e., load the job, click on the appropriate option and follow prompts, etc.). Other exceptions are similar. If it hasn't happened (or isn't going to happen in a sufficiently timely manner) through a PostVisitReport, go ahead and do it directly from the form. (In the example cited, incidentally, it would be sensible when the PostVisitReport is made for the reporting person to answer 'No' when queried on whether any parts need ordered; in such context that answer would be correct because the process was already done prior to the PostVisitReport being made, as in fact the record would already show.)

⁷⁹ Please note that we've used the word "completed" here somewhat loosely. In general, a JobRecord needs to actually be placed into "Recorded to SlsJrnl" status—and thus truly be done—before it will be moved out of the JobsCurrent file and into the JobsArchived file. This rule, however, is subject to some exception, as detailed in footnote 119 at page 182.

⁸⁰ In addition to these obvious tasks, ServiceDesk takes advantage of the ArchiveJobs event to do some other critical housekeeping—particularly in regard to the CstmrDbase. For a description of such matters please see the discussion that begins at page 307.

information was operational only to the extent that, during the process of creating the job, ServiceDesk pulled information from it and created an entry in the ScheduleList based thereon.

We mention this to emphasize an important point. ServiceDesk has a *ScheduleList* file (see Chapter 6). This consists of a set of entries that describe each appointment that exists at any point in time. Regardless of what other data as may exist either in a Callsheet or JobRecord, it's not going to be recognized as a true appointment (i.e., displayed in your DispatchMap, etc.) until and unless a corresponding entry is inserted into this ScheduleList file. Again, the process is done *for you* when a job is created from a Callsheet that includes a useable appointment description in its 'Date & Time' box. But, from an existing item's JobRecord, there is no further job-creating process to do (for, obviously, the job already exists). Therefore, the context is different, and must be handled differently.

In fact, in early versions of ServiceDesk the method was for a user to simply go to the ScheduleList form and there manually select its 'New Item' option, then insert a new appointment entry into the file, essentially from scratch (which included manually typing-in applicable invoice number, customer name, grid coordinates, etc.). Obviously, that was somewhat burdensome, but the strategy has not been completely abandoned. In fact, all existing-job appointment entries must still be created, ultimately, with the user actually within the ScheduleList form. The difference is that now we've added processes to make it much easier, shepherding you there and filling-in most of the spaces for you.

With that as an overview, we'll now explain the specific processes (which are considerably more simple than the above explanation). Again, begin in the JobsCurrent form with the relevant JobRecord loaded. Then you can follow either of two paths.

First, much as if you were doing the same thing from within a Callsheet, you can right-click on the job's address line to Item-Locate its position to the DispatchMap. Thus you'll see the customer's name flagged there in red, making it easy to determine an efficient appointment time. Then (still, just as though you were doing the same thing from a Callsheet), you can click on the customer's flag, view the drop-down combo-box that's shown,⁸¹ and with its use input the appointment reference, then and there, from within the Map. But now the result will be different from when a similar process was initiated in a Callsheet. There, if you recall, the appointment reference would be inserted into the Callsheet's 'Date & Time' box for you (awaiting operative effect when a job was later created from the Callsheet). Here, the system will immediately take you to the ScheduleList form, where it will have directly prepared a new entry, awaiting only any additional polish you may want to provide (such as indicating an assigned tech in the appropriate box, for example), and your action (simply press Enter) to save it. When you do press Enter, obviously, the new entry will save—and the appointment is made, entered, done! Also (and incidentally), the system makes an entry in the job's History describing the event.

As a second method for scheduling on an existing job, you can bypass the DispatchMap and use an option in the JobsCurrent form labeled '*Scheduling*' (either click on that option button, click on the Date & Time box, or hit Alt-S on your keyboard). This will take you directly to the ScheduleList form. If there's presently a pending appointment on the job, it will present you with an option box, asking whether you want to cancel that appointment, change it, or create a new one. Otherwise (and like the via-DispatchMap method) it will simply present you with a mostly filled-in proto entry. The difference is that here, since date and time were not previously indicated from the on-Map-Scheduling context, the actual appointment notation will not yet have been filled-in. However, the same tools are available here as are provided in a Callsheet's 'Date & Time' box (i.e., DatePicker form, etc.).

⁸¹ If you don't like the set of time-frames that are offered in this list, incidentally, you can easily make a different list. To do so, see the instructions provided at note 67 on page 108.

All in all, it's a very simple process (though to perceive it properly you must keep the *concepts* straight). Also, bear in mind there's a whole chapter on scheduling and dispatch (Chapter 6) that describes many details in regard to manipulating and using schedule entries once they are created (either in this context or from Callsheets, as described beginning at page 70), along with many tricks in regard to special kinds of entries, and so on (see especially the section beginning at page 99). And, if needing to understand the technical details in regard to how ServiceDesk interprets appointment notations, don't forget the Appendix discussion beginning at page 283.

iii. PostVisitReports

If it is to keep a full history of each job (not to mention managing your stocking parts inventory, ordering in non-stock items, tracking receipt and deposit of funds, etc.), ServiceDesk obviously must have a means of collecting information regarding what happened on each and every appointment as dispatched to your techs (i.e., what was done in the customer's home, what parts were used from stock, what parts need ordered, what monies were collected, etc.). Thus, it follows that you must in some manner tell ServiceDesk what happened after every such appointment is filled.

We have a term for this process of informing ServiceDesk about what happened on any given appointment: we call it the "PostVisitReport." You should understand how critical it is in the process for you to make these PostVisitReports—after each and every appointment is filled. Otherwise, ServiceDesk has no means of being informed—and keeping track for you—of what happened on the job (what the tech found, what he did, what parts he used from stock, what he needs to order, what monies he collected, and so on). Really, PostVisitReports are a critical element in the process. They tie a lot of things together; and are the primary means of collecting the raw data via which the system maintains job histories, inventory control, non-stock parts ordering, and funds control. They are the focal point of the interface between you and the computer in regard to the ultimate work of your business: what the tech did in the consumer's home.

Given the central nature of these PostVisitReports, it's important for you to understand the entire setup and context that's been provided for you to do them in—which we'll get to in a moment, but first a distinction. Within ServiceDesk, there are expectations that you'll do certain things after the job has been completed (in particular, recording the completed sale that it entails to the SalesJournal, see page 165). But please note the words used there: "*after the job has been completed*." Please note those words carefully, and distinguish them from the context that's involved in making PostVisitReports—which, by contrast, are expected to occur *after any appointment has been fulfilled*. You see, there's a difference between those situations. On any given job, you might have one, two, three, or who knows how many visits by the technician to a customer's home (hopefully fewer, of course, but we all know how this business sometimes goes). The concept we want you to get here, the distinction, is that ServiceDesk expects a PostVisitReport to be made *after every such visit*—informing it about what happened when the tech was there. Other tasks, tasks that we refer as *Post-Completion* matters, are only done once in connection with any job, after the job is truly complete—complete because, essentially, all appointment/visits that were needed in its connection have been done. PostVisitReports, by contrast, are done after every single visit. Sorry to belabor the point, but sometimes new users have had a hard time getting this distinction. We hope you do now.

So, now that you get it, let's talk about how these PostVisitReports are actually done.

As it happens, ServiceDesk offers you the option of following any of several strategies in this regard. In fact, since there are so many options in *how* it may be done, it becomes slightly complicated to explain how each of the alternatives work, relate to one another, and fit in with the greater whole.

The first complication arises because there are two different on-screen forms that are provided for the purpose, and they each accomplish exactly the same purpose: they just do it in a different way. You can use one or the other, depending simply and solely on your preference. Regardless of which you use, the end result will be the same. Specifically, we're referring to two different *PostVisitReport* forms (which are the vehicle for PostVisitReports). One is the '*PostVisitReport Form: Dialog Method*'; the other is the '*PostVisitReport Form: Fill-In the Blanks Method*'.

If you're curious, the main reason we ended up with two different forms—for the same purpose—is that historically the first was the only one that existed. In late 2002, however, we developed the second, as an improved method for certain situations. The first was still deemed superior for other purposes, however, so it did not make sense to do away with it. And thus, we now have the two alternative forms/methods, existing side-by-side, and you have the freedom to pick which you want to use in any context.

Speaking of "contexts," that's the other complicating factor in this area. That's because (assuming you're more than a one-man shop) you may want to have your technicians make their own PostVisitReports (as absurd as that may sound on its face, there are some very good reasons for doing it this way, and for some operations it will be very beneficial). Or, you may want to have an office person do it on behalf of the technicians. But even then, there are disparate possibilities, for on the one hand you may want to setup for the officer person to do the reports after the fact, probably on the day following, based on the tickets as turned in by the technicians from the previous day's work. On the other hand, you might instead want to setup to have PostVisitReports done in real time, immediately upon the technicians' completions at each job, based on telephone or radio communication with the office. Or, as still another possibility, if you're a one-man operation, you might want to do PostVisitReports in real-time, as you finish each job at the customer's home, based on input then and there to your laptop computer.

It's because of all these disparate possibilities, basically, that our discussion can't be quite as simple as we'd otherwise consider ideal. Still, we'll try to get through it in as basic a manner as possible.

Two Different PostVisitReport Forms

Again, there are two different on-screen forms that are provided for the purpose of accommodating your report back to ServiceDesk regarding what happened at each and every scheduled appointment.

First is the *PostVisitReport Form: Dialog Method*. As earlier indicated, this is the older of the two forms. It's generally accessed by pressing **Alt-F7** on your keyboard. As you'll soon see upon trying this method (and consistent with its name), this form takes you through kind of a *dialog*, asking you a series of questions via which it collects information concerning all the essentials about what happened on the job. Your task is simply (and rather easily) to answer each such question. As you'll see, there are 'Yes' and 'No' buttons provided, upon which you may *click* in any given instance to so answer the question that's presented. Please realize, however, that in the alternative to so clicking with your mouse you may simply hit "Y" or "N" on your keyboard (as the context suggests). Indeed, there are many conveniences provided in the dialog, and if you're observant you'll see that after a little practice it's easy to make it through the dialog with great rapidity.

The second context for reporting, the newer of the two, is the *PostVisitReport Form: Fill-In the Blanks Method*. It's generally accessed by pressing **Shift/Alt-F7** on your keyboard. As its name obviously suggests, this context presents a more static form (rather than a dialog), allowing you at your leisure to fill-in the blanks that relate to the various aspects of what happened on the job (what was done, parts that were used, parts that need ordered, monies that were collected, etc.).

Regardless of which form is used, the very same information is ultimately collected from you. And regardless of which, this information is fed by ServiceDesk into each of the various contexts where it's needed. In other words, the system will place narrative text into the Job's History section describing what happened on the job, based on entries here. It will create PartsRequest records, via which the ordering of non-stock parts may be managed. It will adjust inventory levels in regard to stocking parts. And it will create money-received type entries in your FundsJournal, via which you can perfectly track all your items of incoming money, shepherd them through to deposit, make sure they square with sales, etc. It will do all this on the basis of information collected in these reports, regardless of which of the two forms you use, and regardless of which of the possible contexts for using them.

Speaking of contexts, that is the more specific subject to which we now turn.

In-Office Reporting: Following Day Method

Probably the most common method in use among present ServiceDesk users is that, on one day, the technicians go out with their set of jobs. That evening or the following morning, they turn in the tickets/invoices from that work. Upon those tickets, they're written in pertinent information regarding what they did on the job. At this point someone in the office, someone assigned to the task, takes that stack of tickets, sits down in front of ServiceDesk, and makes the PostVisitReports, based on what the techs have written upon the tickets.

In terms of going through the sequence, there are several ways it can be done. If, for example, you're using the newer PostVisitReport form, you can bring it up directly by pressing Shift-Alt/F7, then simply type in the invoice number of each job on which you're reporting, as prompted. Of, if you've already brought up the JobRecord of interest within the F7 form, and *then* you hit Shift-Alt/F7, the system will assume that's the job you want to report on, and load with that as the selected item. Or, if you're in the DispatchMap and looking at the reference to any given appointment, you may from there do a **Shift-Alt/Rt-Click**, and the system will QuickLink to the newer PostVisitReport form, while pre-loading with the expectation that it's the clicked-upon appointment on which you're intending to make your PostVisitReport (if you wanted to use the older, dialog method, do a mere Alt/Rt-Click instead).

All in all, if you do have a stack of invoices on which you need to make PostVisitReports, it's probably easiest to simply strike Shift-Alt/F7 and go through the sequence for each job, one after another (or do the mere Alt/F7 if you prefer the older, dialog method).

There's nothing wrong with this method. It works very well. There's also nothing jazzy about it. It's pretty normal and typical, which is probably why it's so popular.

Regardless of whether you decide to use this or the other In-Office Reporting system (about to be discussed), you'll probably find that you prefer the newer method of making your PostVisitReports (i.e., based on the newer *Fill-In the Blanks* (Shift-Alt/F7) form, rather than the older *Dialog* (Alt-F7) method. But please try each regardless, so you have an understanding of the alternatives, and opportunity to consider which of the two you really prefer.

In-Office Reporting: Immediate Call-In Method

This method is solely of interest to multi-tech operations, so if you're a one-man shop, you won't need to concern yourself with its possibilities.

The concept is simple. You just have your technicians call-in to the office as they finish each job. An office-person, speaking with the tech, brings up the preferred PostVisitReport form (whether Dialog or Fill-in-the-Blanks method), and completes a report based on the information he gives her.

We have a number of clients using this system, and they positively swear by it. To glimpse just part of its advantage, suppose Mrs. Jones calls your office ten minutes after the tech has left. Within two seconds, you can have the technician's full report in front of you—allowing you to speak with her intelligently (rather than saying, for example, I'll have to get back to you tomorrow after I've had a chance to speak with the tech). Suppose it's one of those calls where she's saying "He was only here five minutes, and blah, blah, blah." With the benefit of real-time information, you can instantly reply: "Well actually, Mrs. Jones, I see here that he started at 12:10 and finished at 12:55, which by my calculation makes 45 minutes, not just five." (I've personally had almost precisely that conversation several times.) Of course, that's just one kind of benefit. If parts need to be ordered, the process can be initiated that much sooner. If you're doing work for home warranty companies and need to get authorization, that can also be expedited, and so on. Overall, it allows you to provide a significantly higher quality of service to your customers—on top of giving you the serenity of knowing in real time what happened on each job.

If you're already investing in the employee-time that's involved in having your tech's call in as they arrive and/or depart from each job, and are doing so solely for the purpose of keeping tabs on where they're at (see page 87), it would be almost silly not to add the major benefit of real-time PostVisitReports, based on just a little more employee-time than you're paying for already. Even if you're not, it may well be worth making the investment.

To implement the system, you should first go to the ServiceDesk *Settings* form (Ctrl-F1) and, in the purple section, check the option titled '*Do Immediate PostVisitReports as Techs Call In*'. This lets ServiceDesk know this is the procedure you're using. On that basis, ServiceDesk knows to assume, when a PostVisitReport is begun, that the current clock time is, in fact, the technician's completion time on the job. Thus, it knows to fill-in that time, and not force the operator to do so.

You should then consider whether you're going to have your technicians call-in only as they complete each job, or both on completion and upon arrival. You can do it either way. The advantage of adding *arrival*-call-in is that the system can use this event to log the technician's start time (based on the computer's then current clock time). This saves effort when the technician later calls in his completion, because now the system is able to deduce for itself both start and end times, and can place them both into the history with no separate input from the operator. Regardless, there's no requirement for you to be consistent on the matter. Just make sure, if and when your technicians call in their arrivals, that the call-taker knows to check it off as such within the DispatchMap (i.e., do a Ctrl/Left-Click on the job's reference there; see page 87 for a more complete discussion of this function)

Regardless of whether you've separately checked-in the technician's arrival on a job, when he calls in his completion, DO NOT so check it within the DispatchMap (as you would, in other words, *if* that was your only purpose). Instead, initiate the PostVisitReport. You can do this in any of the standard ways (i.e., bring up the form directly and type in the invoice number, or first locate the applicable JobRecord in the F7 form then hit Shift-Alt/F7 for the PostVisitReport-Type2 pre-loaded with that particular record). However, for this purpose it's likely easiest to Quick-Link in each instance from the DispatchMap. From there, you can simply locate the job's applicable reference (List or Graphic), and do a **Shift-Alt/Right-Click** on it).⁸² In result, the system will

⁸² Bear in mind that all these kinds of commands are available in convenient reminder format from within the DispatchMap itself. Just **right-click** in any empty space there and you'll instantly see the applicable ContextualCommandSummary. It's nicely categorized, to help you easily find the command/action you're looking for. In the case of Quick-Linking to the PostVisitReport, for example, you'd look under the section titled "QuickLinks." While there, you might notice that, in addition to the Shift-Alt/Right-Click action as discussed in

instantly display the PostVisitReport interface, pre-loaded with applicable job, appointment, and start and end times. Thus, it doesn't typically take very long for your office person to collect the rest of the needed info from your tech, and for him to be on his way.

Another benefit when using this method is that, if anyone in the office has addressed email to the tech who's calling in, there's a button on the right-hand side of the PostVisitReport form (Type 2) that will show itself conspicuously in red (otherwise, if no email is pending, the button will be dim and disabled). The purpose is to catch the operator's eye, alerting her to the fact that some message needs to be brought to this tech's attention (perhaps regarding a cancellation, request from the boss to speak with him, or whatever). She can then simply click on the button, peruse the email, and inform the tech in regard to its contents. She could even type a reply back on his behalf, if he so requested.⁸³

Still another benefit concerns ServiceDesk's **Arrival On-Time Sentry**. This feature continuously monitors each technician's progress within his route (based, of course, on your work in checking off his arrivals and taking PostVisitReports upon each completion). If at any point it appears he's threatening to arrive late on a job,⁸⁴ the system will put that job's DispatchMap references in to a flashing/blinking mode. The purpose is to bring the peril to your attention in the office, so you can take appropriate remedial steps. It's a very powerful and important tool.

You may notice, when you're doing PostVisitReports via the immediate call-in method, the DispatchMap becomes even more of a command central—at least for the office person who's managing dispatches and the techs' call-ins. He or she will center almost all work from there, while keeping perfect and easy track throughout the day of the status on every job, and the position of every tech. Also, the manager at various points in the day the manager/owner can glance at the DispatchMap at quickly get a sense for how the techs are doing: how they're faring within their schedules, their ration of completions to non-completions, and so on.

All in all, we highly recommend this immediate call-in method for doing your PostVisitReports—if you can make the necessary personnel available in real-time, as required for taking the calls from the techs. Do be assured in this regard that when done properly, the time investment is quite moderate. Most PostVisitReports, done this way, should be completed within less than a minute.

Having Techs Make Their Own PostVisitReports, Locally:

Although somewhat less sexy, there is a typically underappreciated method that we think has great merit. It's what we did in our own former service office, and it worked very, very well. Most people, we've found, think they could never get their techs to go for such a method (or that their techs would do a horribly

the text (to link to the PostVisitReport-Type2), you could in the alternative to a mere Alt/Right-Click, which would instead link you to the PostVisitReport-Type1 (i.e., the older interface). We don't even mention that in the main text, because we think the newer interface is by far superior for the purpose discussed, but the option is there, and you'd see it just be reviewing that ever-handy, don't-forget-about-it, ContextualCommandSummary.

⁸³ If using this feature, obviously, you'd want to be careful to avoid placing anything of a private nature in an email as addressed to one of your techs, for it would *likely* end up being read by one of the operators—possibly resulting in embarrassment to both her and the tech.

⁸⁴ The system uses a simple formula to determine the threshold at which the flashing will begin. It simply takes the size of the time-frame scheduled, divides that by 6, adds 30 minutes, and calculates upward by the resulting amount from the end of the scheduled time frame. Thus, an appointment scheduled for between 9 and 12 would end up with a "flashing threshold" of 11:00 o'clock (one-sixth of 3 hours is 30 minutes, plus 30 minutes = 1 hour, calculated back from 12:00 o'clock makes 11:00 o'clock). An appointment scheduled for between 12 and 6 would produce a threshold of 4:30 (one-sixth of 6 hours is 1 hour, plus 30 minutes = 1.5 hours, calculated back from 6:00 o'clock makes 4:30). If the calculation would result in a threshold prior to the beginning of the time frame (as would happen with a very small time frame), the threshold is set at the beginning of the time frame (i.e., in spite of the formula's standard result otherwise, an appointment from 9 to 9:30 will have a flashing threshold of 9:00 o'clock).

lousy job), but we urge you not to sell them short. Another concern is security. People fear letting the technicians into the system. But, as you'll soon see, we have fully met those concerns.

Some of the benefits in having techs make their own PostVisitReports are as follows. First, communications with them are enhanced: by logging in every day, they have perfect access to E-Mail communications, and can respond in turn. You can easily post specific messages to a particular tech, or general messages to all of them. It's easy, and somehow carries more impact than mere notes left for them to read. Second, when they need information about a past job, there's no longer a need for them to ask you or another office person to look it up: they'll simply find it themselves in the CstmrDatabase. Third, because of the query-dialog format of the PostVisitReports process, they're forced to answer questions which, if merely filling-in blanks on an invoice, they might have failed to respond to. And fourth, all problems with office personnel attempting to decipher a technician's handwriting (when reading critical items like Model and Serial Number) are ended: the techs enter such info directly by keyboard, ending all ambiguity.

Anyway, assuming you're persuaded of the benefits in having your techs make reports on their own behalf, you'll first need to assure there's a computer station they can use for this purpose (i.e., one no one else in the office is needing at the same time). Typically, it's beneficial to set aside a machine that's exclusively for their use in this regard. An excellent choice would be an older computer that's not quite up to the performance levels wanted for other ServiceDesk applications. To setup this station, run ServiceDesk just as you would from any other networked computer (or use any networked computer on which ServiceDesk is already running). Then, from that computer's main Callsheet interface, press **Alt-W** (for Window/tech-mode). At this point, ServiceDesk will switch itself into a mode that is specially tailored for direct interface with your technicians.⁸⁵

More specifically, it shows a full screen displaying the ServiceDesk logo. When any key is pressed from this window, the TechInterface form appears, and from it ServiceDesk queries for the initials of the operating technician. When acceptable initials are entered, the technician is greeted by name, and invited to read any pending intra-office E-Mail. In the absence of (or at the conclusion of reading) E-Mail, ServiceDesk displays the PostVisitReport form (in this case, it is exclusively the older, dialog-method, for it's simply much more appropriate for technician usage), and the technician is urged to report on each of his jobs. Upon exiting from the PostVisitReport form (hit Esc), the technician is informed if there are any outstanding items he has failed to report on, and if so, is urged to go back and make the reports.

Both before and after making his PostVisitReports, your technician is able to take advantage of a few other utilities. First, in addition to reading his own E-Mail, he can compose and send E-Mail to others within the office or technical staff. More significantly, he has the ability here to use the CstmrDatabase, looking up past or current jobs, just as you may do. Thus, without bothering you, he can learn things about the history and status of jobs that he may need to know (though he's locked out from doing any edits). Plus, he can personally review (but is, again, prohibited from editing) the DispatchMap, thus allowing him to see his day's route graphically, preview what's on the schedule in coming days, review past day's schedules, and so on. Also, if your technician is paid on an hourly basis, he can use ServiceDesk's TimeClock feature, to clock-in or out of work. The procedure for these functions is straightforward, and obvious on the form itself.

When your technician is done conducting his business, he should repeatedly hit Esc until exiting all the way back out to the logo screen (i.e., with one Esc he exits from the PostVisitReport form, with the next

⁸⁵ If you've got a particular computer station that you intend to have used for this purpose, you can select a feature in the 'Settings' form that will cause ServiceDesk to switch automatically into the TechInterface/Window mode, immediately upon startup. If you also have Windows set to start ServiceDesk automatically upon bootup, your technicians can be given authority to turn that computer on themselves, and all they'll ever see (or have a chance to mess around in) is the particular TechInterface mode where you want them.

from the TechInterface form, and so on). At this point he's essentially logged out from his session, and ServiceDesk is ready for the next tech.

As another security feature, incidentally, you should know that once switched into its TechWindow mode, ServiceDesk will resist any and all efforts to move back into the main ServiceDesk environment—except by use of your Owner/Manager Password. The reason is to keep technicians out of mischief in areas they have no business, and to assure they don't even inadvertently switch out of the mode, which, if it happened, might prevent subsequent technicians from having access to the mode they need. Thus, even while giving technicians the access they need to report on their own jobs, you're able to maintain your own data security.

You might think technicians would object to having to do this little item of work, but it's our experience that if they simply understand it's a part of their job—specifically, that their own work on any particular job assigned is not done until they make this report—there's little or no problem. Indeed, even if you pay them solely on a commission basis, if they simply understand that they've not earned their commission until and unless they make this report, you should find there's little problem.

Having Technicians Make Their Own PostVisitReports, Remotely

Still another option is for your technicians to make their own PostVisitReports, remotely, from their own home computers or laptops. To do so, obviously, they'd need to log into your network via Dial-Up Networking or via the Internet (see page 310). This situation is especially ideal for technicians that live somewhat remotely, and don't come into the office every day, but may even handy for the owner/tech who just wants to report on his own jobs from the convenience of his own home. Once again, security concerns are obvious, but there are effective ways of handling them (if interested in this method, please read the above-referenced section in the Appendix, and if more questions remain, contact us).

An Option in terms of Operating Setup: To Integrate PostVisitReports in a Seamless Sequence with Post-Completion Tasks

In a typical office setup, many items of work are done in a *batch* manner. If you're using the most typical method for handling PostVisitReports, for example, you'll have an office person sit down with the stack of invoices as turned in by the techs from the previous day's work. She'll go through the whole stack, making PostVisitReports on each, one after another. Either at the beginning or end of this effort, she'll separate the tickets on which the job's are done from those on which parts need ordered (or return visits are otherwise still needed). The latter stack will go to whatever place you've set aside for those purposes, while the former will now feed to another batch process. In particular, if you're a servicer doing warranty work, that stack will be reviewed by someone for any warranty claims that need to be made, and that person will then go through the stack, making such claims—doing that added step, for all such invoices, again as an entire batch. Then, and finally (or perhaps this step would be done before the one just described; it's optional), the stack of invoices will be run through still another time, again by someone in the office, for the purpose of recording each, as a completed sale to the SalesJournal.

That's typical, and is probably best for a larger office, where specialization of tasks, separating different areas of responsibility/accountability, and so on, can result in increased efficiencies. If yours is a smaller office, however (where it's essentially the same person doing all the above tasks anyway), you'll may find another option preferable. The difference, essentially, is that instead of doing Step1 on a batch of invoices, then (when that's done) doing Step2 on the whole batch, and then (again, when that's done) doing

Step3, you'll instead do Steps 1 through 3 on a single invoice, then the whole sequence on another invoice, and so on.

Facilitating this kind of integration was one of the reasons we created the *new* PostVisitReports form. The integration is available exclusively from it (*not* from the old dialog-method form).

To make the integration work, all you have to do is check-off an option box within the new form that's labeled "*when job is done, link automatically to post-completion tasks.*" With this done, the system will watch as you save every PostVisitReport. If you've indicated (as part of the PostVisitReport) that the job has been completed, the system will at this point query as to whether you'd like to link immediately to either the FinishedForms (appropriate for making warranty claims) or SalesEnter (needed to close-out the completed sale) contexts. Regardless of which you choose, the system will link you immediately to that context, with the maximum possible quantity of information carried there for you, from the context you were just in. This saves effort because, among other things, there's no need from within those contexts to separately indicate the item you are reporting upon. Instead, the system advance loads data for you, since it knows in advance what you want.

Also, if you've linked first to the FinishedForms context (as needed, for example, to make warranty claims, or perhaps if you simply prefer to print a final invoice with every detail included), the system will, upon completion of your work there, offer to link further (and automatically) to SalesEnter context, where in this case it's able to fill-in the entry line for, in its entirety, in advance. All that's left for you is simply to confirm by hitting Enter.

If suitable for your office setup, this method of integrating the PostVisitReport with Post-Completion tasks can be very powerful, adding enormously to the ease and convenience of completing these processes. You'll simply have to decide whether the approach is best for you.

The One-Man Shop: "Has Laptop, Will Travel"

A final possibility (or at least the last we've thought to write about here), is if you're a one-man shop, and if you wish to essentially carry your office with you, via mobile electronics. For this situation, we *highly* recommend the integrated approach just discussed. Indeed, we'd suggest going further, in a sequence you might call "Integrated-Plus."

Specifically, we'd suggest not even printing advance ticket/invoices, like most organizations do. Instead, from the DispatchMap in your computer, identify the next job on your schedule. Do a quick-link to the JobRecord (Ctrl/Rt-Click), to get the full address and particulars, then drive there. Upon arriving, press Shift-Alt/F7 (from that same JobRecord, still displayed), to bring up the new PostVisitReport form, pre-loaded with the present job. Hit Enter to confirm it's the present appointment you're reporting on, then (since the system will at this point have already placed you in the 'Start Time' box, simply type in "now," and the system will insert the correct time for you. Then go into the house and do the job. When you finish, return to your truck and again type "now" (in the 'End Time' box, where the system will have already placed your cursor). Then fill-in the rest of the form, and save. After that's done (and assuming the job itself is also complete), you can link automatically to the various Post-Visit activities (as above-discussed), and (on your mobile printer) instantly print a perfect, finalized invoice for your customer. Wala, yippee, hurray. Couldn't be more efficient or easy.

For the Future

We're guessing that in a very few years the vast majority of technicians will carry wireless "Palm"-type devices, via which they can type-in their own PostVisitReports while finishing each job, for immediate and direct transmittal to the home office—where the information will then be available, in real-time and with no separate office effort required. Here at Rossware, we intend to be on forefront of this development. Watch for it in future upgrades.

A Bit of Reflection

In designing ServiceDesk, we've been mindful of the fact that many software systems require the input of data you would never otherwise have bothered with. In result, they end up creating even greater office burdens than they were supposedly designed to eliminate. Being determined to avoid this error with ServiceDesk, it's with a little sadness that we must admit there is a small amount of new work in the PostVisitReporting process. Even so, if you'll try the process, you'll find the amount of effort is minimal, and the payoffs are absolutely immense.

In result of the data it collects from your PostVisitReports, ServiceDesk is able to keep a complete on-line history of every job, a history that is instantly accessible to any person in your office, at any time, and at the press of a few keys. Thus, ServiceDesk is able to assist you in tracking your jobs, to assure their proper progress and completion. It facilitates the process of ordering non-stock items, logs their receipt and so on. It keeps an up-to-the-minute inventory of precisely what items of stock are on each truck (and in the shop) at every point in time. It keeps perfect track of your funds collected, facilitates deposits, and on and on. All this is based on just a little effort in entering information regarding what happened on each dispatched job.

iv. The WipAlert System

Almost from its inception, one of our purposes in designing the WorkInProgress system was to create an automatic sentry that would in some manner monitor each of our outstanding jobs, and alert us if some item began to fall through the cracks because of inadequate attention. There is, as you know, already a means to do something similar in regard to Callsheets (i.e., the Callsheet alarm system, see page 68). However, that particular sentry is designed to help you avoid neglecting things like returning someone's call, getting a dispatch scheduled, or acting on a simple "To-do" note for which you've created a Callsheet. It does not pay attention to WorkInProgress (i.e., actual pending jobs and their status).

Indeed, a sentry that would perform this latter task is considerably more complex in concept, and had to await development until most the other features in ServiceDesk had been invented. It is, therefore, one of our more recent additions. We call it, simply, the WipAlert system.

To implement this new sentry, you must select a particular computer station on which you wish to have it operate (your head secretary's station is probably the best choice). Designate this station (so ServiceDesk is aware of your choice) by going to its own Settings form. In the LocalSettings section, you'll see an option titled "*Send WipAlerts?*" Check this and then save your settings (remember, only do it from this one station—unless, of course, you want to have multiple copies of the alerts sent elsewhere).

Within about one minute of turning this feature on, an invisible routine will run, checking every item in your JobsCurrent file. For every item that appears to be past-due for attention, the routine will create a Callsheet informing you of the fact. In response, you may then inspect each such item's JobRecord, to see

what kind of attention is due.⁸⁶ As you give each such matter the processing attention it needs, simply document the fact in its History.

Suppose, for example, that in response to a WipAlert in connection with some job, you've checked on a part that is long overdue in coming, and called the customer to assure you are doing everything you can to expedite its arrival. Simply append a note in the item's History indicating this action. Or, if it was simply an item that was past-due for its PostVisitReport, or for recording to the SalesJournal following its completion, or some other such thing that automatically updates an item's History when finally done, simply take that specific needed action (which again, the WipAlert will have helped you avoid failing to do). In either case, with the job so updated, the sentry system will now be satisfied that it's not being neglected, for some time at least.

The amount of time the sentry system allows you, before again thinking there's neglect and therefore creating a new WipAlert, depends on the Status the job is in. If it's in "ATTEMPTING TO SCHEDULE" status, for example, the system figures you ought to be making at least daily attempts to contact the customer, and of course documenting such attempts in the item's History (this refers to scheduling after a part's come in, or after you've been stood up and therefore need to re-schedule, or something similar, since initial scheduling of every job is managed and policed from within its originating Callsheet). Therefore, if the last History entry of an item in such status is more than about a day old, you'll get a WipAlert on it. If an item is in "WAITING FOR PARTS" status, on the other hand, the system figures that at least three days might be an appropriate wait before an alarm needs created. Each status has its own such alarm period, as seemed appropriate to us in designing it. In figuring the number of days in a period that elapses across a weekend, incidentally, only Saturday is figured (thus, while Monday to Thursday counts as 3 days, Friday to Monday, the same actual number of days apart, counts as only 2).⁸⁷

After the WipAlert routine first runs, it logs the fact that it's run for the present day, and will not run again until it detects a new date. If you keep the station running overnight, therefore (we think this is most convenient), the routine ends up running itself precisely at 1:00 am the next day, and you (or whoever works at the station involved) will have fresh WipAlerts waiting when you come in later that morning. This has been working very well in our office.

⁸⁶ We've created a shortcut, for this context only, that allows you to instantly bring up the particular JobRecord that pertains to any particular WipAlert. Simply **right-click** in the CustomerName box of the Callsheet that contains a WipAlert (in the alternative, you may **right-click** in the Callsheet space that's between the two CustomerTelephone number boxes; or you may **double-left-click** on the CustomerName box—both of which are alternatives developed in the same vein as those discussed beginning at page 328). This will work only if the CustomerName box has text beginning with the statement "WipAlert:". In such circumstances, ServiceDesk will instantly display your JobsCurrent form with the appropriate record loaded for you.

⁸⁷ The number of days allowed for each status, as a grace period, is set by default within code, but may (if wanted by the user) be customized. The standard defaults are as follows (in number of days, not including Sunday):

Working to Schedule	1
Currently Scheduled	10
Dispatched to Tech	2
Tech Reported, Not Done	1
Waiting for Parts	3
Pending Authorization	5
Other	4
Completed	7
Recorded to SlsJrnl	30
Marked for Deletion	30

If you want a different grace periods than these, the procedure for setting up your own grace periods is very simple. You simply need to create a text-file type of document (use any word processing program, such as WordPad, for example) that has 10 lines of text, each such line consisting of a single number, that number corresponding to the number of days that you want for each of the above categories of status (in the same sequence). Save the document as 'GracePeriods.TXT' in the \sd\netdata folder of your FileServer drive. When ServiceDesk boots up, it looks in that location for a file under that name. If it finds one, and if it's formatted correctly, it pulls the numbers found and substitutes those as grace periods in place of its own default ones.

Actually, it worked very well until our primary secretary got a little lazy (or perhaps just too preoccupied with other work). At any rate, for a month or more, she simply deleted her WipAlerts, without properly using them as reminders to do necessary follow-up work. One afternoon, in consequence, the boss got a call from an angry customer. On reviewing the JobRecord pertaining to this customer's job, he saw that more than a month had gone by (waiting for a part) without any of the necessary reminder calls, to the customer, indicating that we were doing all we could to further the job along. Yet, the secretary had received daily reminders (specific WipAlerts pertaining to this very job), essentially telling her such calls needed made. After appropriate scolding, the boss decided he needed something that would alert *him*, should his staff every again disregard the WipAlerts in such manner.

Thus, as a monitor on the monitor (so to speak), we now have a WipAlert-*Supervisor* feature, which can similarly be "turned on" (it too is a 'local' setting) from the Settings form. Essentially, at 1:30 am each morning it runs the same checking process as does the standard WipAlert feature, but allows two additional days idle time before generating an alert (thus, the person who responds at the primary alert level would have to fail to respond, for at least two days, before this next-level alert is generated). And, of course, this alert sends its notifying Callsheets to the desk of the person who has this feature turned on (not to the subordinate), and the alerts themselves have text alerting to the higher degree of warning. Our secretary knows of this new feature, and knows the boss has it "turned-on" at his desk. He has yet to receive an alert.

Obviously, if you're not interested in the WipAlert system, there's no necessity for you to use it, but we think you'll find it a most valuable tool. During the 18 month period our office used a beginning WorkInProgress system that lacked any WipAlert function, we found ourselves accumulating a large number of old JobRecords where performance had simply been dropped, somehow ignored, the invoice lost or misplaced, perhaps completed but never billed and recorded to sales, and similar failures. Back in the old days, before there was *any* JobRecords, these jobs would have been lost entirely, without our even knowing it. And with a basic, un-sentried system, they accumulated in our JobsCurrent file, with little attention being paid (the file was began to bulge from such items before we even realized it). Now, however, we're happy to know that whenever even a single item even *begins* to slip through cracks, we'll know about it rapidly, and almost be forced (the system can be a pest until satisfied) to deal with it quickly.

v. The JobsArchived Form

Like many other complex systems, ServiceDesk has evolved since its initial creation and in the process has sometimes retained vestigial remnants of older parts, even while they were largely supplanted by newer creations. The JobsArchived form (press **Ctrl-F7**) falls somewhat into this category, for while it was initially created to provide search and review capability in regard to *archived* JobRecords, the CstmrDbase feature now fulfills this need more ably and easily—at least for standard searches. You may, therefore, find there's seldom need to use the form's own, built-in lookup features (at least the standard ones, though they are still there and can be used if when wanted).

However, it may be useful for you to know that, even in the CstmrDbase context, you will be continuing to make at least ancillary use of the JobsArchived form, for it is in fact an abbreviated instance of this form that ServiceDesk uses to display each of your CstmrDbase findings, even when conducted from other contexts. More importantly, you should know about a few utilities, accessed from the JobsArchived form, that are not available elsewhere—ones you'll find useful on at least limited occasion.

First, the JobsArchived form allows you to view your archived JobRecords on a *record-by-record* basis (i.e., PgUp, PgDn), which may in some circumstances be more useful than being able to search only for specifically matching items, as from a CstmrDbase context.

Second, there are limited opportunities to *edit* an archived JobRecord from within the JobsArchived form (just click on what you want to edit, then on-screen prompts will guide you). This is sometimes important. Suppose, for example, you discover that you had an address wrong, but not until after the job's record was archived. You want it right in there, though, because when the same customer calls again, you're likely to create the next order based on the data set from the old. Using the JobsArchived form, you can make such a correction.

Third, you can search for a record by *P.O. Number* from within the JobsArchived form (much as you can from within the JobsCurrent form, the difference being which of the two files you're searching in, whether among archived records or current ones). This is not something you can do at all from the CstmrDbase context.

Fourth, you can search for a job by its location *street name* from within the JobsArchived form. It may seem odd to those not in a service call-performing business, but it simply happens sometimes that you're trying to retrieve some information from a past job. You can't remember the name, telephone number or similar info, but the tech can remember the street. With the JobsArchived form it's simple to find all jobs performed on that street (no similar utility is provided in the JobsCurrent form, incidentally, so such a search cannot be done on current JobRecords, and of course there's no ability to do it within the CstmrDbase context).

Fifth, you can search for a job by *invoice number* from within the JobsArchived form. This must be distinguished from a somewhat related facility in the JobsCurrent form, where you can *instantly* locate a *current* job by invoice number (instant because the system uses a logic engine there, based on the fact that the current invoice numbers are always in-sequence). With this feature, by contrast, you can locate archived invoice numbers via a *record-by-record* search. Basically, in any of these special searches ServiceDesk begins by looking at the most recent archived JobRecord, then the one before that, then the one preceding that, and so on, until finding what you're looking for. Finding records that are a significant distance back, therefore, can take considerable time.⁸⁸

Sixth, it may sometimes happen that your CstmrDbase indexes become corrupted, inaccurate or incomplete. If there are items you know should be coming up in a CstmrDbase Search but are not (or if you click on a reference and the wrong record displays), you'll know this must have happened.⁸⁹ When it does, you'll want once again to make use of the JobsArchived form. On it you'll see there's a command button labeled "Make New Index." When you press this, ServiceDesk will run a procedure that creates brand new CstmrDbase indexes for you, beginning from scratch. Thus any corruption that may have crept into these

⁸⁸ In many cases you'll find it faster, if you know the name on the job, to do a CstmrDbase search (press **F12**) instead; if there are only a few entries under the name, peruse through them until finding the job which pertains to the invoice number you're seeking. Alternatively, if you do not know the name you can probably find a SalesJournal entry on the basis of a SalesViewForm-based invoice number search (press **SHIFT-F3**), and much more quickly than if searching for the same number, within the JobsArchived file. Once the reference is located (generally quite fast in this form, even if very far back in time), you'll have a name, which can then be used in the CstmrDbase format to almost instantly locate the archived JobRecord in question (again, assuming there are not too many CstmrDbase references under that name; if there are, you may be forced to use the JobsArchived form's slow, Invoice-Number-Search feature).

⁸⁹ One possible cause, for example, would be if one of the users in your system archived the JobsCurrent file (remember, ServiceDesk updates the CstmrDbase indexes at this time also, see page 123), but failed to follow ServiceDesk's instruction in copying the resulting new indexes to other stations. The problem would be compounded further if one of those non-updated stations, say a few days later, performed a ArchiveJobs routine of its own. With its own indexes out-of-date to begin with, the new revisions would result in an index set that's in even worse shape—which the station user might then dutifully (but unknowingly in regard to the defect), copy to all other stations.

indexes (as they're updated from time-to-time via the ArchiveJobs routine as run from JobsCurrent form) will be eliminated (see page 283 for a full technical discussion).

Again, you may still search for a job within the JobsArchived form by customer name, address or telephone number, but these functions are much more powerfully addressed from any of the various CstmrDbase methods (see page 66 for a summary). Another advantage of a CstmrDbase search is that the JobsCurrent file is searched and accessed at the same time (except items added since the most recent index update, see page 283 for a technical explanation). For reason of these differences, we think you'll rarely wish to use the JobsArchived form for its more limited version of such functions. For those other functions discussed, however, keep it in mind as a most useful tool.

B. Managing Special-Order Parts

It may be that some service companies have little burden in regard to ordering non-stock parts. For the rest of us, however, the burden is frequent and substantial. Please be assured, ServiceDesk has a well-polished system to manage this activity.

First, a definitional distinction: *stocking parts* (aka “inventory,” and as specifically discussed in the *next* major sub-chapter section) are items you acquire and hold with the expectation you'll likely use them on *some future* job (i.e., when first acquired, they are not for any particular job, and you have no specific immediate use for them). By nature, they are speculative. *Special-order* parts, by contrast (and as discussed in *this* major sub- section) are never held as “inventory.” They are acquired for specific jobs (or perhaps to fulfill particular POS requests)—specifically, when it's realized, on any such particular job (or POS request), that said part is needed, and further realized that, because it's *not* a stocking item, it must be specifically ordered for immediate need.

i. Birth of the Internal “PartsRequest”

The first thing that must happen, in managing a special-order part, is for an internal “request” to be generated. This request can be created via any of several mechanisms:

1. A tech in SD-Mobile makes the request, via his Mobile interface;
2. A tech makes the request via a PostVisitReport, as entered (by him) directly within ServiceDesk;
3. An office person creates the request, via a PostVisitReport, as performed on behalf of a tech;
4. An office person creates the request via a POS operation; or
5. An office person creates the request via a button as provided within the JobsCurrent form, and as connected with the job then displayed.

Regardless of how created, each of the parts requests you have *pending*, at any moment in time, are stored in a particular location, and can be accessed via a particular form. Sensibly, this is called the *PartsRequest* form (**Alt-F8** is the shortcut). Besides being a place where requests can be viewed and/or

edited, it's more typically used as the interface (in some but not all of the above-enumerated contexts) where requests are actually created.

So, the simple idea is, before we can manage a special-order part, we first have to have an underlying, internal request to form its basis. This request is an internal record that describes details about the request. It relates only to special-order parts, and not to stocking inventory. It's an internal record that says, essentially: "Hey, here's an item we don't stock that we need to get, because it's needed on XYZ job." And, it provides the underlying, request basis for initiating the processes of making inquiries to suppliers, actually placing an order, keeping track of the order, checking in the part when it arrives, etc.

The request is the foundation. For parts connected with jobs (as opposed to POS-based requests), it's typically created via any of mechanisms 1 through 3, as above-listed (i.e., in a *PostVisitReport* of some kind, whether via Mobile or internal PVR Types I or II). All three PVR contexts have their unique methods to collect what is, ultimately, the same result: an internal parts request, representing an item that needs to be ordered—or, at least, inquired upon.

In regard to this latter, the system recognizes that sometimes an item simply needs to be researched, to determine price and availability, pending a decision as to whether an order is desired, or not. The underlying "request" accommodates this via an option to indicate whether the order is "Definite" (meaning just get it regardless) versus "Tentative" (meaning research-only for now).

ii. Processing PartsProcess Items—General Concepts

So, you've got several mechanisms to create a *PartsRequest*, plus a form (the Alt-F8 *PartsRequest* form) that holds each such request, reviewable on a page-by-page basis (i.e., each page of display in this form represents a separate underlying request). But, obviously, giving birth to these requests and storing them does not accomplish the ultimate purpose. We also need a basis to perform the underlying, needed work (i.e., looking up the parts, getting them ordered, etc.).

The *PartsRequest* form, in itself, would be a lousy tool for facilitating these further purposes. It's not designed for them.

For perspective, consider an old-fashioned, paper-and-ink managed office. In that "old days" scenarios, it's typical that each part-request is represented on a sheet of paper (often it's the service ticket itself, but some old-method offices use other forms). It was also typical that, at some point in the day, the person responsible for parts-processes would gather each of the slips representing new parts requests, and lay them out on a large flat surface. Essentially, he wanted to get a "feel for the territory," sort of assembling and re-assembling the slips, saying to himself: "Okay, these three items I'll go to Vendor X for, and these four to Vendor Y," etc.

In an all-electronic system such as *ServiceDesk*, the same guy (or perhaps gal) is going to want something equivalent to that "large flat surface."

That equivalent, in *ServiceDesk*, is called the *PartsProcess* form (shortcut is **F8**).

Please notice, between the two shortcuts so far described (Alt-F8 and F8), the *PartsProcess* form has the easier one. We gave it the easier shortcut because it's your main, operative form—the one where you really do the bulk of your parts-process work.

The general idea, in the F8 PartsProcess form, again, is to provide the electronic equivalent of that large flat desk—or, actually, *several* of them.

If you're an average size shop, you'll likely have a few scores of parts-request items pending at any moment in time. Some of these requests will be brand new, with no process work (inquiring with vendors, placing orders, etc.) having yet been performed. Others may be in a state where you've made inquiries with particular vendors, and are waiting for a response back. Still others may be in a state where an order has been placed, and you're waiting for the shipment to arrive. There are still more possibilities—such as that you've looked up and got price and availability, and are waiting for the customer to give you a yea or nea.

What if, in a paper-and-ink system, you had a *separate* large tabletop on which to spread the tickets holding parts requests that fit each such category (i.e., one for requests on which nothing's been done at all, one for requests where you're waiting for a response back from the vendor, and so on)?

That's the general concept behind ServiceDesk's PartsProcess form. It gives you several "virtual" tabletops—one for each category of progress in which your underlying parts request may lie. On its first display/menu page, you pick the particular "tabletop" you want to work on.

PARTS ORDERING AND INQUIRY (CURRENT FILE)

Mini-Manual		Description of items needed				Inquiry/Order		Info Provided		Order Status						Import Shopping Cart	
Ref #	ItemTp	ItemMk	Model #	Serial #	Rqst	Vendor	W/hom	Mthd	Date/Tm	Confirmed	Expected	Received	InvNbr	\$ Cost	\$ Sell fo		
Cstmr Nm		Part Description	Qty	Instr		Part #	Avibity	\$ whsl	\$ retail	PO Nbr		Notes			BinLoc		

Select Your Display Basis

Pick the "TableTop" you want to work on

Review by Item Status

- ☐ no limit, show All
- ☐ items needing inQuiry/order
- ☐ Waiting for info from supplier
- ☐ waiting for approval from cUstomer
- ☐ On order, awaiting arrival
- ☐ paSt due for arrival
- ☐ in need of Pricing by manager
- ☐ Core return items ...
- ☐ all processes Done

OR

Search on Basis of ...

- ☐ customer Name
- ☐ SD Invoice number
- ☐ part nuMber
- ☐ P.O. numBer

To further filter showings, select first for

clear filters

Applicable Technician

Underlying Machine Make

Applicable Vendor

Underlying High Volume Client

In other words, when you pick a particular display category, you get the specific "tabletop" that pertains to the kind of work you're presently interested in performing.

What happens behind the scenes, when you pick a particular “tabletop” (aka “display category”) is the system goes through each of the pending requests (again, depending on your size operation, at any time there may be several scores of them), and determines if it should properly be deemed to belong within the category (i.e. upon the “tabletop”) you’ve selected. For any item that’s determined should fit that category, it’s added to the “tabletop” display.

So, you pick your display category (either mouse-click on the menu item or keyboard-strike the indicated shortcut), and, instantly, ServiceDesk does the behind-the-scenes categorization, and presents you your tabletop.

The surface of this variable “tabletop” is arranged so that up to 18 requests can display within a single page view. If you’ve picked a category that involves more than 18 requests, the ones beyond 18 simply fall to subsequent pages (use your keyboard’s PgUp and PgDn keys to navigate).

So, here’s the concept. Each request appears along and within what we call an info-band. Each info-band stretches horizontally from the left-edge to right, and holds two lines of text:

The screenshot shows the 'ITEMS NEEDING INQUIRY/ORDER' window (Page 1 of 3). It features a table with columns: Ref #, ItemTp, ItemMk, Model #, Serial #, Rqst, Vendor, Whom, Mthd, Date/Ts, Info Provided, and Order Status. A red circle highlights the item '72796-10-1' in the 'Ref #' column. A blue callout box points to this item with the text: 'right-click on any particular line-item reference to produce its underlying full-request form'. Another blue callout box points to the 'Info Provided' column with the text: 'right-click anywhere in the colorful label area to produce the contextual "CheatSheet"'. The 'Parts Request (5 of 107 items)' form is open, showing details for item '72796' (WASHER MAYTAG WA606). The form includes fields for Invoice, Rqst #, Rqst Dt, Tech, Customer, Item Type, Item Make, Model #, and Sundry #. It also has buttons for ShowJob, ShowActions, Run Purge/Archive, Export, Create New, Save New, and Exit. The 'CheatSheet' is visible on the right side of the form, listing various shortcuts and actions.

Ref #	ItemTp	ItemMk	Model #	Serial #	Rqst	Vendor	Whom	Mthd	Date/Ts	Info Provided	Order Status
65109-1-1	Refer	U-line	ULN-C029FB-00	Z991407-090107	1						
72796-10-1	WASHER	MAYTAG	WA606								
72687-3-1	REFER	GE	ZISB48DSS								
72722-1-1	WASHER	MAYTAG	WA606								
72796-1-1	DISHW	MAYTAG	FE92054978								
72802-1-1	TEST	TICKET									
72796-2-1	WASHER	MAYTAG	993								
72791-1-1	COOKTOP	Viking	SHU6000								
72807-1-1	WASHER	MAYTAG	993								
72790-4-1	RANGE	DACOR	XH5000								
72825-1-1	OVEN	GE	SR58J-2CZ	6X716565929	1						
72826-1-1	REFER	GE	ZISB48DSS	DH035126	1						
72826-2-1	REFER	GE	ZISB48DSS	DH035126	1						
72835-1-1	Cash				1						
72824-1-1	OVEN	G&S	SR58J-2CZ	6X716565929	1						
72758-4-1	REFER	GE	ZISB48DSS	DH035126	1						
72758-5-1	REFER	GE	ZISB48DSS	DH035126	1						

The entire left-third of each band (green text) brings in information directly from the underlying PartsRequest form, which serves as its anchor. This request-specific info is somewhat abbreviated—so as to allow space to fit more process-related work-info to its right. If you’re working on an item and need greater detail about the underlying request, a simple right-click on the item’s reference number (top-left textual item in each band) quickly displays the full/underlying PartsRequest form, with applicable record loaded in it.

Please notice the colorful label area at top of this PartsProcess form. It's intended that the labels there help you identify the information that's intended, within each actual info-band, for the info that goes into each equivalent-position space. So, to know what each operative space is for, just line it up visually with the equivalent position label in that colorful section at top. That's what will tell you.⁹⁰

In regard to the remaining right two-thirds of each info-band, they're to fill-in details that pertain to all continuing processes, as performed in conjunction with *fulfilling* the underlying request. If you check the label areas, you'll get some idea of their flavor.

In such regard, the first element of added information may well be the particular part number that's needed, in conjunction with the request. Or, perhaps not. The fact is, some offices have their techs lookup the needed part numbers before creating the underlying requests. Others leave the lookup to someone in the office. For now, let's suppose yours is in the latter camp.

iii. Managing PartsProcess Items—Specific Operation

So (we'll at least pretend), you're the parts guy. You not only do the ordering; you do the lookup too. You reach the point in your day where it's time to perform the daily ritual. You need to gather up all the requests, as generated by your techs since you last did this task (probably yesterday). There may be other requests, too, such as those from guys at a parts counter. Anyway, instead of gathering paper slips, you instead go to the F8 form. There, you pick the display category "*Items needing inquiry/order*," and see a list of info-bands where only the left-third is filled in. You look at the first, note the type and make of machine, what's wanted, and determine how best to look up the requested part (if you're in appliances, please don't neglect to consider *SmartParts* (Alt-F10) as a good candidate for the easiest and fastest method).

At any rate, using whatever method is easiest, you determine the correct part number. Now, click in the info-band, and type it in the indicated box:

Ref #	ItemTp	ItemMk	Model #	Serial #	Rqst	Vendor	Whom	Mthd	Date/Tm	Confirmed	Expected	Received	InvNbr	\$ Cost	\$ Sell for
Cstmr Nm	Part Description			Ptyl Instr	Part #		Avibity	\$ whlst	\$ retail	PO Nbr	Notes		BinLoc		
65103-1-1	Refer	U-line	ULN-C029FB-00	Z991407-090107	Ship if I/S										
U-lin/Arthur	Compressor			1 Definite	5156168										

type the part number
in the Part # box

Or, we may suppose your tech already identified the number upon creating the underlying request. If so, you'd of course not need to look it up. Instead, you would have seen it automatically pop into the appropriate box for you, just as soon as you clicked within the info-band to display its editing boxes (pulled for you from the appropriate place in the underlying request).

Another possibility would be that you use your vendors to do the lookups (why expend your labor for the purpose when they're willing to do it for free?). The system accommodates that, as well.

⁹⁰ There's something else about that colorful label area, too. In other contexts, we've described contextual "Cheat-Sheets" — excerpts from the Command Summary as applicable to a particular work context. Like Callsheets and the DispatchMap, your PartsProcess interface is laden with otherwise hidden commands and tricks — powerful tools that, early-on at least, you'll need as a handy aid to remind you. Since the colorful label area at top is otherwise un-operative (i.e., it's only a label), it fits the rule that, if you right-click within an otherwise un-operative space (and in a context that offers a Cheat-Sheet), it will produce the Cheat-Sheet as applicable there. You can right-click elsewhere within the PartsProcess form too, so long as the spot where you're clicking has no operative purpose otherwise.

More specifically, it accommodates a plethora of methods for conveying requests to your vendors—whether the requests are for lookup, pricing and availability, or simply to place an order.

The most old-fashioned method of connecting with a vendor, of course, is to simply *call* via telephone, and you can certainly do that here. Suppose, with respect to each item, you call an appropriate vendor. You can ask for the lookup if it was not already done in-house, and upon receiving the part number back, type it into the appropriate space (along with other sensible info into appropriate boxes, such as an indication of availability and quoted price). Or, if you already did the lookup, you can simply ask about price and availability, and type that info into appropriate boxes, as it's provided. Or, maybe you're simply telling your counterpart at the vendor's desk that, in fact, you're placing an order. If so, fill-in the added appropriate boxes to indicate that.

Of course, we all know that, to place inquiries and/or orders via telephone is very inefficient. Instead, you may be going to a vendor's website to perform these functions, and, if so, you can fill-in appropriate boxes to indicate info received (and actions performed)—much the same as if you'd spoken with a human (there's even a box to indicate the particular method used).

Or, you can go for greater automation. Specifically, you can have the system itself generate either a fax or email request for you, one appropriate to each vendor of interest. Here, the general notion (reaching back to sorting slips on a large desk for those requests that will fit best with one vendor, versus those that will fit better for another, etc.), is to eyeball-peruse the current/new requests, and make precisely that kind of evaluation. As you do so, use simple mouse actions to assemble requests as applicable to each vendor.

This process is simple. Begin, for example, by noting that you have one or more items that will best fit for Vendor X. Then tell yourself, "Okay, I'm assembling a request for Vendor X." Now scan down through all the open requests, and for each that should be included in Vendor X's request, do a simple Ctrl/right-click on its info-band. You'll notice, as you do this action, it changes the info-band's background to yellow. That's to designate it's been marked for inclusion in the particular request you're now preparing.

72722-1-1	WASHER	MAYTAG	WA606	12345678	3	Definite
AHS/JONES	test item					
72796-1-1	DISHW	KITCH	FE92054978	KUDS220T0	1	Tentative
HENDE/JONES	test item					
72802-1-1	DISHW	GE	SHU2300	1561615	1	Tentative
TEST TICKET	upper rack					
72796-2-1	DISHW	KITCH	FE92054978	KUDS220T0	1	Definite
HENDE/JONES	upper arm					
72791-1-1	Cooktop	Viking	SHU6000	156156156884	1	Definite
Jones	Door Panel					
72791-2-1	Cooktop	Viking	SHU6000	156156156884	1	Definite
Jones	bottom panel					
72807-1-1	washer	maytag	993	990	1	Tentative
Jones	A-20357/347389					
72790-4-1	RANGE	DACOR	XH5000	156156156	1	Definite
AHS/Fierro	test					
72825-1-1	OVEN	GE	SR56J-2CZ	6X716565929	1	Definite
GE/Contract	Display Kit					
72826-1-1	REFER	GE	ZISB48DSS	DH035126	1	Tentative
Test 2	widget 2					
72826-2-1	REFER	GE	ZISB48DSS	DH035126	1	Tentative
Test 2	widget3					

a Ctrl/right-click marks items for inclusion in a request

So you simply look through the list, designating each item you want to include in Vendor X's request. When done, hit Enter on your keyboard. This invokes a dialog whereby you can choose whether to email the request, fax it, etc.

The basic idea is, by this means you can easily create a request for each vendor, and easily convey it to each (at least each that is amenable to receiving requests in this fashion). ServiceDesk will do the underlying work, to formulate the request to fit the circumstances, according to how you've prior filled-in applicable boxes (for example, if you've not done the lookup to provide a part number, it will ask the vendor to provide that lookup; if the request is tentative, it will ask the vendor for price and availability (P&A-only); if the order is definite, it will be asking the vendor to ship, etc.).

Of course, conveying the request does not complete the process. Each request is formulated in a manner that asks your vendor to respond with appropriate answering information (such as, for example, the part number if the vendor was asked to do the lookup, price and availability in all instances, and whether the part is in fact being shipped, if shipping was requested). The expectation is, once you've made the inquiry/request, the vendor will respond back with these elements of information. And, of course, there's likely to be something of a wait before that happens.

So what do you do during that wait?

Let's go back to considering the paper-and-ink, multiple tabletops notion. When you write on each slip of paper to indicate that at such and such date and time you made a particular kind of inquiry and/or request with a particular vendor, you're not likely to keep those slips on the same work-area/tabletop. No, you'll much more likely move them to another space—a space where you keep slips on which you're waiting for a response back from vendors.

It's the same in ServiceDesk—except ServiceDesk does the moving for you. Once particular boxes for an info-band have been filled-in in a manner that indicates the initial request was made (hint: this is done for you when you go through the above-described process), the band get moved from the *"Items needing inquiry/order"* category of display. Specifically, if the box which indicates order *confirmation* remains blank (while other boxes indicate the inquiry went out), the info-band is moved to the *"Waiting for info from supplier"* category of display.

The idea with this category is simple. In response to your faxed or emailed request, the vendor contacts you back with the requested answers (depending on circumstances, that "contact back" might be by fax, email or even telephone call). Regardless of method, when such info comes back, you need to go to the *"Waiting for info from supplier"* category of display, and fill-in the provided-back data. This fill-in process, properly performed, will take the items out of that category of display (off that tabletop), and move them appropriately to new ones.

As an example of this process, if you'd indicated the order was "Definite," and if the vendor replied that he was shipping, you'd fill-in the box indicating the order was confirmed. You'd likely also fill-in the box indicating an expected arrival date, based on how long it takes to receive shipments from that vendor (please note that each of these boxes have tricks to make filling-in dates super easy; float your mouse pointer over each for tips). Bottom line is: any fill-in that sensibly indicates an info-band belongs on a different tabletop sensibly moves it to that tabletop.

At least it does so *for the next time* a particular tabletop is loaded. You'll notice the immediate response, however—if you've filled-in boxes in a manner that should remove an item from the current display category—is that the info-band turns grey. This is to prevent you from being worried or confused, as you might

otherwise, be if an item that you were working on suddenly disappeared. Don't worry; any item that's turned grey within a particular display category will not appear within that same category next time it's selected.⁹¹

So (and to take stock of where we are in this discussion), we've discussed how to perform appropriate work on your "nothing-has-yet-been-done-on-these-requests-and-we-need-to-do-it" table (aka "*Items needing inquiry/order*"), We've further discussed how, when you did this first work in a manner that produced the expectation of a later response back from your vendor—and when that response came back—you then moved to a new table, to appropriately type-in the responding-back information.

At this point, all your requests are likely to be in one of three states: (a) the part is on order; (b) you have price and availability info to pass on to the customer (for his yea or nea as to actually placing an order); or (c) the initial vendor's response was not acceptable (e.g., price was too high, part was not in stock, NLA, etc.). Please notice, the system provides further display categories (e.g., tabletops) for items (a) and (b), which we'll discuss shortly.

In respect to item (c), if the initial vendor's response was not acceptable, it's apparent we need to inquire with another vendor, and perhaps even make a succession of other inquiries. We do not want to create a new "Request Item," because it's the same and original underlying request that we're still seeking to fulfill. Nor do we want to replace information, in the first info-band that we've typed regarding the first vendor's response. We need to keep that there, so we know there's no need to inquire from him again. Instead, there's a very handy solution. We call them "*daughter*" bands. From any original info-band, you can make up to seven "daughter" bands—to facilitate further inquiries (and or actual orders), with other vendors, but still as tied to the same and original underlying request. We're not going to tell you here how to create daughter bands (though we'll reveal it's a simple, modified mouse-click)—because we want you to practice using the PartsProcess form's contextual Cheat-Sheet. You should use it whenever you need to learn (or remind yourself) of specific commands for specific actions. Go ahead: go to the F8 form's Cheat-Sheet (right-click in

⁹¹ Since ServiceDesk is studying how boxes have been filled-in to determine which display category each process item belongs in, it may be helpful to understand the precise criteria it's using for this determination. To the greatest extent possible, we've designed it to closely mirror what human logic would be, as follows:

1. **'All Items In File'**: Obviously, this category selects every item regardless of its content
2. **'Items Needing Inquiry/Order'**: To show in this category, an item's '*Instrctn*' status must be set to other than "Declined" and its '*Request*' status to other than "Dormant," plus its '*Confirmed*' box must be empty. In addition, it must not fit the criteria for Categories 3 or 4, as below described.
3. **'Waiting for Info From Supplier'**: To show in this category, an item must fit the criteria as described in the first sentence under Category 2. Additionally, its '*InquiryDate*' box must be filled-in with a date, while '*Availability*' box remains empty.
4. **'Awaiting Approval From Customer'**: To show in this category, an item must fit the criteria as described in the first sentence under Category 2. Additionally, its '*Instruction*' box must indicate "Tentative," and the '*InquiryDate*', '*Availability*' and '*\$ Sell for*' boxes must be filled-in.
5. **'On Order, Awaiting Arrival'**: To show in this category, an item's '*Request*' status must be set to other than "Dormant," and its '*Confirmed*' box must be filled-in, while its '*Received*' box remains empty.
6. **'Past-Due for Arrival'**: To show in this category, an item's '*Request*' status must be set to other than "Dormant," and its '*Confirmed*' box must be filled-in while its '*Received*' box remains empty (same as above). In addition, the "Expected" box must be filled in, and the present date must be beyond the expected date.
7. **'In Need of Pricing by Manager'**: To show in this category, an item's '*Request*' status must be set to other than "Dormant," and either: (a) its '*Received*' must be filled-in while its '*\$ Sell for*' box remains empty; or (b) its '*Instruction*' box must indicate "Tentative," while its '*InquiryDate*', '*Availability*' and '*Wholesale*' boxes are filled-in, with its '*Confirmed*' and '*Wholesale*' boxes remaining empty.
8. **'Part Arrived, Process Complete'**: To show in this category, an item's '*Instruction*' status must equal "Declined," or its '*Request*' status must equal "Dormant," or its '*Received*' and '*Sell-For*' boxes must both be filled-in.

Please note that items must be in the last category (i.e., # 7) before they will be ready for movement out of the current PartsProcess file and into its archive.

the colorful label area at top), and look (under “*MANIPULATIONS*” and “*General*”) for the entry that reads “*Request New Info Band*.” That will tell you how to use this method.

So that’s how to deal with the occasional need to make multiple inquiries (or orders) as connected to a single, underlying request.⁹² What about dealing with items where the customer wanted you to first acquire price and availability, then call them back?

As a rule, it likely makes most sense for the parts person himself to call the customer back, for a year or near, immediately upon acquiring the information. It’s easy to bring up the underlying JobRecord (with all appropriate contact info)—by doing the right-click on any info-band’s item reference number (this brings up the underlying request form, where you may then click on its “*ShowJob*” button). But, of course, sometimes the customer will not be available for immediate discussion. We suggest adding a note to the JobRecord’s history indicating the effort was made. Additionally, it makes sense to periodically review the F8 form’s “*Waiting for approval from customer*” tabletop, and, for any items where a year or near has not been received, renew the effort (try, in other words, to keep that “tabletop” cleaned up—just as you do all the others).

When and if you get a year, of course, you can appropriately change applicable boxes to indicate the request is now “Approved,” and place the order with a vendor, much as you would have had the request initially been “Definite.” If you get a near, you can simply change the request status to “Declined” (at such point, ServiceDesk will figure your work on that item is complete, and once again appropriately move it to a different tabletop). If your customer never responds and you finally tire of the effort, there’s another put-this-item-to-bed category called “Dormant” (look for it; you’ll see it).

Finally we are left with the general subject of how to deal, after the fact, with items actually ordered.

In terms of immediate response, this is perhaps the most simple. The parts come in, and you fill-in boxes to indicate the circumstances of their arrival. Essentially, you’re “checking-in” the parts (bear in mind, we’re talking about special-order parts here, and not stocking parts, which are “checked-in” through a totally different process). This is done, obviously, as shipments are received (or any *equivalent* event).

More specifically, as you open a box of just-received special-order parts, you’re going to open your F8 form, and pick the “*On order, awaiting arrival*” category of display. This will show you all items you’re expecting to receive. So, the general idea is, you pull an item from the box, then look within the displayed info-bands (use PgUp and PgDn to move between multiple pages, if applicable) to locate the one that pertains to the item in your hand. Once you’ve located that info-band, fill-in boxes to indicate date received, the vendor’s invoice number, and so on—as applicable to the circumstance.

The above method works just fine if you’re handling a relatively small quantity of special-order parts. If you’re handling more (so that, for example, at any point in time you have *many* pages in the “On order, awaiting arrival” display), perusing through *so many* items (to find the request that matches an item as just pulled from the box), is too laborious. For that situation, you can make the display-selected items more specific to what you’re likely pulling from a particular shipment. Specifically, you can narrow the selection

⁹² If you happen to be in the appliance repair field (as always, we apologize to our clients in other fields for describing something that, at present at least, can only be used in this one), there is an incredible tool that can fly right past the occasional need to deliberately and separately inquire with a succession of vendors. It’s not our tool (it was developed by another company), but we do exclusively link to it. It’s called *MyPartsHelp* (<http://mypartshelp.com>). Believe it or not, once you are subscribed to the MyPartsHelp service (with credentials appropriately setup within ServiceDesk), all it takes is a Ctrl/Right-Click on the part number within any info-band, and inside of about one second P&A info for each of your preferred vendors will display. The pricing info is specific, even, to your own account. The availability info is specific to each of the vendors’ locations. If you don’t see what’s wanted among your preferred vendors, one more click and you’ll get a nearly instant nationwide search. It’s such an incredible tool—you just won’t believe its power until trying it.

criteria by applicable vendor, and even PO Number (see the face of the F8 form's first-display/menu page for instructions).

Upon filling-in boxes to indicate an item has been received, you'll often have your work interdicted with a message. The message will indicate that it appears no more parts are on order for the underlying job, and will ask for your consent for the job's status to be changed into "*Working to Schedule*." This change facilitates other office processes in achieving the re-scheduling purpose (assuming, of course, the job wasn't already scheduled for a return visit in *anticipation* of receiving the part).

The interdicting message may make further offers.

If you have the customer's email address (i.e., within the underlying JobRecord), it will offer send an email to the customer, informing parts have arrived, and requesting a telephone call to book the return visit.

Even better, if you're using SD-CyberOffice, it will offer to make it an email that includes a hyperlink—on which the customer can click, and be taken to an interface on your website to re-book, day or night, and without other human intervention. That's true space-age stuff, and you'd better believe it impresses the customer.

At any rate, once the part is checked in, part-process work on the item (as such) is essentially done.⁹³ The underlying request and its connected process info-band will be moved to the *PartsProcess* archive (contents accessible via **Ctrl-F8**), thereby leaving the *current* work-area uncluttered by the work that's already one.⁹⁴ From here forward, operations in the office (as connected with any job that had one or more special-order parts) resume with job- and schedule-management processes.

—At least, special-order parts-processes are done *for the most part*.

Why the qualifier?

In answer, read the next section

⁹³ Actually, there is a potential further stage. Some service company owners (yours truly being among them) feel that pricing on special-order parts cannot be optimized via any formulaic solution—that, essentially, human judgment is needed on a case-by-case basis. This is because sometimes cost on a part is very low, such that if typical markup was used the ultimate retail would be much lower than anyone would likely guess as normal. Some owners feel they should take advantage of this, and profit by pricing the part more in the region where a layman would guess it should be. Contrariwise, sometimes a part comes in with a cost far higher than any consumer would expect to pay, even at retail, and some owners feel it's best in those situations to apply little if any markup. There is no way to assess these situations except with human judgment, and it's not typical that the same person who does the parts check-in is simultaneously adept at exercising this judgment. If wanted, you can structure the PartsProcess system so, once actual received-cost information is put in on an item (but still without sell-for pricing), it's automatically moved to a tabletop titled "*In need of pricing by manager*." It's a simple task for the "manager" to review that tabletop (at least daily), and type in pricing as applicable. If this is your preference, you'll need to open the PartsProcess form's *CheatSheet*, and click on the selection labeled "*Set whether sell-for price required prior to archive*."

⁹⁴ As an adjunct to checking in these special-order parts, you may want to create labels for them (much as when checking in stock parts, see page 165). In this case, however, since the entire checking-in process is rather less formalized (e.g., in the case of checking in stock parts there's almost a dialog you must go through), there's no concrete inquiry prompting you to do so. Instead, it's up to you. At any time you want (and in regard to any info-bands as displayed in the PartsProcess form, whether newly checked-in or not), you can simply do a **Alt/Rt-Click** on an item (or any set of items). In response, the info-band will turn blue, which indicates it's designated for inclusion in a label print (distinguish this from marking items for fax transmission, where a Ctrl/Rt-Click turns an info-band yellow). After each of the items you want are so marked, just hit Enter—which begins a dialog for printing your labels.

iv. Taking PartsProcess Items “To the Grave”

Until approximately the 2006 to 2008 era of development, there was very little further done with PartsProcess items, beyond what’s been described. Basically, if you checked in a special-order part, ServiceDesk *assumed* it was used on the underlying job. It simply *assumed* so. There was no mechanism to assure it actually happened. There was no mechanism to verify *if* it happened. There were no mechanisms to assure, if the part was *not* used at all, some other appropriate action was taken—such as returning to the vendor for credit, or a deliberate decision to move the item into stock, etc.

To use a metaphor, we were good at giving birth to special-order requests, and at managing their matriculation through to graduation from normal finishing school (assuming normal graduation occurred). But we had no mechanisms for dealing with (or even counting) dropouts.

Documenting Usage of S/O Parts, When Usage Occurs:

The first element of change involved creating a method to check-off, when a special-order part is actually used on a job, that it was used. We added a box in the Type-II PostVisitReport form that displays items prior-ordered for the job, and invites the reporting person to indicate (via a simple checking action) whether each such item was in fact used.

PostVisitReport (reporting on Invoice # 73029)

Date & Time of Visit: Tech: CB, Date: 8/11 THU, Start Time: , End Time:

Describe What the Tech Did:

Machine Info (UIS): Type: REFER, Make: WHIRLPOOL, Model: 1561566, Serial: 4545566

Funds collected:

Were these prior-ordered items used? ☐ 1 259551 (Cold Control)

Is this job done? ☐ Yes ☐ No

Is autho req'd to proceed? ☐ Yes ☐ No

Cancel/Exit, Okay/Save

For the internal-to-ServiceDesk PVR-Typell Report, the system lists any parts as special-order received here, with request that operator check box to indicate if the part was indeed used on the job

When creating SD-Mobile’s PVR interface, we provided a nearly identical function there.

Given this structure, the parts-counter/POS guy has a simple task to perform when either the customer comes in to pickup a POS/special-ordered part, or if he ships it. He needs to do a double-click on the part number that has the double-carets.⁹⁵ This tells the system the item was placed with the customer (i.e., “used”). In response, the system inserts similar text (as described above for special-order items being used on the job) in the underlying PartsProcess item’s BinLoc box. Plus, it changes text within the FinishedForm/POS interface to show usage (i.e., it removes the double-carets).

With the above explanation, we’ve described how special-order items get checked-off as having been used. That’s all well and good, but what happens if a parts does not get used? That is the subject to which we now turn.

Documenting any Other Final Disposition:

The primary question in this segment is: How do you document any non-usage, final disposition of a special/ordered part (such as, for example, returning to the vendor for credit)?

The *simple* answer is, much as each PartsProcess item’s BinLoc box is used to indicate actual usage (when such occurs), you’ll use precisely the same box to indicate any other particular disposition.

More specifically, when you are working directly in the PartsProcess form (regardless of whether in its F8/Current or Ctrl-F8/Archived mode), you’ll use a built-in dropdown from the BinLoc box to pick one of the dispositions offered:

Description of items needed					Inquiry/Order			Info Provided			Order Status				
Ref #	ItemTp	ItemMk	Model #	Serial #	Rqst	Vendor	Whom	Mthd	Date/Tm	Confirmed	Expected	Received	InvNbr	\$ Cost	\$ Sell fo
Cstmr Nm	Part Description			Qty	Instr	Part #	Avblty	\$ whlst	\$ retail	PO Nbr	Notes				BinLoc
73029-4-1	REFER	WHIRLPOC	1561566	4545566	1	Definite	259551		8/14/11 10:35 am	8/14/11		8/14/11		41.10	84.59
Jones	Cold Control														

OF-8/13
RA Rqstd
RtToVndr
CrdtRcvd
MvdToStk
WriteOff

As you can see, there are six options to choose from. The first is the same as is inserted by the system for you when, from any of the applicable contexts, you simply tell it the parts used as per original intent. You could select it here manually, if in fact the part was used, but the insertion had not been made via normally-intended means. The other’s speak somewhat for themselves:

⁹⁵ The system provides many contextual tooltips to remind you of what are underlying circumstances, potential actions, etc. These arise when you float your mousepointer over an applicable location:

Parts Used	
1	CC801
1	259551^^

tooltip when you float your mousepointer over box

To checkoff that item was delivered to customer, Double-Click in this box

In the next image you can see how the double-carets go away after the double-click, and how text fills-in following the description to show the item was passed to the customer (additionally, this image shows a tooltip that comes up when floating over the pricing box, to inform you on what is the basis for the auto fill-in there):

Parts Used	
1	CC801
1	259551

S/O part, cost 41.10, priced on basis of SellFor price indicated in PartsProcess record

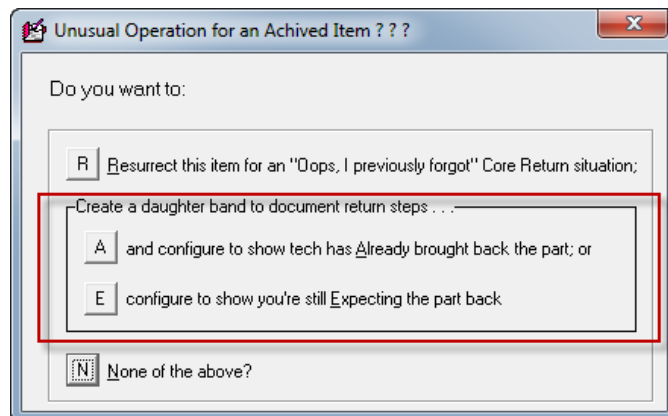
RARqstd (to signify you've requested Return-Authorization from your vendor)
RtToVndr (to signify you've returned it, with expectation of receiving credit)
CrdtRcvd (to signify credit was in fact received)
MvdToStk (to signify a deliberate decision was made to move the item into stocking inventory)
WriteOff (to signify a deliberate decision was made to count the item as a loss)

You might notice that placing an otherwise unused part into these other disposition categories is a hand-on process. There are some things that, simply, humans must do.

You might also notice, if you use the system as above-described in a very simple manner (e.g., you simply select “*RtToVndr*” when you’ve returned the item, and change to “*CrdtRcvd*” when that event occurs, etc.), there is no inherent documentation as to *when* such events occurred. The only indication of such fact is a single, naked status indicator. This may be fine for some operations, but others will want more explicit documentation of what’s involved in the sequence of events. This is where you’ll want to take a further step.

We prior discussed “*daughter-bands*,” as adjunct info/process-bands that may be created to underlie a primary PartsProcess item request. In that discussion, added daughter-bands were described as useful when on the basis of a single request you need to place inquiries (or actual orders) with more than one vendor. For the present context, we’re revealing another use. Specifically, from the Ctrl-F8 *Archived-PartsProcess* window, it’s possible to create a special variety of daughter-band, configured for the particular purpose of managing return of the primary underlying part (i.e., as received within the main band). The idea is that this special variety gives you added places to put in dates, and such, as applicable to particular return effort (and credit received) events.

To create this special “*manage-return*” specie of daughter-band, just **right-click** on the primary item of interest (i.e., the info-band for the item you want to return), from within the Ctrl-F8 window. In response, you’ll get an option box:



As you can see, in regard to creating the wanted new info-band, you have two options: one is obviously applicable if the part has already been retrieved from the tech; the other if it has not. Make the appropriate selection, and you’ll instantly see a new daughter-band appear under the primary item, with some boxes already appropriately filled-in for you:

Archived Parts Request and Process Files (Page 684 of 684)

Mini-Manual				Description of items needed				Inquiry/Order			Info Provided			Order Status					Import Shopping Cart	
Ref #	ItemTp	ItemMk	Model #	Serial #		Rqst	Vendor	Whom	Mthd	Date/Tm	Confirmed	Expected	Received	InvNbr	\$ Cost	\$ Sell to				
Cstmr Nm	Part Description				Qty	Instr	Part #		Avlbty	\$ whlsl	\$ retail	PO Nbr	Notes			BinLoc				
72960-2	WASHE...	WHIRL...	WTW5100SQ0	CU0520799		Will Call	ma			2/24/11 10:31 am	2/23/11	2/23/11	2/23/11	63092885		31.6				
Porter	lid switch				1	Definite	9761959							done direct by tech, Pre-diagnosis w...		OF				
MANAGE																				
RETURN																				
0.00 0.00																				
OF 8/13																				
72489	Begeleann	CU0520799			1	Definite	Shp it 1/S	DAC	Voice	6/23/11 8:05 am	6/23/11	6/24/11	6/23/11	54682	15.54	40.4				
							D007004113													
72908-1	REFER...	LG...	test	939		Dormant	CST	tim	Voice	7/19/11 12:40 pm	7/19/11	7/20/11	7/19/11	9308	12.00	32.9				
Bhw/Spaet	BOX				1	Declined	S-4686DV													

This is the new "manage-return" daughter-band

So now you have spaces where, much as you filled-dates on the parent band to indicate when you'd ordered a part, when you expected its arrival, when it arrived, invoice number on which it arrived and price on the invoice, you can use equivalent-position boxes to indicate then the part was returned, date by which you're expecting credit, date credit was actually received, invoice number and amount of credit, etc. (Just as in any other context, of course, you'll see such editing boxes actually displayed when you click on the item for editing.)

So mechanisms exist to enable meticulous documentation of what happens on every special-ordered part, whether used, returned or otherwise. But such mechanisms are worthless if not used. Indeed, even if there's an *effort* to use them, they remain nearly worthless if not combined with a system that allows you to review and police, to assure all items eventually reach a proper end-disposition. This is our next topic.

Assuring All Corpses Are Buried:

Again, our overriding concept is "cradle-to-grave" management of special-order parts. In the prior two segments, we discussed how items ultimately go into the grave (i.e., either by *use* or some other deliberated end-disposition). The problem is, service offices are extremely busy places, and even well-meaning employees may, if not well-policed, let at least some parts fall-through-the-cracks, never appropriately being used or brought to other appropriate disposition. It's not as though, after all, (like real corpses) they emit a nauseating stench that advertises their need to be buried. Instead, they sit quietly on a shelf (or kicking around in a technician's truck), costing the business serious money.

What you need, therefore, is some mechanism via which you can regularly canvass the field, illuminating such corpses as need to be buried. And, of course, once they are illuminated, each needs to be buried (as per descriptions in the prior segment). That canvassing method is the subject of this segment.

In a nutshell, the Initial-Menu in the PartsProcess form's *Archived* mode (Ctrl-F8) has a section listing particular functions designed for this canvassing:

PARTS ORDERING AND INQUIRY (ARCHIVED FILE)

Mini-Manual		Description of items needed			Inquiry/Order		Info Provided			Order Status				Import Shopping Cart	
Ref #	ItemTp	ItemMk	Model #	Serial #	Rqst	Vendor	Whom	Mthd	Date/Tm	Confirmed	Expected	Received	InvNbr	\$ Cost	\$ Sell fo
Cstmr Nm	Part Description	Qty	Instr	Part #	Avblty	\$ whsl	\$ retail	PO Nbr	Notes	BinLoc					

Use selections here to canvass for items that need to be put in the grave, but have not been

Select Your Function

Show Pages

- begin First
- begin middle
- begin Last

OR

Search on Basis of ...

- customer Name
- SD Invoice Number
- part number
- PO number
- Vendor
- Vendor invoice number
- model number

OR

Manage 'To-the-Grave'

- SHOW items Received but not disposed of
- CREATE DOCUMENT regarding above

filter first by ... clear filters

Technician
-- SHOW FOR ALL --

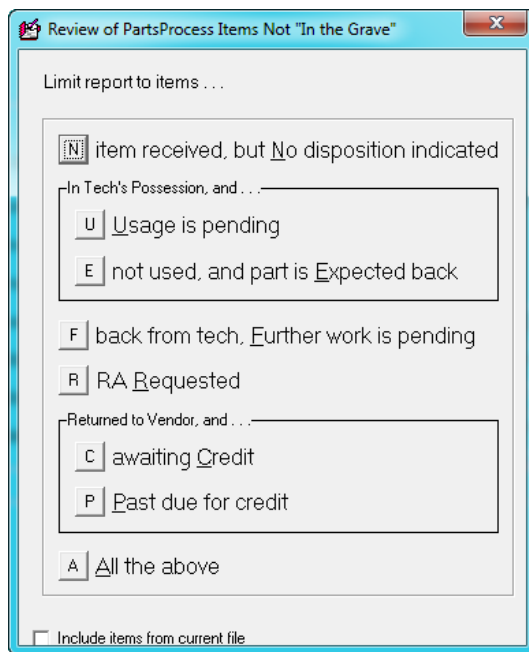
Vendor
-- SHOW FOR ALL --

OR

Run a Report

- Usage analysis
- raw data
- non-usage Analysis
- raw data

The main options, as you can see, are either to examine the items on-screen in their native/actual context, or to create a separate document that contains the information. You're likely to also notice there are filtering options (i.e., so you can limit your canvassing to items that apply to a particular tech and/or to a particular vendor). Regardless of which option you pick, you'll be presented with a set of sub-options, as follows:



As you can see, the options allow you to get rather specific. The main idea is you need to regularly open the review in particular *sensitive* above categories, to assure there are no rotting corpses there.

And, of course, you'll use your native intelligence in doing so.

For example, parts in the "Usage Pending" and "Awaiting Credit" categories are generally of much less concern (and hence merit less frequent and concerned canvassing) than parts in the "Expected Back" and "Past-due for Credit" categories. Regardless, it should be a daily ritual for someone in the operation to canvass at least most of the categories, to assure none has members that are due to be moved to the next station (and, of course, to invoke appropriate underlying work to assure that it happens).

By using these tools, you will reduce your "parts-leakage" cost (referring specifically to special-order parts that fail to reach a proper end-destination) to near zero. It's important. Most service company owners have little idea how much they are losing via such leakage. Most lose a lot. And, it's a double-edged sword. You not only lose via the direct money-outgo from never used parts, you also lose via the burden they then impose by taking up space in your building and trucks (where they also add weight, fuel expense, etc.). By harnessing just these tools alone, you'll make significantly more money.⁹⁶

v. Managing Core Returns

Sometimes you order a part on which there is a "Core Charge" — meaning, besides the direct-quoted price on the part, you're also required to pay a temporary fee that's designed to assure you return the old part that's being replaced. If you don't return that old part (i.e., the "core"), you won't receive any refund on the extra charge. Instead, you're out of pocket for it, and sometimes core charges are very substantial (in the

⁹⁶ I'll venture a "soapbox" point at this juncture. It's that most service company owners are far too prone, when they've special-ordered a part then find they can't use it, to figuring: "Ahh, I'll just put it into stock." I believe that sentiment is almost always a mistake. Parts that are put into stock in such circumstances are almost never used — making the decision to throw into stock tantamount to throwing away money, but worse, because now there's the burden of storing the junk (it's how most service company offices fill up with tons of stuff that's never used). Please don't do it. The reason you were in the position of special-ordering the part, in the first place, is because you'd prior not deemed it worthy of stocking. The fact that you now have the part should not change your judgment.

Consumer Electronics industry in particular, they are often hundreds of dollars). Given this, proper management of core charges and credits can make the difference between business success versus failure – making it imperative to assure that, for each “core” that should be returned for credit, the needed return in fact occurs (plus verify appropriate credit is received, etc.).

ServiceDesk’s system for managing this relies on a feature that will now be discussed for the third time: Daughter Bands. As you may recall, these are *generally* designed to deal with situations where the vendor that you initially check with, on an underlying F8 request, either does not have the part in sufficient quantity, can’t ship soon enough, or if his price seems too high (i.e., you need to shop elsewhere). In such a case, you simply open new info/process bands (daughter bands), and use them to document other inquiries and/or orders as placed with other vendors (yet still pursuant to the same underlying request). In all, you are permitted to use up to eight info bands as connected with a single request (the parent plus seven daughters).

Our strategy for managing cores mainly depends on using — in respect to any part that’s ordered and which involves a core — one such daughter band, but in a special way.

Specifically, when ordering and/or receiving the part (it can be done at any point, really), you should create a daughter band in the normal fashion (right-click within the right two-thirds of a parent that’s not enclosed within editing boxes). Once this daughter band is created, click in the *Rqst* box to expose its dropdown, then select the bottom listing, labeled “CORE.” Upon such selection, ServiceDesk will insert that word to the *Rqst* box, and insert the word “RETURN” in the *Avlbty* box. This serves to setup the daughter in a manner that makes it so: (a) it’s visibly apparent (to you as the user) that its purpose is to manage the core return, and (b) ServiceDesk itself can treat that as its purpose.

Once this *special* variety of daughter band is created, you should use its boxes in a manner that is *analogous* to what you do with a part that’s ordered (i.e., filling-in particular boxes as applicable to various progress events). But here, of course, the meaning of the boxes as filled-in will be changed — to fit the actual and specific circumstances as applicable to getting a core sent back to the vendor. In other words, the meaning of the fill-ins will be different for this context: somewhat similar, but not the same.

On the following page, we provide an illustration with notations to show, specifically, how we believe boxes should be differently used in a Core-Return situation (again, via a “daughter band,” as applicable to a parent via which the replacement part was actually ordered and received).

Please notice that items 1 and 2 (as labeled in the illustration) fill-in for you, on the basis of your selection of “CORE” from the *Rqst* box dropdown.

Item 3 will also auto-fill for you — specifically, when you “check-in” the part to which the Core-Return daughter-band applies.

Item 4 will auto-fill (for ModeA, as there labeled) *if* your tech is using SD-Mobile, and if when queried via Mobile he confirms having retrieved the old part, upon replacing with new.⁹⁷ By contrast, your parts management person should manually change to the ModeB designation upon receiving the underlying core from the tech.

⁹⁷ An important element in the overall scheme involves the fact that SD-Mobile will be programmed to remind the tech, as he begins any repair involving a core, of the fact he’ll be needing to retrieve the older part. It further reminds, as he concludes the job, and requests his confirmation that he has retrieved the old part and is properly in-process to transport it back to the office.

Items 5 through 10 should each be manually filled-in by your parts management person, at appropriate steps in the return process. By such filling-in, you may easily keep track of each element in the process, to assure ultimate and timely completion.

But the above is not all.

The PartsProcess form also possesses a viewing/filter option designed specifically to assist in monitoring Core-Return items in each of several categories.

If you look at the illustration of the F8 form's MainMenu display as shown on Page 124 (near this chapter section's introduction), you should notice that, compared to the actual menu as currently offered, it's out of date. The current menu features an option — not shown there — labeled “Core return items.”

Select Your Display Basis

Review by Item Status

- ☐ no limit, show All
- ☐ items needing inQuiry/order
- ☐ Waiting for info from supplier
- ☐ waiting for approval from cUstomer
- ☐ On order, awaiting arrival
- ☐ paSt due for arrival
- ☐ in need of Pricing by manager
- ☐ Core return items . . .
- ☐ all processes Done

— OR —

Search on Basis of . . .

- ☐ customer Name
- ☐ SD Invoice number
- ☐ part nuMber
- ☐ P.O. numBer

To further filter showings, select first for

Applicable Technician: [-- SHOW FOR ALL --]

Underlying Machine Make: [-- SHOW FOR ALL --]

Applicable Vendor: [-- SHOW FOR ALL --]

Underlying High Volume Client: [-- SHOW FOR ALL --]

clear filters

If you select that option, you'll next be presented with the following dialog box:

Display Core-Return Items

Which sub-category do you wish to view:

- ☐ N underlying replacement Not yet received;
- ☐ E return of core Expected from tech;
- ☐ R core in office, pending Return to vendor;
- ☐ W returned to vendor, Waiting for credit;
- ☐ P Past due for credit;
- ☐ D credit recieved and process Done; or
- ☐ A All the above?

In some respects, these options parallel the operative purposes as existing for reviewing actual parts as not yet placed in-the-grave, reviewed (with a somewhat similar-looking options box displayed) a few pages back). Just as there, there are choices to allow your parts operations person to review particular categories, for the sake of review/policing, and assuring that all items keep moving appropriate toward their proper end destination. The difference: there we were referring to actual *new* parts, as purchased but not used, being

returned, and here were talking about the old replaced-parts/cores going back to a vendor. Regardless, the similarity is there are display categories⁹⁸ to aid the review/policing process, and they should be used on a regular (likely daily) basis.

To summarize, the general strategy for managing cores is as follows:

1. Create an appropriate CORE-RETURN daughter band for any/every part that involves a core charge;
2. Periodically review the above-shown categories, to assure that items within each are being expeditiously processed from one stage to the next; and
3. As items move from one stage to another, document such movements by appropriately filling in applicable boxes.

Security is ultimately found in the fact that, once a Core-Return daughter band is duly attached to a parent PartsProcess request, the entire request bundle will refuse movement to the archive until the Core-Return daughter band is appropriately marked to show proper membership in Category 6 (“*credit received and process Done*”). In fact, it will pester you by virtue of its very, not-yet-processed-as-it-should-be-presence within your current work area, until and unless you certify that the ultimate step (receiving credit after return) has been accomplished.

We believe, by following the above prescriptions, you can easily assure proper processing on 100 percent of your core charges. We further believe it’s absolutely what you *should* do. If your operation involves cores at all, please figure implementation is an essential necessity.

C. Managing Your Stocked-Parts Inventory

There is probably no service company that does not find it prudent to maintain a minimum stock of parts and/or supplies. Nor would you expect to find any that do not need a systematic method for tracking precisely what they have purchased and entered into stock, what they have used out of it, what they presently have left (including where those quantities are located), and how the latter values compare with minimums

⁹⁸ A footnote on Page 146 provides details as to the criteria as used by ServiceDesk to determine which display category any *normal* parts process item belongs in. In a parallel fashion, we here provide a description of the criteria it uses to decide which of the above Core-Return sub-categories an item belongs in:

1. **‘underlying replacement Not yet received’:** The *Received Date/Tm* box on the parent item is empty.
2. **‘Replacement for core Not yet installed’:** The *Inquiry Date/Tm* box is empty.
3. **‘return of core Expected from tech’:** The *Inquiry Date/Tm* box has a date, and the *BinLoc* box does not have text in the “*OF-m/dd*” format.
4. **‘core in office, pending Return to vendor’:** The *BinLoc* box has text in the “*OF-m/dd*” format, while the *Confirmed* box is empty.
5. **‘returned to vendor, Waiting for credit’:** The *Confirmed* box has a date, but the *Received* box does not.
6. **‘credit received and process Done’:** The *Received* box has a date.
7. **‘All the above’:** Any item where text in the *Rqst* box consists of the word “Core”.

Please note that once any item is in Category 6, the parent request with it’s daughters will likely be ready for movement to the Archive — meaning that the quantity of items you’re able to view in this category, at any time, will be limited.

needed before re-stocking. Many companies already have systems in place to meet these needs; some do not. Regardless, ServiceDesk provides *the system* you'll now want to use—designed as elsewhere for maximum effect with minimum effort, and taking advantage of information already garnered from other aspects of operation.

We'll begin our description of this system with a broad, conceptual overview. In such regard, it may first help to know that ServiceDesk maintains four principal files in conjunction with its InventoryControl function:

First is the *MasterPartsPlan*, which as the name implies is a listing of each type of item that you intend to maintain in stock, including its description, as many as three different part numbers, abbreviations for the type of machine (or application) it's used on, the make, your regular and preferred customer prices (assuming you use the latter), and minimum quantities you wish to maintain on each truck and in your stock room.

Closely related to the *MasterPartsPlan* is the *MasterPartsIndex*. Although its function is mostly invisible, you may benefit from knowing that it's here that ServiceDesk maintains an alphanumerically sorted list of all the alternative part numbers you've specified in your *MasterPartsPlan*, along with a truncated item description. This index is used to facilitate cross-referencing and rapid searches within other files.

Next is the *InventoryList*, which features a unique entry for every item of inventory you have in stock at any moment in time. In other words, for each item of inventory (or set of items in bulk, if so managed; see page 147), you'll have an exclusive entry referencing it in the *InventoryList*. This entry includes a referencing number, a notation indicating its physical location (whether in your office stock room or on a particular truck), a date indicating how long it's been in that location, and the amount you paid for it.

Finally, there is the *InventoryJournal*, which keeps a running record of every inventory *movement*, whether its from a supplier into your stock, from your stockroom to a truck, from a truck to use in a repair (and hence a sale), from a truck back into office stock, a return of items to the supplier, etc. This file includes a date, abbreviations indicating from whence and to where the item was transferred, the quantity, and cost of items involved.

Bear in mind that while these files underlie the various processes involved in InventoryControl, you do not have to know their names, nor should you confuse this description of files with the forms you'll use to access their information. It may simply help to know these are the kinds of information the InventoryControl system uses to serve your needs.

In regard to forms themselves, there are primarily two dedicated to this system: the *MasterPartsPlan form* and the *InventoryControl form*.

The first (as its name implies) is used to create and maintain your *MasterPartsPlan* (sometimes called the "MasterList or just List"). Having such a task, it's a form that you'll become rather intimate with when first setting up your system (press **Ctrl-F10** to access it). After that, you'll use it only when needing to revise the MasterList, and so you may go for many months at a time without ever touching it. In such a sense (at least to the extent you find it unnecessary to revise your MasterList), you might say this is a "use and forget" form.

The "InventoryControl form" is also aptly named, for it's function is to access and manage the item-by-item listing of (and record of movements among) actual items in your stocking inventory, to facilitate re-stock to each truck, re-order when inventory is low, and so on (press **F10** to access it).

As you might guess, there's interaction between the kinds of information that's gathered by these two forms. It's based on information created in the *MasterPartsPlan* form, for example, that the *InventoryControl* form deduces how many of a given item should be on each truck, and knows when restock is needed, etc.

A final point to remember concerns the conceptual and functional distinction between what we're describing here (described within this manual as either the "InventoryControl" or "InventoryControl" system) and the *PartsProcess* system described in the last section. The two are very, very separate. In regard to the present topic we're discussing a comprehensive system for managing every item of *stocking-type* part that you intend to maintain in inventory. With the *PartsProcess* system, by contrast, we intend to manage functions associated with non-stock parts (i.e., items you do not intend to maintain in inventory, and so acquisition of such items essentially involves a "special" order). The *PartsProcess* system, you may also note, is not at this point as comprehensive. While it very ably facilitates the ordering process, it does not presently keep track of what happens to such parts (or of how many you have sitting around at a given moment) once they are received. Regardless, please be very mindful of the distinction. Stocking items and non-stocking items are handled *very* differently.

Again, this initial description has been intended as a broad conceptual overview—regarding the *InventoryControl* system. We'll next discuss specifics, but first let's note one added matter.

Of all the systems in *ServiceDesk*, this one requires the most work—by far—in terms of setting up for use. Indeed, it requires quite a lot of *pretty hard* work (not complicated or hard to figure out, just tedious), and can't be used, for the most part, until that work is done (afterwards, thankfully, continuing maintenance is a delightful breeze). We mention this because, for a new user, we think it probably makes most sense to delay use of this system until after you have other, more basic elements of *ServiceDesk* operating. It's true that you won't immediately enjoy its benefits, but presumably you'll be no worse off (in specific regard to *InventoryControl*) than you were previously. If you do so delay, there's one thing you'll need to remember in terms of running the rest of the system without this *InventoryControl* portion yet setup. When doing *PostVisitReports*, one of the queries is whether the technician used any parts from stock. Until you've setup this system, it will be necessary to *lie* in response to that query. Even if a part was used, claim "no." The reason is because the system will have no way of properly responding to a "yes" response until after inventory information is finally setup.⁹⁹

i. Creating a MasterPartsPlan

As your very first task in setting up an *InventoryControl* system, you must describe the parts you intend to maintain in stock (make a list, in other words). That is, of course, where the *MasterPartsPlan form* comes into play. Press **Ctrl-F10** to access the form.

For the most part, we think this form is self-explanatory. It lists 39 items to a page (once you create them at least), allows you to move between pages using the PgDn and PgUp keys, and employs the common

⁹⁹ Actually, as of October 2003, the above advice no longer tells the complete story. At this point in time we added an on-the-fly capability. The thinking was, regardless of whether you have the inventory system *within ServiceDesk* setup, you do have a physical inventory, and you are using parts out of it. And even though *ServiceDesk* won't be able to keep track of what you have in that inventory until after you've setup the system internally, you may have an interim need—particularly if you do warranty work. That is to keep track of what was used from stock for the purpose of automatically inputting such items into your warranty claims. With this thinking in mind, we've now created the ability, within the *PostVisitReport-Type2*, for you to indicate items used out of stock *before* you've otherwise setup your inventory system. There will be no simultaneous reductions of such items from your stated inventory (can't be, because at such point you still have no stated inventory); but the system will record the fact of such usage in the appropriate place where it will then be available for automatic importation into the on-screen NARDA when it comes time to make a claim (or into any other *FinishedForm* for any similar purpose, for that matter).

ServiceDesk convention of allowing you to make edits to any item by first left-clicking on it, following which it is enclosed in editing boxes. To delete an item, right-click on it (also a common ServiceDesk convention). To add new items, either click on the first empty space, or select the form's Add-Item command.

Of course, you should know something about the types of information ServiceDesk expects you to list in respect to each item you'll be describing. Go ahead and bring up your MasterPartsPlan form, then hit Alt-A to add your first item to the list. Immediately, you'll see a row of editing boxes into which you're expected to enter information.

The first three boxes, you'll see, are all for PartNumbers. Why three different PartNumbers? In our experience, we've found that a given kind of stock item can often go by several different PartNumbers, and it's helpful to have more than one, of these alternatives, at your ready grasp. Thus, we've allowed up to three. They are permissive, which is just another way of saying there's no requirement for you to use all three. The possibilities are simply there if you want to use them. The only requirement is that you list a unique number (or other identifying equivalent) in *at least one* of the three spaces.

Organizationally, we do have some specific intent about what kind of numbers you might most beneficially use in each box (though this is not required; you can use each in whatever manner you please so long as that minimum requirement is met). Our thinking is that, ideally, you may want to use the first box for an OEM (i.e., original equipment manufacturer) part number, if applicable. If yours is an appliance repair company, for example, and you're entering a part used in Whirlpool dryers, we suggest entering Whirlpool's part number here. The second space, more typically, we think should be used for an Industry PartNumber if applicable (in many cases there's a number that's recognized within an entire industry as referring to some product generically, for example, such as "R22" for that kind of refrigerant). The third space, since many service companies buy a large percentage of their stock from Johnstone Supply, we've intended for use with a Johnstone part number (it's six spaces long, while the others are 16 and 10, respectively). Of course, you may again use it for whatever kind of identifying information you wish, or for none at all if that's your preference.

Following these first three PartNumber spaces (remember you must have a unique identifying number in at least one), are two very small spaces, allowing only two characters each. The first of these is for an abbreviation indicating the brand of machine or fixture the item is used with (if it's use is not limited to a particular brand, we suggest using the letters "AL" to denote applicability to all makes). The second is for an abbreviation indicating the kind of fixture or appliance the item is used with (i.e., you might use "DR" in reference to a dryer part).

The very next space, far longer than the others, is for your *description* of the item. Obviously, descriptions are exactly that, and can take whatever form you prefer. However, we have a specific suggestion in their regard. For the first word in the description, consider using a term that's as general as possible, then a comma, then more specific details. The reason is because ServiceDesk will sort your list alphabetically, by description. If you were to list one item as "1/3 HP CONDENSER FAN MOTOR," for example, another as "BLOWER MOTOR," and still another as "FASCO BRAND UNIVERSAL MOTOR," the items would all end up in rather different places on your final list, making it much less organized and clear. Instead, it would be smarter to list the above examples as "MOTOR, CONDENSER FAN, 1/3 HP," "MOTOR, BLOWER," and "MOTOR, UNIVERSAL, FASCO BRAND." In this way, all three listings will end up being together after ServiceDesk alpha-sorts your list, and it becomes much easier for you to review each motor you've included in the list, or to locate any just by looking alphabetically under "MOTOR."

Following the Description block, there are two spaces for you to insert the price, or prices, you intend to sell the item for. The reason for two price spaces is because you might have one price for regular

customers and another, discounted price for home-warranty or other special clients (if not wanting to offer any second category of special prices, just leave the second field blank).¹⁰⁰

Finally, there are two remaining, and very small spaces at the end. The first is for the quantity of the item you intend to keep in stock on each standard truck,¹⁰¹ and the second for the quantity that will be considered the minimum for your stock room before ServiceDesk prompts you to reorder.

In regard to these last two quantity spaces, there's an important option which allows you to insert something other than a quantity. There are many kinds of stock, obviously, where a little bit of some bulk supply will be used at a time, but typically not the entirety (like one clamp out of a box of clamps, for example, or a little freon out of a jug, etc). For such items, it makes a lot more sense to simply document having placed that supply into stock, and then to assume some supply remains until there's a report that it's been exhausted—rather than attempting to constantly quantify how much of the box or container remains. If this is the kind of item you're listing, don't put a quantity in either of its two quantity boxes. Instead, type-in the letters "SU" (abbreviation for "supply"), and ServiceDesk will subsequently interpret the item accordingly (when any item is so listed, the quantity value that's kept in inventory will refer to the number of such supplies (i.e., boxes, jugs, etc.), rather than to the amount of material left within any specific box or jug.

It does involve some amount of work, obviously, to create this list of the various items you intend to maintain in stock—not a huge amount, but considerable nonetheless. Within our own service business, we'd developed an effective list that worked very well for us (we were getting approximately 75 percent first time completions with about \$3K in inventory on each truck). If yours is likewise an *appliance* service business, we invite you to use our list (and reduce or possibly eliminate the work of setting up your own). A copy is provided on the installation CD (using Windows Explorer, look for 'StckList' in the 'OtherFls' folder and copy into the 'c:\sd\netdata' folder on whichever computer you are using as FileServer). At the very least, you might find it easier to start with our list as a foundation, and add or delete entries to taste rather than starting from scratch. And, even if you're in another trade (in which case our list would probably not suit your needs at all), you can, if wanted, load our list and look at it as an example (just delete the same file when wanting to replace it with your own list). Also and incidentally, if any of you in other trades are particularly proud of whatever list you ultimately develop—and are willing to provide us with a copy—we'd be happy to make it available to other ServiceDesk users on future CDs (along with the one oriented toward appliance service that's currently there).

In regard to other features on the MasterPartsPlan form, you'll notice there's a command button for doing a Sort/Index routine (sorts items alphabetically and makes the MasterPartsIndex that ServiceDesk needs in other InventoryControl contexts; be sure to run it before otherwise using the form's data), and also a button for printing a copy of your list (it's useful to take a printout of item-types when counting your beginning inventory). Naturally, there's also a 'Search' button, which allows you to quickly find listings based on *any* of the alternative part numbers, or even description. Finally, there's a command button that's less obvious in purpose, labeled "Make Ascii File." Its function is to create a version of your list that can be loaded into your word processor, and there edited or formatted for whatever separate purpose you might like to make of it. Most companies will equip each of their technicians with a catalog-type listing of stocked parts, for example (based on the MasterPartsPlan). With the special formatting that can be done in a word processor, you can setup to print your list in multiple columns on a single page, or on both sides of a single page, or whatever else

¹⁰⁰ An enterprising *Mike* from Alpha Omega Service wanted to list flat-rates, but noted that ServiceDesk presently has no built-in provision for this. He reasoned that the second price field, that we *intend* for stating a discount price on the part (if any), might instead be used for stating his flat-rate to install the part (he further reasoned that for jobs that do not involve a part, he could create a fictional part listing). Good thinking, Mike. Why not? If this suits your purposes better, terrific. And thanks for sharing the idea.

¹⁰¹ If you want to specify different quantities for different trucks, refer to the discussion on advanced features, page 169.

is the handiest format for your technicians to use and keep in their clipboards (as needed for them to correctly price parts, know what they're supposed to have on their trucks, etc.).

Again, bear in mind that the MasterPartsPlan form has one primary function: it's the tool you use to create and maintain your own MasterPartsPlan (additionally it creates an index that ServiceDesk uses in related stock functions, and you can use it to create a version of your catalog-type listing in Ascii format, but these are peripheral to it's main function).

ii. Informing ServiceDesk of Your Already-Existing Stock

After you've created your MasterPartsPlan, you'll be ready to begin informing ServiceDesk of what items you already possess, at the point where you begin using this system. Unless you're a spanking-new company, you're obviously going to have lots of inventory that's accumulated during previous years of operation. Whatever it takes, you're going to need to come up with a listing of every item (or every separate container of supply, see page 147) in existing *intended stock*.¹⁰² If the system is going to keep track of what you've got, after all, it's got to know what you're starting with. You may already have such a list, or may need to physically go through everything—both in your storeroom and on all the trucks—and count it.

The easiest way to perform such a counting is to have a sheet of paper that lists each of the items that need counting at each location (we'll call it an Inventory-Counting-Sheet). Then go to each such location (with its own respective Inventory-Counting-Sheet in hand) start with the first listed item, count the quantity found, then write it in. Proceed to the next item in the list, count the quantity found, write it in, and so on.

The best way to make these Inventory-Counting-Sheet is from the **F10** form. From there, select the option to '*Review Existing Inventory*' then '*List of Pt#s, showing total Qty each*'. Now select the location you want to make a list for, then click on the '*Print List/Labels*' button. At this point select the option for '*an ordinary List*' and select your printer. You'll find the system makes a very nice printout that's perfectly suited for going to the location to perform the counting and write-in of found quantities (it even includes a space next to each listing to write-in in the quantity found). The best idea is to print such a sheet (it likely will consist, in actuality, of many pages) for each location that needs counting (i.e., one for your office storeroom and one for each truck).

With these sheets prepared, you're ready to do the actual counting. Just go to each location with its appropriate Inventory-Counting-Sheet, and do it. We know it may be a lot of work, but it's *got* to be done (and just think, if you haven't previously had an accurate and comprehensive accounting of intended stocking-type inventory, finally you'll have it).¹⁰³

¹⁰²We emphasize the phrase "intended stock" to make a point, which is there's no need to tally *everything* you possess. If you're like most companies, you tend to acquire parts that, if you could easily make it so, you'd much rather have back at the distributor and the money in your pocket instead. The idea here is to inventory just the items you intentionally stock (i.e., the same ones you've listed descriptions for in your MasterPartsPlan). Don't worry about all that other "junk." At this point you're best off not even considering it.

¹⁰³When we first setup the system in our office, we scheduled an "inventory party" one Saturday morning (like most, we did not previously have any actual count of what existed on each truck). We assembled all the trucks in a parking lot side-by-side, and basically emptied them all out. We then made separate piles throughout the parking lot of each kind of stocking item (it took a lot of space). After everything was sorted, we then gave each technician the Inventory-Counting-Sheet that applied to his truck. His task was then to go to each of the piles, pull the quantity of item for his truck as specified on the list (i.e., the list says how many of each item each truck should be stocked with), and mark on the list the quantity he actually pulled and put back on his truck (which, if there were not sufficient items in a given pile to fully stock each truck, would of course be less than the specified amount). We found, incidentally, that many trucks previously had far more of a given item than they should have had, and several had less. Plus, every truck had numerous items of non-stock parts that they should not have had. On balance, we found that in regard to some items there was a significant surplus (above and beyond what was needed on the trucks) to return to our storeroom stock. In regard to others, there was not enough for all the trucks to meet their minimums. At any rate, after the event was concluded we found ourselves possessed with what we needed: printouts showing (with actual quantity written-in by hand) what was actually on each truck at that point in time (not to mention that

Once you've filled-in each of these sheets with actual counted quantities, it's time to input the information to ServiceDesk. Remember here that when you setup item *descriptions* in the MasterPartsPlan, you were not in any way indicating items actually possessed; you were only describing a *plan*, what you *intended* to possess. Now it's time to do the real thing: check in actual items, tell ServiceDesk you have 2 of these, 3 of those, and so on.

There are couple of different ways this can be done. The *normal* way is that items are "checked-in" when received in a shipment (or otherwise purchased from a supplier). You'll see the F10 form has a nice facility for this, which we'll discuss shortly. But there's also another *potential* way. It stems from the fact that, in any inventory system, you'll sometimes find yourself with a quantity on hand that's different from what the system had reckoned. To deal with that, we also have a facility in the F10 form to "adjust indicated quantities." In established operation, that latter function's purpose is solely to correct for differences between what's actually found and what the system formerly reckoned. In the initial setup situation, however, the facility can be used to do a kind of pretended "checking-in" of actual quantities.

As it happens, the best overall strategy for inputting your initial quantities of stocked items into ServiceDesk is a combination of these two methods. In regard to the office storeroom, you should *pretend* you're receiving a shipment into stock (of all the items you already have there), and use the method (within the F10 form) as provided for that ostensive purpose. Then, to input the quantities found on each truck, you should use the 'Adjust indicated quantities' method.

Here follows a brief overview of each such step.

For your office storeroom inventory, put its completed Inventory-Counting-Sheet in front of you on your desk. In ServiceDesk, bring up the InventoryControl form by pressing **F10**. As the form displays, you'll see that it offers several options, including one labeled "*Receive items into stock*." Select this option and you'll instantly be presented with a query as to which supplier sent the package of items you're checking in. Here, of course, you're dealing with a "package" consisting of everything you already have in stock in that storeroom, but the system insists you name a supplier, so make one up. In particular, it's a good idea to make the name helpful for the circumstances by calling "*Initial Stock*" (or something similar). So, in the space provided, type in "Initial Stock". Now ServiceDesk will note that you have no supplier in your list by such name, and will ask if this is a name you want to add. Indicate yes, and ServiceDesk will then add a category of supplier (supposedly) by that name.

Now you can begin checking-in actual storeroom quantities. As you'll note, the system works in something of a dialog fashion, asking you first to enter the part number you're checking in, then the quantity, then the price paid for it. In regard to price paid, it may be difficult to come up with exact historical costs, and unless you're personally a stickler for such matters, we don't recommend sweating over in fact. In fact, for our purposes we thought it was best to use the W.A.G. principle. This is the method that's used by sophisticated engineers when they need to calculate something less critical than the chord width for the beam supporting an airplane wing; it stands for Wild Ass Guess. The idea is to get the information as quickly and easily as possible. In probably won't matter much, in the long-haul, if you use the W.A.G. principle exclusively in this initial valuation of parts.

You'll notice there's a drop-down list to assist you in selecting the item being entered (you can type the description or any of the three alternative part numbers and matches should appear in this list). To select an

accuracy-of-intended-distribution between and among the trucks was much improved). It was left simply for office personnel to do a similar accounting of stock in our storeroom (writing in the found quantities on another Inventory-Counting-Sheet), then of course to input the information to ServiceDesk.

item from the list, you can click on it, cursor down to it then hit Enter, or keep typing until it's the top item in the list, then hit Enter—whatever is easiest. Enter as many items as you wish in a session (but do not exceed 99, for that's the most the system can absorb in a single session-sequence), then hit Enter with nothing in the box to conclude a particular entering session (as instructed, incidentally, by an instructional note in the form). You'll then be asked to confirm if you're concluding the present set of entries, and if so will be queried as to the shipping cost of the package (not easily applicable in the case where it's initial stock, so report "0").

When entering these initial storeroom quantities, there will likely be many, many entries to make. It will probably be best to do 10 or 20 entries at a time, then conclude and begin a new session, repeating until all is eventually entered.

Once this portion of the process is complete, ServiceDesk will now “know” about all the items of intended stocking inventory in your storeroom. Now it needs to know what's on your trucks.

Again, the method we'll use for “checking-in” initial truck inventory is the “Adjust Indicated Quantities” method.

For reasons we'll not bother to explain here, make sure you do the storeroom first, as described above. Then, for each truck, take its completed Inventory-Counting-Sheet, put it on the desk in front of you, and within ServiceDesk bring up the InventoryControl form by pressing F10. Now select the option to ‘*Adjust indicated quantities*’ then ‘*by reviewing entire List*’. Select the applicable truck, as prompted, then you'll see you're presented with an on-screen list, showing each item description in the same sequence as the Inventory-Counting-Sheet that's on the desk in front of you. All you have to do is look at the first written-in quantity (as physically counted on the truck), type it on your numeric keypad, hit enter, then the next, and so on. In such fashion, you can rip through literally hundreds of items very quickly. As the instruction prompts, you can simply hit your keyboard's Esc key when ready to record, then follow any added prompts. It's that easy. (If wanted, you can break this up into multiple sessions as well.)

With all of this having been accomplished, you'll have everything in place for what will—from now on—be an almost effortless basis for maintaining near perfect and constant control of your stocking inventory. Yahoo!

iii. Maintaining Your Inventory

When the above preparations are all complete (and assuming you do the counting/input portions over the course of, say, one very long and busy weekend),¹⁰⁴ you will possess what is essentially a snapshot image of your inventory situation at a given *point* in time. However, we know inventory is not a static thing. It's changing all the time as parts get used and new ones are shipped in. To keep the picture accurate, the system obviously must be informed of all movements, in or out.

In such regard, let us first consider movements of items into your inventory (now referring to items actually being added physically, as opposed to merely registering items already there, per the last section).

¹⁰⁴We recommend you attempt to complete all counting and input of initial stock *between* active business days. The reason is because you want to begin the system with an accurate at-a-given-moment accounting of your inventory situation. This becomes difficult if, say, you've counted four widgets on John Smith's truck on Saturday, then he uses one on Monday. On Tuesday he goes to report its use, but can't because you've not yet “told” the system he has *any* on his truck. Then along comes Wednesday and, finally, you go to input the (now supposed) fact that he has four on his truck. Complications like this *can* be dealt with, but it's much simpler if you simply get it all done before subsequent real-world events begin to change from the initial snapshot that's registered during your one-time inventory party.

Indeed, and even more specifically up front, let us consider that, before any items are moved into your inventory, you probably need first to *order* them.

Ordering restock of depleted inventory is obviously no trivial matter (and again, should be contrasted with the process of ordering non-stock parts, as discussed beginning at page 122). However, ServiceDesk makes it easy—and it can all be done electronically, if wanted. All you must do is select the InventoryControl form's '*Order stock replenishment*' option. The system will soon (after a bit of searching) display a list showing every item on which it finds fewer items in stock than the MasterPartsPlan-specified minimums. The list identifies each item and shows how many are in stock compared to specified minimums. It further deduces and displays the difference (i.e., how many are needed to restore minimums), and indicates the quantity that should (assuming no pilferage, etc.) be on your storeroom shelf. It invites you, quite simply, to specify how many of each such item you want to include in a particular order. All you must do, in regard to each such item, is hit a number (i.e., quantity) on your keypad, then Enter.

In this manner, you'll quickly formulate a restock order (it might take a few seconds). If your company is like most, there are probably some items that you prefer to purchase from one supplier, and some from another. That's easy. Just formulate one order for the first, specifying quantities for the items you want to order from it. Formulate a different order for the second, and so on (and as often as you prefer to place restock orders, whether weekly, monthly or "whenever you get around to it").

After formulating any restock order, the remaining task is to transmit it to the supplier. We do this, essentially, by "printing" it—with the verb there having something of a broad meaning. On the one hand, you can literally print a sheet having all the order information on it (simply follow the prompts to do so). You could take this sheet and mail it, telephone the supplier and read off of it, or much more probably put it in your fax machine and send its image to the supplier. Alternatively, you can specify your computer's internal fax as the "printer" and fax the "order sheet" that way, so that you never even need touch paper. In our office we prefer to "print then fax," because we automatically then possess a hard-copy version of what the supplier (presumably) received.

At any rate, as operations continue, you'll undoubtedly soon be placing orders for restock, and receiving shipments in response. With each shipment received (or even if someone drives to a supplier's warehouse and picks stuff up), it's imperative that you inform ServiceDesk of each and every item entered into inventory. This is essentially the same process as described for entering initial inventory, except that now we'll be checking in a *genuine* shipment (rather than pretending to be checking in a shipment as a *means of* checking in our initial stock). And here, of course, we'll be using a real supplier's name, real, on-the-invoice costs, and so on.

One of the features you'll notice when checking in parts is that ServiceDesk asks if you'd like to print labels for the items received. These labels can be rather useful, carrying much information that may not otherwise be on the parts. If you'd like such labels, simply indicate 'Yes' when queried. The system is configured to print individual labels for each part onto Avery # 4013 labels (sometimes listed as # 04013). These are sized 3.5" X 15/16" (spaced one label per vertical inch), and arranged in a single column on tractor feed paper for maximum printing convenience. A printed label will look something like the following:¹⁰⁵

¹⁰⁵When you consent to print these labels, as with any other kind of printing in ServiceDesk, you'll be presented with the PrinterSelection form. In this case, however, you'll see that the form offers an option that is not normally shown there. There's a little item that says 'Print to File'. It's purpose is to allow the label data to go into a file rather than to your printer. The reason is because we've had some clients with special needs in terms of label printing (such as bar-coding, for example), and we've not been ready to accommodate these needs from directly within ServiceDesk. Instead, therefore, we here provide the option to print the data to a file. They can then load this file into a separate application, and handle their special and unique needs from there. Another matter you'll notice if you try this option is that the system then queries you in regard to which invoice number the shipment came in on. The reason

WD15X93	563	K35175
FILL VALVE		GE DW
Rec'd 10/11/98 from JS .16.154.		
1 per trck, 3 min strg, \$24.84 ()		

Here you see the three alternative PartNumbers arranged along the top line. In the second line, you see the item Description followed by your own MasterPartsPlan abbreviations for ApplicableMake and MachineType. In the third line, ServiceDesk prints an abbreviated note regarding when the item was received and from whom, followed by a moderately disguised¹⁰⁶ code which indicates the price you paid for it. Finally, in the last line is an indication of how many of this item each truck should maintain in stock (according to info you've input to the MasterPartsPlan), the minimum you should have on hand in storage before re-ordering, your standard retail price, and (in parentheses) your preferred discount price, if any (no dollar sign is included here).¹⁰⁷

Besides new items going into your inventory, there's obviously old items going out (i.e., being used and sold). These movements must also be registered. However, and contrary to what you might expect, this is not a task that's *normally* performed in the InventoryControl form (although, should the odd need arise, it *can* be done from there). Instead, it's done much more conveniently in the PostVisitReport form, as part of the PostVisitReporting process (see page 110). As part of the dialog in that process, the system asks if any stocking parts were used on the job. If the query is answered in the affirmative, it collects information about which parts were used, and from which location. It then makes appropriate entries and adjustments in your inventory files (not to mention a narrative-type report within the job's History).

Of course, as parts are used from each of the trucks, they need to be restocked. Again, we're back to using the InventoryControl form, and again the process is basically the same as when we "transferred" initial inventory to the trucks, except in this case we'll be using the system to inform us about what actually needs restocked, and reporting on what (at the moment) we are actually transferring, physically. Again, there are alternative methods for registering transfers ('Review needs and disburse to truck' versus 'Disburse without prior review'), and you may use either depending on preference.

In our office, the habit is that each tech arrives at staggered times in the morning to report on jobs from the day before. Upon concluding (meaning that, among other things, he's made entries regarding all items used from stock), and before taking a stack of new jobs and heading out for the day, he asks for restock. An office person then strikes an appropriate sequence of keys in ServiceDesk ('F10' to bring up the InventoryControl form, 'T' to select the 'Transfer to/from trucks option, 'N' to select the 'review Needs and disburse to truck' option, and finally the tech's initials), and quickly sees list showing all items that need restocked. If it's one or two items, we may simply make a mental note, go to the storeroom and pull the items. If more, we hit Alt-P to print the list, then go to the storeroom with paper in hand. At any rate, we then return to the screen and report on how many of each item were transferred to the tech. Typically about a one-minute operation, this keeps all trucks constantly replenished with the stock they are supposed to possess, and everything is recorded for future reference if and when needed!

is because the same clients who've had these special printing needs also needed to keep track of which invoice number each stocking part was purchased on. This allows them to do that too. If you have similar needs, obviously, the feature is there for you.

¹⁰⁶ To decipher the code, just drop the leading decimal and first digit, along with the trailing decimal and last digit. What's left in the middle is the price you paid for the part, including shipping (\$6.15 in the above case).

¹⁰⁷ You may also print labels for items already *existing* in stock, if wanted. First, select the 'Review Existing Inventory' option from within the InventoryControl form, then its 'List of P#s, showing total Qty each' sub option. Indicate the category you wish to list, then the system will find and display all findings for that category. At the same, it will display a command button labeled 'Print list/labels'. Click on the command button (or press Alt-P). You're then presented with options, one of which is to create labels for all items in the selected category.

As you can see, the system is very easy (after initial setup, at least), and in terms of what it accomplishes, most powerful. Indeed, there are many other features on the Inventory form that we've not bothered here to describe, for they are largely self-explanatory (simply review and explore the options provided, and you should quickly gain an understanding). There are just a few items that bear further explanation.

First, occasionally it may happen that one of your techs report using an item, then they realize it was entered incorrectly, and it's really something else they used. To try to keep things straight, they inform you. That's where the bottom two options on the InventoryControl form come into play. First you'll need to cancel the incorrectly indicated usage. There's an option specifically for that purpose, and so labeled. Then you'll need to enter the correct item, for which there's an obvious counterpart option.

Second, unless you're very lucky, you'll probably lose one of your techs, from time to time, and have to replace him with another. You may notice that the system identifies each truck according to the initials of the tech it's assigned to. Correspondingly, all of the parts checked out to a truck are identified by that tech's initials. This means, if you remove one tech's name from your roster and replace it with another, you're going to still have a bunch of parts assigned under the old tech's initials—even though, presumably, the new tech probably has taken over the old tech's truck and all its associated inventory. What we need, therefore, is an easy means of changing the assignment references, for each of the parts formerly assigned to the old tech, to the new tech's initials. There's a utility of this specific purpose (review the options) on the InventoryControl form.

Finally, it's inevitable in any inventory system that inaccuracies will eventually creep in. This may happen because someone uses a part and forgets to report on it (or perhaps returns an item and forgets to report it). There may be pilferage, inaccurate reporting on parts received, or any of several other possibilities. Regardless, when you find a discrepancy between what the computer claims and what you actually find on the shelf, it needs to be corrected (the system needs to be informed, in other words, of what's really there). ServiceDesk provides two alternatives for this information.

In the most typical case, you'll find yourself wanting to enter corrective information on a piece-meal basis (say you've noticed that one or two items are indicating an erroneous quantity when performing restock to a truck, for example, and you want simply to correct these). Here, the best method (after choosing the '*Adjust indicated quantities*' option) is to further select the '*by Selected item*' sub-option. At this point, a drop-down list appears, from which you can select the item whose quantity you wish to correct. Upon selection, you'll be prompted to indicate the correct quantity, and the system will indicate how many additions or deletions are necessary to make the indicated quantity match. If it's additions that are needed, ServiceDesk will search to find the most recent cost of such items, and propose the additions be valued similarly (if it's unable to find any exemplars of cost, it asks for input from you).¹⁰⁸

If, on the other hand, you're wanting to do a complete and thorough recount of all items (whether in regard to office stock or what's on any particular truck), the alternate method will be easier. In this case, you should begin by printing out a listing which shows the quantities of each item, as presently indicated. To do this, select the InventoryControl form's '*review existing Inventory*' option, then its '*List of Prt#s, shwng qty of each*' sub-option. Request a viewing of items in the pertinent location, then request a physical printing by clicking on the 'Print list' command button. This will give you a long printout which you can then take to the

¹⁰⁸ Note that these corrections are meant to deal entirely with quantity indicated in office stock (abbreviated 'OF'). If a truck actually has more of an item than indicated, you need to transfer such items (within ServiceDesk) out of office stock and to the truck, as necessary for the indicated truck stock to show accurately, and vice versa. If such actions then render the indicated office quantity inaccurate, then you'll use the features, described in the main text, to adjust the latter.

relevant location, using it as the prompt, counting each item off the shelves in the sequence listed, and writing in the correct quantity if it differs from what's indicated. ¹⁰⁹

After finishing this exhaustive counting, you must next return to the InventoryControl form, select its *'Adjust indicated quantities'* option, then the *'by reviewing entire List'* sub-option. Now you'll see the entire listing on-screen, and have the opportunity to easily indicate correct quantities (should they vary from indicated) next to each. At conclusion, you'll be asked to confirm the intent to save alterations to the file.

One thing you should be aware of in regard to these procedures is that, besides adding items to or deleting from the InventoryList to make quantities accurate, ServiceDesk also makes entries (as with all other transfers) in the InventoryJournal ("IJrnl") to document the process of transfer. If you review this journal from time to time (select the InventoryControl form's *'Review Purchases and Usage'* option), you'll see that, using two-letter initials, it indicates transfers from a supplier to the office, from the office to a particular truck, from a truck to a sale (indicated by the '\$\$' symbol), and so on. In the case of correcting quantities indicated in office stock, it will show items as being transferred either from or to 'SC', indicating 'Stock Corrections'. Thus, if you are in the future interested in knowing how many adjustments have been needed to maintain an accurate indication in inventory, you can simply look at the IJrnl for all items going either to or from 'SC'. If there are too many 'SC' references to stock *removals* (i.e., more than could reasonably be expected to result from mere inaccurate reporting), obviously, you may deduce that you have a problem with pilferage.

There are, as mentioned, several other features which are quite self-explanatory upon examination (just explore the possibilities, you'll see what they are). On the other hand, there are a few reporting-type features that we've not yet developed, but of course intend to in the future. There's no inventory-aging feature yet, for example, and no provision for tracking frequency of use among items, or for tabulating values of input and output over a given space of time. Please let us know if you feel an urgent need for such capabilities, and we'll give them higher priority in our development schedule.

iv. Advanced Features, Including Specialized Stocking-Plans for Disparate Inventory Locations

The foregoing has described the basic InventoryControl system as originally developed. For most operations, those basic-level features should prove more than adequate, and to the extent they do there will be no need for you to read further in this section. If, however, you want to list more information in conjunction with each part item than is allowed within the main section of the MasterPartsPlan form (such as additional part numbers, preferred vendor, expected purchase cost, bin locations, etc.), you'll want to consider the advanced capabilities that we'll here discuss.

In particular, you may notice that the basic system is setup with the assumption that you'll be stocking all your service trucks with the same identical inventory set. The reason is because this keeps things simple, and is what the majority of service companies do. If, however, you've setup your operation so that different trucks are specialized for distinct purposes (such as, for example, having one or more trucks setup for refrigeration work, another setup for servicing laundry equipment, etc.), this section is especially for you.

All of these advanced features are accessed from via the MasterPartsPlan form, and from two contexts therein.

¹⁰⁹As an alternative, and if you've elected to bar-code your stock, you can use a bar-code scanner to conduct the inventory. It will create a file which you can then have ServiceDesk import. This file will contain all the needed comparison data.

First, if you want to setup different stocking plans for different trucks (or other alternative locations), you'll manage initial setup via the *InventoryLocations* form, which is accessed by first clicking on the MasterPartsPlan form's "Other Housekeeping" button, then by picking the "Manage Locations/TruckTypes" option. This exposes a two-section interface. In one section (to this form's left), you are allowed to create up to six different kinds of specialized stocking plans (these are in addition to what may be considered the "standard truck" and "standard office" plans, which exists by default and get their planned minimums from the "Trk" and "Offc" columns within the face of the main list). In the second section, you are allowed to explicitly list actual inventory locations. This may or may not be needed, depending on circumstances. ¹¹⁰

Second, you'll notice that on the MasterPartsPlan form itself there is a green vertical stripe, located just to the right of where the part items are listed. The general idea is that if you want to list any extra information for an line-item in the list (such as the quantity that you wish to maintain on a particular specialized plan, for example), you may simply click adjacent to it upon this green stripe. At this point you'll see a nice little window that allows you to add all the various kinds of information you may want—including even *notes* about the part if wanted. You can think of this, in a sense, as a "MoreInfo" form in regard to part listings—only this one we refer to as the *Supplemental-Info* window.

Using the Supplemental-Info Window is quite straightforward. Most functions should be obvious, but one element requires explanation. Suppose you've just created a specialized plan type (from the *InventoryLocations* form), and now need to indicate, for each part line-item, the quantity you want as minimum under that plan. If you have several hundred line-items in your overall list, this can be a lot of work. Happily, there's a shortcut. To illustrate, consider a client who (and as is typical) had a bunch of trucks all setup for the standard/default plan (with intended minimums appropriately filled-in within the intended column on the face of the MasterPartsPlan form). This client had a special need, though, in that on two particular trucks he wanted to stock several score more items, oriented for high-end work. It's for those trucks that he needed a specialized plan, but the plan as needed overlapped with the standard plan, but added more. Why should he have to directly populate each such item's specified quantity (within that plan's minimum-quantity box in each line-item's Supplemental-Info window)? This is where the shortcut comes in. If you look at each specialized plan, minimum-quantity box, you'll see it has an *asterisk* ("*") next to it. If you click on any such asterisk, the system will offer to set all quantities under that plan type the same as for "standard" trucks. It's to give you a potential head-start in populating for quantities as particularly wanted for the specialized plan.

Upon having entered any information into a parts listing's Supplemental-Info window (and exiting back out of that window), you'll notice there's now either or both of two single-letter references listed adjacent to that part listing within the green vertical stripe. If there's a 'T' it means there's quantities given (that's intended stock quantities) for one or more kinds of specialized Trucks. If there's an "O" it means there's **O**ther kinds of additional info.

In regard to the specialized plan function, the idea is you'll still use the one master list of parts, just as when using the basic system. You simply must include within this master list an entry for any and every kind of part you intend to maintain in stock, even if for many listings a particular item will perhaps be stocked within only one kind of location. For any such item, you'll indicate through this advanced feature how many you want

¹¹⁰By default, ServiceDesk assumes you have an office storeroom as one inventory location, and that each tech in your Settings form's roster of technicians is also an inventory location. It maintains this assumption so long as no locations are explicitly listed in this section. If you list any here, it drops the assumption regarding each tech, but maintains its assumption regard an office storeroom. For this reason, please realize that if you explicitly list any locations here, you'll need to list all, except the office storeroom. The need to explicitly list arises in either of two circumstances: (1) if you opt to use any specialized plan types, because it's only via the explicit listing you can tell the system which plan type any particular location (aside from the office storeroom) should be using; and (2) if the default assumption (an office storeroom is a location, and so is each listed tech) does not fit your circumstance (e.g., some techs are not inventory locations, or perhaps you have one or more other storerooms, besides the one in your office).

maintained as minimum. For any standard location use the indicated column in the main list, etc. Via this system, you can explicitly structure up to eight unique stocking plans (one for the office, one as the default truck type, and six more as added/specialized types).¹¹¹

D. Tracking and Depositing Funds

As important as it is to keep track of your stocked-parts inventory, there's just one ultimate reason why: to facilitate the proper and efficient completion of jobs—which is done, obviously, so you can collect money upon completion—money that produces the income that is your *raison de etre* for being in business. It would be a shame, to say the least, if while carefully regimenting every other aspect of your business, you then failed to positively account for, and track, every item of incoming money.

It is a reality that many small businesses lose shocking sums, without even knowing it, because of undiscovered employee embezzlement or simple misplacing of incoming funds. Obviously, a means is needed to document and track each item of receipt, whether consisting of check, credit card draft, or sum of cash—to assure each reaches deposit to your bank, or at least to a deliberate disbursal.

The primary venue for this purpose is the FundsForm, accessed by pressing **Ctrl-F9**. Among its many other functions, the FundsForm displays the FundsJournal, a file designed to contain a unique and separate entry for every check, bankcard draft, or sum of cash your business takes in. Of course, for it to have all these entries, there must be a means for it to acquire them.

¹¹¹When this design was formulated, we did not anticipate anyone would need a larger quantity of unique plans. Turns out, however, some users want unique plans for each truck, and they have more than seven trucks (out of eight potential plans total, one is for the office, which leaves but seven unique plans remaining). Since we always seek to satisfy, we are planning a system re-design that will accommodate such per-truck (and large quantity) uniqueness. In the meantime, though we provide this footnote to address the question (if yours is the situation described) you best utilize the system with its current constraints?

We suggest one of two general strategies:

1. Do not rely on planned minimums (aka “re-stock points”) for at least some of your trucks. To understand why this is practical, we'll first assure you understand a distinction. Sometimes people confuse a stocking plan (which is all about re-stocking points) with an inventory tally (which is all about what's actually possessed). When talking about a present maximum of eight unique plans, we positively are not implying there is any similar limitation in the quantity of locations ServiceDesk will accurately, completely and uniquely manage in regard to actual tallies. There are no such limitations in this second and separate respect. Any “plan's” purpose is solely to facilitate re-stocking, for the sake of bringing inventories back up to minimums when they've fallen below — and (as my dad used to say) there's more than one way to skin a cat. In particular, there are two ways to accomplish restock.
 - a. One method is by using planned minimums, as principally-described in the text (the system shows you items where tallied quantities are below plan levels, and prompts refill accordingly);
 - b. The second/alternate method is by picking an option, in the F10 InventoryControl form, where SD prompts for refill on the basis of items prior used. So long as you simply begin with a particular location at desired quantities (and add as desired), this works fine, and does not in any manner depend on minimum/restock-points as present in a plan. The one downside is if you decide to no longer stock an item at a particular location, you have to remember to refrain from refilling when performing the next re-stock process.
2. The second option would be to group some of your trucks into shared plans. Suppose, for example, you have 13 trucks. Suppose four have stocking needs sufficiently similar that it's practical for them to use the same plan. Another three are likewise sufficiently similar to one another. Another pair is likewise. This means you could directly plan restock points for nine of your trucks with just three plans, leaving four plan spaces available for your remaining four trucks. It might not be quite as ideal as totally unique plans for each, but may be reasonably workable.

i. Entering Funds Collected.

In this regard, ServiceDesk recognizes two broad contexts in which a typical service company collects funds. First, is where moneys are collected either before or concurrent with job completion, as is always the case with COD jobs, and may happen to some extent even with billed jobs (as when collecting deductibles on a home-warranty type job, for example). In this context, the funds are usually collected by a technician, and it's similarly him who brings them to the office. The second context is where the job was completed with funds still owing, so the job is billed, and payment is received thereafter. Here, payment usually comes by mail, and is received directly by the office. Depending on the context, the processes for making entry of the receipt within ServiceDesk (and assuring that all appropriate entries are made) are considerably different.

The first context is addressed, for the purpose of entering funds received, via the PostVisitReporting process (see page 110). As you may recall, one of the queries you (or your reporting tech) must answer, during this procedure, is whether any funds were collected from the customer, if so what kind, how much, and so on. Based on your answers, ServiceDesk will make an appropriate FundsJournal entry for you, regarding all details of the fund received. This reporting venue is used, for funds received prior to or concurrent with job completion, for several reasons. One is convenience. Since most collections in such context are done in the course of a technician's actual visit to the home, it follows that you'll be making a PostVisitReport, in such cases, regardless. Thus, no separate effort is required to report on the money received; it's simply done in consequence of other procedures you're doing already (for those few items collected separate from a scheduled visit—yet with a job still pending—you can still use the same PostVisitReporting context). This venue also allows ServiceDesk to make a redundant recording of the collection within the job's History, making the narrative there, as it pertains to the particular job, more complete and informative.

In the second context, when payment is received after a job has been completed and recorded to your SalesJournal (in this case it will have been recorded as '*billed*'), you should note that its JobRecord will have already been closed out. Thus, the PostVisitReporting process will no longer be available as a venue for making reports on the job. Thus, a different ServiceDesk venue is needed for recording payments that come in on a job that was formerly completed and billed. Here, we simply use a feature, directly within the Funds form, that's specifically provided for the purpose (press Ctrl-F9 to access the form, then select its '*rcv Payment on frmly billed job(s)*' option).

Basically, the system will walk you through a procedure in which it collects info about the particular item of money received (most always either a check or credit memo for parts used, in this context), then asks about each invoice the payment should apply toward, in what portion, and so on. In consequence of your answers, it makes entries in several files. First, it makes a single entry to the FundsJournal regarding the particular receipt you've reported. Second, it makes as many entries as are needed (some checks obviously apply to more than just one invoice) to a file called the ApplicationsJournal (press **Alt-F9** to access it). Here, ServiceDesk keeps a running record, solely for your benefit, of how you instructed it to apply every portion of every check received for such purpose. Third, it makes an entry to every relevant AccountsReceivable record, showing application of the indicated amount. And finally, if such application satisfies the total due on a receivable (most often the case), it makes a PayCode 3 entry to your SalesJournal, which of course documents the fact of final and complete payment (see page 169 for an explanation of PayCodes). Thus, a lot of entries are made in various files for you, based on your execution of a single, simple procedure.¹¹²

¹¹² A distinction worth noting is that, where funds are received in the first context (and reported on via the PostVisitReporting process), the resulting FundsJournal entry will always match the recorded receipt to a particular invoice number. Where reported on in this second context, by contrast (and reported on via the FundsForm's own internal procedure), the resulting FundsJournal entry will be matched with a specific invoice *only if* the receipt applied to just that invoice, and no others. If it was applied to multiple invoices, the field that would otherwise indicate an invoice number will simply read all zeros (i.e., it will seem to indicate '00000' as the applicable invoice number). For an indication of which invoices the item applied to in this case, you'll have to consult the ApplicationsJournal.

To recap, there are two general methods by which we report on each item of money received. For items received in connection with jobs that are not yet recorded to the SalesJournal, we use the PostVisitReporting process (each such report results in a narrative description in the job's History, as well as the all-important FundsJournal entry). When receipts come in after a job's been recorded to the SalesJournal (meaning such receipts must be in payment on billed jobs), we use the specific function, provided for this reporting purpose, in the FundsForm (in result, an appropriate and specific entry is again made to the FundsJournal, together with corresponding entries in plethora of other files).

ii. Providing Absolute Funds Security.

Now that we know how ServiceDesk collects information for its detailed record concerning every item of money received, the next question is how it uses this data to insure you never lose any such item of money without knowing it, nor that you ever mistakenly credit yourself with receiving a fund when in fact you haven't, nor that you ever record as ostensibly paid a sale that in fact wasn't, nor that you ever initiate a job that doesn't end in a properly documented sale with funds collected, etc.

Let us first examine the last-mentioned security need: how do we assure that for every job created, there is an internally-documented resolution showing its proper conclusion and sale amount? This purpose is met, most conveniently, by the WorkInProgress system. Quite simply, once any job is initiated in ServiceDesk, its newly-created JobRecord goes, unavoidably, into the JobsCurrent file—and there it stays, inexorably, until it is finally released by the act of it's recording to the SalesJournal. If some tech absconded with a job you initiated and called it his own, therefore, or if he simply lost it, the result is that the invoice doesn't come back for recording to the SalesJournal. And without this act, the job's record remains in the JobsCurrent file, attracting attention as it gains quickly in age (particularly if you use the WipAlert system, see page 118).

For our next security need, let's consider how ServiceDesk assures that no sale is ever recorded as paid, unless in fact it has been paid. This security need is provided somewhat differently depending on whether the job is being recorded as already paid at time the sale is recorded, or as being first recorded as billed, then later paid.

In the first context, ServiceDesk fulfills its automatic sentry function by making its own little check each time your operator attempts to make a PayCode 1 sales entry (the type indicating a job is already paid). Quite simply, it peruses the FundsJournal to assure adequate receipts can there be found, for the indicated invoice number, and matching the amount indicated for the sale. If it cannot find such amounts, it simply disallows the entry (if too much is found, on the other hand, it provides notice of the fact so corrections can be made).

In the second context, it should be understood that ServiceDesk automatically creates a unique Accounts Receivable record, for each and every billed job, each time a billed sale is recorded (PayCode 2 in the SalesJournal). Significantly, this A/R record can never be credited with any payments (absent use of the owner/manager password, at least) unless it's done, by ServiceDesk, through the specific process that's provided, in the FundsForm, for reporting on billed funds received (see description several paragraphs back). Of course, this process in itself results in appropriate and matching entries being made (reflecting receipts adequate for any A/R credits) to the FundsJournal. And further, a billed item can't be finally recorded to the SalesJournal as paid (a PayCode 3 entry) absent the same process. In consequence, in both this context and the first, it's impossible for any non-password-equipped staff member to credit any sale with payment, in any context, unless in fact there are entries in the Funds Journal reflecting at least ostensive receipts in an adequate amount.

An incidental concern involves the fact that many times billed jobs are already partly paid when the sale is initially recorded (particularly in the case where you've received home-warranty deductibles). As an item's Accounts Receivable record is created via the SalesEnter system (see following section), it must show that partial payment. At the same time, we want to assure that no greater partial payment is shown than has actually been received (or at least as is shown for claimed receipts in the FundsJournal). This is done, quite simply, by ServiceDesk performing the same kind of perusal as when a paid job is initially entered. It checks the FundsJournal and will not allow the entering operator to claim more, for recording as partial payment in the item's Accounts Receivable record, than is reflected there.

By these various means, we assure with great certainty that we'll know about any job which doesn't make it to final and proper recording via the SalesEnter system. And further, for all jobs that are recorded as having been paid before or at time of completion, we assure they cannot be so recorded unless there are corresponding entries, reflecting adequate receipts, in the FundsJournal. And, for all jobs that are recorded as being billed, we assure they cannot be so recorded absent creation of a specific Accounts Receivable record for each (which cannot show even partial payment absent adequate receipts showing in the FundsJournal). And, before any Accounts Receivable record is credited with subsequent payments and then closed (with corresponding final entries being made to the SalesJournal), we again assure that the FundsJournal must show at least ostensive receipts adequate for the specific application.

Thus, and in consequence, the FundsJournal becomes the place where we maintain record of, and verify, the adequate collection of funds in connection with every sale. At least, we should say, it's where we have entries reflecting *claimed* receipts that are adequate for such purpose. Whether you actually have the ostensive funds in possession or not, of course, is quite another question, involving still another step in the verification/security process.

This additional step is achieved without the slightest additional effort on your part, as a built-in aspect of the last thing you typically must do, regardless, in specific reference to funds received: depositing them.

Specifically, within the Funds form ServiceDesk provides a dedicated process for preparing your bank deposits (whether consisting of cash, check or bankcard). To use this procedure, begin by loading the form (press Ctrl-F9), then select its '*Assemble deposit/items report*' option. At this point, you'll be asked what type of money you wish to make a deposit on, then shown a list of all such items that are awaiting deposit, and asked to click on those you actually intend to deposit at present (or you can simply enter a name or check number and have the item selected for you). At conclusion of the sequence, ServiceDesk will print a deposit document for you, listing each of the items and total, with a statement at top indicting your business name, date, and bank account number. Typically, it's a good idea to print two such copies, one for the bank and another to provide a hard copy in your own records.¹¹³

¹¹³ Recognizing that cash is often separated out of the regular flow of funds in a small business, ServiceDesk prepares a report in this category that is labeled somewhat differently, describing the transaction, that is presumably prepared by an underling, as "Cash Submitted to the Owner" on such and such date. In this way, the owner can verify that such cash was actually received, and can personally shepherd it to the desired ultimate destination, whether it's the bank or otherwise.

If you're a business that does OEM warranty work, you may have still another concern. Many manufacturers don't reimburse directly for parts used (i.e., they don't cut you a check to pay you back for them). Instead, once the underlying claim is accepted, they'll have the distributor from whom you purchased the part credit back your account for the its cost. There are essentially two ways you could deal with this. One, when completing each sale in such a situation, you could internally record -0- as the amount used in parts (which makes sense, since essentially you've not "sold" the manufacturer the part at all). Or two, you could go ahead and make the parts amount (even if at cost) part of the completed sale, and when you eventually get the expected credit, apply it as a kind of "fund received" via the same Funds Control system as we are discussing. In fact, our Funds Control system has had the ability to handle parts credits—as though they were just another kind of fund—since about the end of 2002. You can simply follow the same essential procedures, in their regard, as you do in regard to any of the other kinds of "funds." An advantage of this second method is it gives you the benefit of the Accounts Receivable system as a method to assure that those credits ultimately come in, and in the expected amounts, etc. Same thing applies if you receive payments even for labor via Electronic Funds Transfer (EFT).

Security flows from at least two factors here. First, because your operator sees a listing of all pending receipts during the deposit process, it's immediately obvious if any are physically missing (of course, she should immediately bring this to your attention). Second, because no item will check-off in the FundsJournal as having been deposited absent its journey through this process, it follows that its listing will remain in there, under '*undeposited*' status, until and unless the process is actually completed for it. Thus, if the process is not completed for any particular item of claimed receipt, it will continue to show there, attracting ever-more attention as it gains in age—refusing to die until someone pays proper attention to the fact that you have, as a claimed item of receipt, an item which never has made it through even ostensive deposit/disbursal.

Of course, absent still additional measures, it would be conceivable that an unscrupulous operator might go through the deposit-preparing process itself (thus checking-off several receipts as having been supposedly deposited), without actually taking the money to the bank (or even that the bank might itself lose the deposit). Thus, still more verification is needed.

This last purpose is achieved by virtue of the fact that, when your operator completes that deposit-preparing process within the FundsForm (with ServiceDesk simultaneously checking-off each included receipt as having been supposedly deposited), at the same time it makes still another FundsJournal entry. Specifically, it makes a '*deposit*' entry: a line indicating that on such and such a date so many checks (or other kind of receipt item) were deposited, totaling such and such amount. This new entry then becomes something else that must be checked-off, or else it will attract attention as it gains in age. And significantly, a mere operator does not have the power to check this item off. On the contrary, this process requires use of the owner/manager password.

Specifically, it is the responsibility of the owner (or other completely-trusted person) to periodically run, from within the FundsForm, the procedure labeled '*Confirm deposit/item reports*'. Typically, you should complete this ritual on a monthly basis as part of the process when reconciling your monthly bank statement. Quite simply, ServiceDesk will display a list (within any category, whether cash, check or bankcard) of all claimed deposit/item reports that an operator has ostensibly made, and which has not heretofore been confirmed by you. You can simply look at your statement and confirm if, in fact, that item was received by the bank (or received by you, if that's how it was disbursed). If so, indicate the confirmation. The item will then be checked-off (only with use of the necessary password), and of course will not re-appear during subsequent procedures. The beauty is that if any claimed deposit process didn't actually get where it was supposed to go, you won't be checking it off, and thus will see the fault right away during completion of this process.¹¹⁴

Thus, we have complete security, assuring there are actual funds for all ostensive collections listed in our FundsJournal, and that all these are properly deposited—or we'll know about. Of course (and by virtue of other measures described), we also are confident that, once initiated, all jobs will eventually be recorded to our SalesJournal—or we'll know about it. And, naturally, we're also certain there will be entries within our FundsJournal adequate for each sale recorded—or we'll know about it.¹¹⁵

¹¹⁴ You might note that while ServiceDesk tracks each item of receipt, assists in preparing and verifying the deposit, and so on, it does nothing to communicate deposits to the ledger where you actually keep track of your bank account balance. You'll need to make your own entries in connection with whatever system you separately use to document funds to and from your bank account.

¹¹⁵ It might seem that most of this security could be provided simply by totaling all receipts for a period, and all sales, then making sure the two match. However, with a little consideration it should be obvious that such figures will almost never match, because of the many discontinuities, in time, between the moment of sale and collection. This happens not only because so many jobs are billed, but also because we so often collect deposits (for ordering parts, and such matters), deductibles, and other funds long *before* a job is completed, and its sale registered. There are various and complex methods, in financial accounting, for dealing with such discontinuities, but even so, they depend on laborious bookkeeping methods that ServiceDesk happily circumvents, and their end result is inferior, for the most they will immediately disclose, when some fund is lost, is that two large figures don't jibe—leaving it to you to search and scramble in the effort to find out why. ServiceDesk, on the other hand, immediately discloses the precise source of discrepancy, whatever it is.

It would be difficult, to say the least, for an unscrupulous employee to find a way around these means (providing you keep the owner/manager password secret), or even for losses by carelessness to go unnoticed. That's security very few small businesses possess (at least in the absence of an owner who personally, and with great care, handles every item of money, every deposit, every claimed sale for adequacy of payment, and so on). Indeed, since even a careful owner might lapse into occasional oversights, security would still be less complete, in such a case, than ServiceDesk automatically provides.

iii. Dealing With Abnormal Situations

The above was a broad overview, and will provide sufficient knowledge for most of the situations you'll encounter when managing your funds. Even so, there are some abnormal situations you may sometimes need to deal with. If so, you'll need to know how.

As one complicating factor, for example, you may have a client that routinely pays amounts different than you've billed (G.E.'s home-warranty administration is infamous for this). Sometimes they might pay too little, sometimes too much. You may find you're able to apply excesses on some invoices toward deficiencies on others, and keep a tally that's approximately adequate, at least, for overall needs. It may be, indeed, that it's much easier to simply keep a tally of their overall pluses and minuses, and check-off each item as though paid for the right amount, so long as the net position stays reasonable. ServiceDesk accommodates this need with what it calls an Errors and Discrepancies account. The process is self-explanatory, in the FundsForm, from the context of reporting on payments received on a billed job (an EDA account can be viewed, for any client on which you're created one, from the Applications form, accessed by pressing Alt-F9).

Aside from customers that simply "goof up" on the amount paid, you may have some to whom you routinely grant a discount in return for rapid payment (AHS, for example, offers more rapid payment in exchange for a 2% discount on each invoice's face amount).¹¹⁶ This complicates the process of reporting on payments received, because you'll be wanting to check-off each applicable invoice as "Paid In Full"—even though, in fact, you've received 2% less than its face amount. The solution: When reporting the check's amount, simply include a plus sign (i.e., '+') after the entered number (as prompted on the form during the procedure). This tells ServiceDesk the payment is a discounted one, and in response it asks you the discount rate. Then, for purposes of crediting each applicable item with payment, it pretends the check was for the full, non-discounted amount. Following this, as you conclude reporting on each item to which the check should apply and approve recording the transaction, ServiceDesk makes an entry in your SalesJournal reflecting the discount you granted the customer. This is a *PayCode* 5 entry. When you generate a SalesReport later on, the total of such discounts, if any, will be included as part of the report. How you handle this figure within your own financial accounting is up to you (see page 302 for tips on how to input such information to your own separate system). Simply be sure it's factored in.¹¹⁷

Another complicating factor arises when you're working on two invoices for the same customer at once, and they pay for both simultaneously with just one check. Or, you might be finishing one job and at the

¹¹⁶ Though it's none of our business, we cannot resist the opportunity to here comment on the fact that if you're paying 2 percent to get your money even 30 days sooner (in most cases the acceleration is more like just two weeks), you've struck a *very bad* bargain. Compounded to an annual rate, you're paying 27 percent for possession of the money during that 30 day period (if it's only a two-week acceleration, you're effectively paying 67 percent). Not even most loan sharks charge that much. It would be far smarter, if you *need* the funds so soon, to go get a loan at some rate that's more palatable.

¹¹⁷ May we suggest two possibilities. First, you may treat the discounts granted as a *reduction in total sales* (your SalesReport will provide a "Net of Sales After Reduction by Discounts" figure for you, making it easier to pull this number). Or, you may treat the Discounts Granted as an *expense*. Either way, the bottom line on your Profit/Loss Statement (not a product of ServiceDesk) should reflect the appropriate adjustment.

same time open a new invoice for something else, and they pay for both the completion and a deposit on the second invoice with one check. In either case, you've got one check that needs applied in part to one invoice, and in part to another. While there are easy, built-in provisions for dealing with multiple fund applications in the post-completion, billed job context, nothing is inherently built-in to handle such realities when the appropriate fund-entering context is via the PostVisitReporting system. Of course, in this context the need for multiple application of a single fund happens only rarely, so we get by with a mere Band-Aid approach.

Specifically, we create a supplemental pair of FundsJournal entries. The first is labeled "MoveAppFrom" and the second "MoveAppTo." Essentially, if a single receipt applies to two invoices (remember, we're only talking about the context of non-billed, COD jobs here), we'll have one FundsJournal entry that credits it, initially, to just one of the two—then, in addition, this pair of adjusting entries that transfers an appropriate portion of the receipt's application from the first invoice to the second. These "MoveApp" entries will be made automatically in result of queries made at the time the receipt itself is entered through the PostVisitReporting process, but can also be added, should the need become apparent, directly from the FundsJournal. You'll see an option provided there for the purpose (this is another operation, incidentally, that requires use of the Owner/Manager password).

Another special situation arises when you've made *refunds* to customers. Obviously, you'll want to document having done this. The Funds form provides an option for the purpose, and it too is self-explanatory.

Still another situation arises when some item of money proves to be uncollectible. Perhaps you attempt to run a charge against someone's bankcard, for example, and the charge is denied (and you're unable to get the customer to give you a better number). Or maybe a check bounces irretrievably, or an item of cash is physically lost. In any such case, you need a means to document the sad outcome. As in many other ServiceDesk contexts, you can *delete* an item in the Funds form simply by right-clicking on it. Of course, if writing-off a bad fund, mere deletion is not adequate. While we don't want the item to remain in our list demanding deposit, we don't want it removed, either, without documenting the fact that we've incurred such loss. For this reason, when you right-click on an item from in the Funds form, you'll see three options. One, as in other contexts, is for simple deletion of the item (see following section for explanation of circumstances where you might want this). In the alternative, you may "Write-off the item as Uncollectible," or to write it off as "Missing." If you choose either option, the item will be marked as requested—plus the system will make an entry in your SalesJournal indicating the loss. This is a *PayCode* 6 entry, and the total of such losses is another matter that will be reported, for use in your accounting, whenever you compile a SalesReport.¹¹⁸

A final situation involves the occasional need for entry of some fund receipt, into your SalesJournal, absent any of the more automated means previously described. Perhaps, for example, you received a bad check which was replaced either with a new check or some other form of replacement money. This new item needs entered to your FundsJournal (for proper tracking and deposit), yet neither of the normal entry-creation venues (see section I, this division) are appropriate. For this and any other miscellaneous need to manually create a new receipt-of-fund entry, you'll find facility in the Funds form for such purpose (a command button in lower-right corner labeled 'Add items'). If you're adding a new entry in replacement of an old, incidentally, be sure that you also delete the old entry.

¹¹⁸Please note that when you invoke this action, ServiceDesk will automatically invite you to create a *Special Situations Advisory*, thus "Red-Flagging" the involved customer as someone to watch out for in the future (see page 229). The other context that involves write-offs, by the way, is in the AccountsReceivable form (see page 197). The difference, obviously, is that there you're dealing with money, owed to you, that was never even ostensibly paid by the customer. With write-offs in the Funds form, by contrast, you're dealing with funds that someone supposedly gave you, but which ultimately proved worthless.

iv. Housekeeping in the Funds Form.

For the most part, keeping your FundsJournal neat and tidy will involve nothing more than routine use of the operations described above. However, you should understand that doing this involves something of an ongoing housekeeping task. It's something you have to pay attention to.

In particular, and in addition to attending to the various processes already described, you'll probably notice the same thing we have: occasionally spurious entries will appear in your FundsJournal. These are not spurious in the sense that they came out of nowhere. Instead, what typically happens is that someone is making a PostVisitReport. They report on a fund received, then realize they did it wrong, so report again. In consequence, the Journal ends up with *two* entries for the fund receipt, one accurate and one spurious. When your processing person prepares a deposit, she'll find an actual fund item that corresponds with one entry, but not with the other. In the context involved, it will be fairly apparent to her that one entry is a spurious duplicate, and she'll logically not attempt to include that one in her deposit. If she's clever, however, she'll add some notation to its text, indicating her conclusion (as in many other contexts, left-click on any item in the list to edit it). In consequence of not being included in the deposit, naturally, the item is not so checked-off. Thus, until something more is done, the item remains in the list as (supposedly) needing to be deposited.

This is why, as the password-equipped owner/manager, you should periodically check the listings in your FundsJournal for such items, and remove any that are verified as being spurious duplicates. To do this is fairly self-explanatory. First select the 'View/Edit Items' option, then 'All' as the Kind of Fund to be viewed, then 'Receipts,' then 'Undeposited/Unverified'. Look toward the top of the list (spurious items will eventually migrate there as all contemporaneous and older items get moved out of the category by virtue of *having been* deposited) for items that should have been included in previous deposits but were not. Hopefully, your deposit preparer will have added some notation indicating his belief that the item is a duplicate or otherwise in error. You'll want to verify the conclusion before deleting the entry. To do so, you'll want to see what other entries have been made in connection with the same job. Just note the invoice number the payment is connected to (part of the entry information), then do a "search" in the form to find all entries pertaining to that number. Maybe, for example, you'll see there's two entries of \$40 each. If you then hit SHIFT-F3 and search on the same InvoiceNumber in your SalesJournal, you may confirm that the total sale was only \$40. Thus you'll know that the second of two entries must have been a goof by the entering person, and you can go back to its listing, and right-click on it to invoke a deletion.

On the other hand, of course, you might find there are older items in the listing, that should have been deposited already, that are *legitimate* entries. This, obviously, will incite a prompt discussion with your deposit preparer. What's the deal, you'll ask? Why is that item still lingering there, without showing deposit? Preferably, if there was cause for any legitimate item to linger (for us, it's sometimes been a missing check and the preparer has been trying to get a replacement from the customer, or maybe a bankcard number that wouldn't clear, and she's still hoping for eventual success), your deposit preparer will have informed you already, so there'll be no surprises. However, you can see that it's important for you to go through this inspection process, periodically, just to make sure.

At any rate, you may note that several ServiceDesk contexts have systems for policing you, attempting to cajole (and sometimes coerce) you into keeping on top of various tasks. The Funds form (and its associated FundsControl system) is no exception, except here it's as a mere *incidental byproduct* of how the system is setup. As in many other systems that keep track of records over time, there is also the need here to separate the newer, actively working area of records from that which is merely historical. In this case, however, rather than having separate files (i.e., a *current* file for active records and *archive* file for historical ones), we thought it more convenient to use a single file and moveable *partition*. Basically, the system tries to

keep track of how far back, in the list, is the oldest unresolved (i.e., not checked-off, written-off, deleted or otherwise disposed of) item. All entries more recent than that are considered *current* area, and all older considered, well, *old* area. It helps the system perform many tasks more quickly than otherwise when it knows that any current items must be on one side of the dividing line.

The reason this system helps to police you is because, *if* you allow older and older items to accumulate without resolution, the system will not be able to move the *Current-Area Partition* forward as it would like to. Every time you attempt to load the Funds form, in consequence, it will inform you the CurrentArea has grown to large. It will ask for permission to attempt to move the Partition forward. If you've left old items back there, however, it will be unable to, and will so inform you. Thus, next time you go to load the form, it will inform you again. This gets to be very annoying, and should prompt you to start cleaning out those old, unresolved items (perhaps it's simply deposits you haven't confirmed yet; regardless, it's easy to find out by designating the appropriate display).

We are, quite frankly, proud of how our FundsControl system ties together so many other elements in ServiceDesk, providing such positive security, along with convenience for your funds control. We hope you enjoy it too.

Chapter 8

POST-COMPLETION MANAGEMENT

When the technician finishes a job, he of course is generally done with it. But not the office. We still have additional work to do, and that is the subject of this chapter.

Hint: If using the video tutorials, please read this chapter in conjunction with having watched Lesson # 5.

A. Recording Sales, Tabulating, etc.

While collected moneys are your business's lifeblood (and expected surpluses its reason for being), it's no secret that such collections depend, ultimately, on something even more basic: completed sales. It's further no secret that sales figures are the first input into the accounting process—which, in turn, leads to the financial statements by which you can tell whether, in the long run, you can expect to have any money surpluses at all (and on the basis of which you'll calculate income for paying taxes, etc.).

i. ServiceDesk Utilities vs External Accounting

Here again we tread in an area that is already well-traveled by other software. In particular, there are many applications that allow you to record individual sales in the context of a larger accounting package. We have found, moreover, that many ServiceDesk buyers are already setup and practiced in using these other systems. Thus, the question arises as to how much, if at all, your ServiceDesk sales-recording features should supersede those already used in an accounting context, or if they should be used in a fashion that duplicates those already used, or not used at all?

In regard to the possibility of using the ServiceDesk sales-recording features in a fashion that duplicates entries still being made in another context, the answer is simple: DON'T DO IT! No one wants to enter the same information twice; it's just doesn't make sense (except possibly during a limited transition period, when you're testing ServiceDesk features to assure they're adequate to your requirements).

In regard to the possibility of simply not using the ServiceDesk sales-recording features and continuing to rely, as you presumably have in the past, on another application for the purpose, the possibility exists, but you should be aware that you'll be sacrificing several elements of automatic-interface between various aspects of ServiceDesk operation. The normal method by which JobRecords are checked-off as ready for archiving out of the JobsCurrent file, for example, is via your entry of a completed sale through the SalesEnter system (at which point ServiceDesk automatically checks the corresponding JobRecord into "Recorded to SlsJrnl" status). Absent your

use of the SalesEnter process, another means is needed for this indication.¹¹⁹ Similarly, if you do not enter your completed sales in the ServiceDesk system, there will be no means, therein, for automatically verifying that adequate funds are collected, for creating an AccountsReceivable file, for generating automatic Billing reminders, creating Commission Reports, searching past sales in the SalesJournal, and so on.

Since so much utility would be lost in the absence of using the ServiceDesk sales-recording features, we think such a course would be inadvisable in all but the most unusual of circumstances. Instead, we suggest that regardless of what package you've been using in the past, you now rely exclusively on ServiceDesk features for the purpose of recording individual sales. If there are things your old package did in connection with individual sales that ServiceDesk does not yet do, and you really want those features, let us know and we'll try to incorporate them in an update that we'll rush just for you.

Of course, ServiceDesk will not do your financial accounting or the expense side of your bookkeeping. Yet, doing your sales and accounts-receivable is, essentially, taking care of the *revenue* side of your bookkeeping (at least that portion involved with performing service). This raises the question: How do you integrate ServiceDesk (and these revenue-side functions of accounting that it's performing) with whatever other system you have used or will use to do the balance of your bookkeeping and accounting?

At first blush, the problem might seem a difficult one, but in fact it's very simple. If you've been using QuickBooks or almost anything similar, chances are that you were formerly recording each of yours sales, *individually*, within it (and managing each of your accounts-receivable there, etc.). To move into ServiceDesk for those functions, you'll simply cease to record individual sales transactions in that old context, instead using the facilities provided within ServiceDesk. Then, as often as you compile financial statements (whether once a month or once a year), you'll run a report in ServiceDesk that summarizes its revenue side activity. You'll then take those *summary figures*, and input them into QuickBooks or whatever other system you are using. (For detailed instructions in how to do this, see the section that begins on page 302).

In other words, the *detail work* for your revenue side activity (recording each individual sale, handling each receivable, etc.) gets shifted into ServiceDesk. Your financial accounting package no longer gets to handle that particular kind of detail. Instead, it just gets a summary regarding that particular activity, and then incorporates this summary into its calculations of profit/loss, etc.

By following this basic outline, you'll be letting ServiceDesk do what it does best, while at the same time allowing your accounting/bookkeeping software to perform where it most excels.

¹¹⁹For this need (if, in other words, you want to use all of ServiceDesk's features *except* its sales and accounts-receivable functions), we've provided a special option within the 'Learning Steps' window of the Settings form for the purpose. Immediately beneath the Learning Mode 5 line there (which normally would add sales and accounts-receivable functions to the mix), you'll see an option labeled "*Leave out this function for the time-being.*" This allows you to go ahead and designate other functions, normally beyond this one so far as "Learning Steps" are concerned. Operationally, the effect is as follows.

Normally, when you set to any level of functionality beyond (and including) Learning Mode 5, the system expects that a *sales entry* will eventually be made, on each and every JobRecord. Without such an entry being made, no JobRecord will ever be moved (as normally occurs during a ArchiveJobs procedure) out of the JobsCurrent file and into the JobsArchived file—even if it's otherwise checked off as having been done. The reason is a matter of security, for this built-in reluctance assures a final accounting into the sales system for each and every JobRecord that's created. If, however, you've elected not to use the sales system as built into ServiceDesk, that security becomes a matter of some nuisance, for without the action of having the completed sale internally recorded, those JobRecords refuse to get moved into the JobsArchived file, and the JobsCurrent file soon grows heavy with jobs that have, in fact, long been done (aside from not having been recorded to the SalesJournal—which, were assuming for purposes here, you're elected not to do regardless).

In contrast to that normal situation, when you check the "*Leave out this function for the time-being*" option, the system will understand you are not presently intending to use ServiceDesk's sales system, and so won't expect JobRecords to be so recorded prior to being moved from the JobsCurrent file and into the JobsArchived file. Thus (and with this as the setting) during an ArchiveJobs procedure the system will now check simply to see if a JobRecord has been placed into 'Completed' status. If it has, it will be "archived" accordingly.

ii. The SalesEnter Form.

Having now discussed whether you should use the ServiceDesk sales-recording features in preference to other systems you may already have, we obviously need to discuss what those features are, and how they are used.

Basically, there are three forms involved, one for entering sales, one for reviewing past entries, and one for creating reports. They are appropriately called the SalesEnter, SalesView, and Reports forms (accessed by pressing F9, SHIFT-F3, or F11, respectively, or by clicking on the corresponding MainMenu command buttons).

Beginning with the first of these forms, suppose you have several invoices reflecting completed sales, which now need entered. With the invoices in front of you, access the SalesEnter form by pressing F9. The form will now display, showing your most recent ten entries (the intent is that with a reminder of what was last entered, it may help you avoid even the attempt to enter the same items twice). Your cursor is positioned on the entry line, and there is a note indicating the items of data ServiceDesk expects you to enter from the completed invoice. You might note that you're instructed to separate each item on the single line by a comma (rather than having separate boxes for each item), which is more reminiscent of the old DOS environment than Windows. The reason for this setup is because, for items that are entered in the same order so frequently that the sequence becomes a matter of automatic habit, separation by commas is easier and more efficient than tabbing between boxes. Sales, hopefully at least, will be entered with great frequency, so we want an entry method that is as efficient as possible.

There are, as you'll see, nine items of data requested in a single line of entry. These are the **TechCode** (i.e., two-character initials for the tech completing the job), **CustomerName**, **InvoiceNumber**, **MerchandiseSold** (i.e., if you sold a whole machine that is categorized differently, for your purposes, than mere parts), **PartsSold**, **S.CallSold**, **LaborSold**, **InvoiceTotal**, and **Paycode**. The last item should be a "1" or "2", which designate, respectively, either that the job was completed and paid for, or that it was completed but billed.¹²⁰

There's an important convention that applies in regard to entering the CustomerName whenever you've got a third-party payer job (i.e., one party paying, work is being done for another). Enter the last name or initials of the paying party, then a hyphen (no spaces), then the last name where the work was done. Thus, a job might be designated here as "AHS-SMITH"—providing on-its-face evidence of both payer and service location.

Upon completing a line, press Enter. At this time ServiceDesk will review your entry, making several checks to verify its legitimacy.

First it checks the entry's general form, verifying that you have correct quantity of items entered, that your TechCode and Paycode are acceptable values (i.e., the indicated TechCode must correspond to a tech you actually have, the Paycode must be either a '1' or a '2'), that your invoice number is in the proper format (i.e., five or six numbers, no letters), that you have numerical amounts where monetary quantities should be indicated, that each of the constituent amounts correctly add to the indicated total, and so on.¹²¹ Next ServiceDesk checks in the

¹²⁰ Four other possible Paycodes are created in contexts that are entered by ServiceDesk (i.e., it fills-in the details). "3" indicates that payment was received on a job that was formerly entered as completed but billed. "4" indicates that a receivable has been written-off as a bad debt. "5" is to indicate a discount that was granted a customer. "6" is to indicate fund receipt (i.e., check, cash or bankcard) that was written-off because uncollectible or missing.

¹²¹ In summing the entered constituents, the system internally references the sales tax rates as specified by you from within the Settings form. Based on these rates, it infers appropriate sales tax amounts (without you having to separately enter them), and includes these as parts of the *calculated* total (which is then compared, of course, against the *entered* total). A special consideration arises, obviously, in the event of a sale that is either partly or wholly *tax-exempt*. If faced with this situation, simply enter the constituent and total amounts as always. When the system detects an entered total that is adequate to cover the entered constituents without tax, and yet is not sufficient to include full tax, it will query as to whether a tax-exempt sale is intended. With your double-confirmation of intent, it will complete the entry. There is still another special consideration that arises if yours is a territory that's subject to *varying tax rates depending on locality*. If this is your situation, see the technical section beginning at page 326 for instructions on how to deal with it.

JobsCurrent file to find the record corresponding to the InvoiceNumber you've indicated, and makes sure you've indicated the appropriate CustomerName. It further assures, while there, that you haven't previously reported a completed sale on the item (to help you avoid making inadvertent double sales entries), and even checks in the string of entries being made (not yet recorded to the SalesJournal) for the same purpose. Finally, if you're indicating a Paycode 1 (i.e., paid) sale, ServiceDesk checks in the FundsJournal (as described in the last section) to assure there are entries there showing receipt of funds adequate to the sale.

If ServiceDesk discovers a defect in any of these areas, it will inform you of the precise discrepancy, and coach you in its correction. The beauty of this system is that it's almost impossible for you to unintentionally make an erroneous entry.¹²² This not only enhances accuracy in the record, but also frees the operator from the need to exercise great care. Thus, she can enter data almost recklessly, with great speed and abandon, confident that if and when she errors, ServiceDesk will immediately lead her toward correction.

Typically, an operator will sit down with a stack of invoices representing the previous day's sales. She'll go rapidly from one item to the next, entering each in rapid succession, taking very little time as she gains experience.

You might notice that as each such item is entered, it moves up into the list displayed above the entry line. If the operator needs to, she can move up into the list and edit entries made in the present session. She can conclude the session and record her entries either by clicking on the indicated command button, or by simply hitting Esc (to exit the form), at which time ServiceDesk will ask if she wants to 'save,' allowing her to indicate 'yes' with a press of the Enter key (for many people, a simple Esc/Enter sequence is easier than using the mouse to click on the command button, then hitting Esc to exit the form).

While entering paid jobs is straightforward, there are some additional procedures you should know about when entering billed (i.e., Paycode "2") sales. In such a case, you'll find ServiceDesk expects to create an AccountsReceivable record for each entry, reflecting (in the AccountsReceivable file) the fact that you expect eventual payment, from your customer, on it. Specifically, you'll find ServiceDesk presenting, for your review and approval, the contents of the A/R record it intends to create, including the customer's formalized name (as it would need to appear, in case this customer doesn't pay promptly, on a dunning letter), mailing address, and so forth.

It also shows the amount already paid, if any (as gleaned from the FundsJournal), while requiring you to indicate, in a field labeled "Attn designee or Salutation," the name to be used, should a dunning letter become necessary, following either the letter's "Dear . . ." salutation in those instances where the customer is a person (i.e., you'd indicate "Mr. Smith" for a customer named John Smith), or following "Attn: . . ." on the envelope in those instances where the customer is a business (i.e., you'd typically indicate "Accounting" if you have no specific contact person at the business, or the name of that person if you do).¹²³

¹²²It might be considered redundant that we require entry not only of *constituent amounts* in an invoice (i.e., how much for the service call, how much for labor, etc.), and also the total—for ServiceDesk could easily calculate the latter based on the former, and insert that value itself. It might further seem redundant that we require entry of the CustomerName *and* InvoiceNumber—when ServiceDesk could discover and insert the latter itself (based on its discoveries in the JobsCurrent file), and therefore not need your entry. However, these redundancies allow further verification. Without an invoice total, ServiceDesk would have no means of knowing if the entering operator inadvertently hit a 7 instead of a 5 when entering some constituent amount. With the total, you've got built-in verification that all monetary values were *very probably* typed correctly (the chance of doing a typo on a constituent value, and another in the total that just happens to make it come out right, is pretty small). Similarly, it would be easy to type in just an InvoiceNumber wrong, or just a CustomerName—but the chance of doing both wrong and having the errors produce a match in the JobsCurrent file is pretty slight. Thus, with such tiny redundancies in the entry, we achieve a great deal of security in assuring that what's ultimately accepted is most accurate—even while the operator, again, can be relatively careless.

¹²³ServiceDesk distinguishes between the two possibilities, incidentally (treating them appropriately), based on the same convention as used in Callsheets (i.e., if the first word in the name line ends with a comma, followed by another word, it figures you've got first-name-then-last of a person; if there is no such format, it figures it must be looking at a business name).

When everything is appropriately acceptable in the proposed A/R entry, you can click on the indicated command button or hit Enter to indicate your acceptance. ServiceDesk will then place you in the position for your next SalesJournal entry.

As is obvious from the context, these Paycode "2" entries result in ServiceDesk writing new records not only to the SalesJournal, but also to the AccountsReceivable file. Less obvious, but something you should know, is that with either Paycode entry ServiceDesk also writes a note to the job's History, reflecting entry of the completed sale. This, then, provides on-its-face documentation, within the job's History—of its final step to completion. It further marks the JobRecord as being ready for movement out of the JobsCurrent file and into the JobsArchived file, the next time an ArchiveJobs routine is run.

There are a few other incidentals you should know about.

First, you should understand that we make use of four other PayCodes, besides the "1" and "2" mentioned. These are "3," for an entry that indicates final payment on a job that was formerly billed, "4," for an entry that indicates we've given up seeking payment and are writing off the receivable as a bad debt, "5," for an entry regarding a discount that was granted a customer for rapid payment, and, finally, "6" for an entry indicating that a fund item was written off as missing or unredeemable (i.e., a bad check that could not be collected on, a denied credit card that we could not correct, etc.). These four kinds of entries are all made from other contexts *for you* (see pages 157, 179, 161 and 162), so are discussed here only incidentally. What all these Paycodes do (or rather, the necessary succession of entries under them),¹²⁴ among other things, is to enable ServiceDesk to compile periodic sales summaries (see page 172) that show not only total sales, but what quantities were sold but billed, what was formerly billed and now collected, what was written off in any period, and what discounts were granted. These values have great analytical and informative power in their own right, of course, and besides are needed as inputs for your financial accounting system.

Second, it probably does not happen infrequently, for you, that a job is canceled by the customer after it's written up, but before your tech even gets there. Or it's a callback, and the total sale ends up equaling zero. In either case, be sure to record the item as though it were completed sale, albeit one with a sale amount of zero. It's important to do this, at least, to allow removal of the item's JobRecord from your JobsCurrent file.

Third, please remember there is an alternative mechanism via which you may arrive at the SalesEnter form for the purpose of making entries there. Rather than doing it in a *batch* process as is generally described here (i.e., taking a stack of tickets on which all preceding work, such as Post-Visit Reports, etc, has been done, and making sales entries, one after another, for each item in that stack) you may instead choose to use the *Integrated Processes* option, via which, upon finishing a Post-Visit Report where the job has been completed, the system will lead you automatically—but for that ticket only—into the Finished-Form and/or SalesEnter processes as well (see page 116). Or you can link to the SalesEnter process from the FinishedForm context as a standing-alone type of link (see page 188).

Finally, we have some advice regarding the procedure for handling your final, remaining copy of the invoice (the one you've got left after other copies in the multi-part set have been given to the location resident, sent as a bill to the customer, or simply discarded). As you finish recording a stack of Paycode "1" (i.e., paid) invoices to the SalesJournal, immediately use one of those little date stampers to imprint the present date, and do it in some

¹²⁴To clarify just a little more thoroughly (in case it's not already clear), each and every single job, once initiated in ServiceDesk, should eventually (preferably sooner than later) be recorded to the SalesJournal. Even if the job was canceled, it should still be recorded—with a sale amount of course equaling zero. The initial recording, in all instances, must be either Paycode 1 (i.e., there's *no* amount owing on the job) or Paycode 2 (i.e., there *is* an amount owing). For all sales entered in the first category, this should be the first and only SalesJournal entry ever made. For the second, another entry is eventually required: either a Paycode 3 entry, indicating complete payment when it's finally received, or a Paycode 4 indicating you've given up, and are writing the item off as a bad debt. If neither of these latter entries are made, the job's A/R record (created by ServiceDesk when you made the Paycode 2 entry) will remain inexorably in the AccountsReceivable file.

regular place you've decided on (on the invoice's bottom right corner works well for us). This date stamp then provides subsequent evidence, on the face of the invoice, that it's sale was recorded to your SalesJournal on the indicated date. As you follow this practice each day, placing the resulting set of invoices on top of that generated the previous day, you'll end up with an ever-increasing physical archive of your completed invoices.

Of course, you'll also have your retained copy of *billed* invoices to deal with. Here, as you finish entering a stack of Paycode "2" entries, we suggest you pick a different place on the invoice to imprint the date-stamp indicating your entry. Thus, by its different position, this stamp will indicate the date on which the invoice was recorded to your SalesJournal as done, but not paid. Once this is done, the invoice will obviously go into some place you've setup where you keep them while awaiting payment. When payment arrives, you'll obviously locate the particular invoices it pertains to (from this place), and with these in hand, go to the FundsForm procedure (described at page 157) to enter receipt of the check, and to indicate which invoices it pertains to. As you complete this procedure, we suggest you document its occurrence and timing by date-stamping the invoice a second time, but in this case in the same location (again, bottom right corner works well for us) as you did for jobs that were initially paid. Thus, by examining the invoice afterward, you can see from the first date stamp when the invoice was initially recorded as a billed job, and from the second when it was subsequently recorded as paid.

Once they've been by this final hurdle, we suggest you add these once-billed-now-paid invoices onto the same stack in which your COD invoices are also accumulating, as described above. In result, you'll end up with a stack that contains all your invoices, arranged in the order (as identified by the stamp date on them) by which each was entered into your SalesJournal and (whether initially or ultimately) also paid. In other words, the stack ends up containing all your invoices in the date sequence by which Paycode "1" or "3" entries were made on them (invoices in Paycode "2" status, obviously, did a temporary diversion into your AccountsReceivable pile—until they became Paycode "3"s). You may also include in this archived stack, and in the same order, any billed jobs that you ultimately had to write off as uncollectible (i.e., Paycode "4"s).¹²⁵

While referring to this stack of *fully* done invoices (i.e., they're completed *and* paid for or written off), it should be obvious we're using the term "stack" somewhat loosely. In reality, we expect your stack will be segmented into various parts, conveniently placed in neatly labeled, ordered boxes, possibly shelved side-by-side. The main idea is for you to have this physical archive of all your invoices, added to continually in the particular date-of-entry-to-your-SalesJournal sequence described. This has some advantages, to be described shortly.

iii. The SalesView Form.

Having gone to the trouble of creating a unique entry in your SalesJournal reflecting every single invoice's final resolution in terms of sale (or perhaps a pair of entries, reflecting those jobs first billed then paid), it's obvious you might want to view those entries, maybe do some editing, or possibly search for particular entries in which you might have an interest. Press SHIFT-F3 to bring up the SalesView form. As you'll see, it offers several viewing options, allowing you to view either entire pages from the file (whether beginning at a specified date, record number, first page, last page, etc.), or to search for specific records based on name, invoice number, or total amount of sale.

Regardless of your needs, the form's operation is self-explanatory. When viewing entire pages, you can use the PgUp and PgDn keys to move to adjoining pages. If conducting a search and the found entries fill an entire page, you can resume searching on a clean page simply (as prompted) by pressing any key. Search targets

¹²⁵ This writing-off process is done via the AccountsReceivable form. In regard to the date stamp on such items, we use the same spot on the invoices as when entering Paycode "1"s and "3"s, but we imprint the stamp up-side-down.

needn't be complete: a mere name segment will do, and global characters can also be used.¹²⁶ If you need to edit an entry, again follow the normal ServiceDesk convention by left-clicking on it. ServiceDesk will enclose the item in edit boxes, allowing you to change any field as desired (the value displayed as 'total' will change automatically for you, as values are changed in constituent fields). Just press Enter when ready to record your edits (or Esc if you've changed your mind and want to leave the record unaltered).

You'll notice that the SalesJournal records contain minimal information, consisting of nothing more than what you entered during the SalesEnter process, along with six digits denoting the date of your entry. Some early users have requested more data here, but there's a purpose in the brevity. Because of the relatively small individual record size, you'll be able to accumulate many years of sales data in a single, manageable-size file. Plus, it makes it possible to run an unsorted search through a file containing tens of thousands of sales entries in a reasonable amount of time. More extensive data on individual jobs can be found elsewhere (in the JobsArchived file, for example, accessed through the CstmrDbase system). It is simply not the purpose of the SalesJournal to provide such detail: it's a different kind of record.

As one advantage in this system, you'll be able to do something very few businesses can: lay your hands on a particular invoice (i.e., the authentic paper copy), within only moments of formulating the desire—even if that invoice dates from years ago (while the need to review physical invoices from the past is much reduced, given extensive information in the JobsArchived file, there is sometimes no substitute for having that actual paper in your hands).

To accomplish this legerdemain, you simply must allow your invoice archive to accumulate in the systematic, easy manner described in the last section. Then, when you need to locate a particular item within it, all you need to know is the date the invoice was added. Knowing this, finding its location in the stack becomes as easy (in principle at least) as finding a particular page number in a sequentially numbered book (in this case, your date stamps on the invoices are the substitute for page numbers).¹²⁷ It's in knowing the proper date, to locate in the stack, that your SalesView form comes into play. With its search features,¹²⁸ you can quickly locate the date of any item's final recording (i.e., entries with Paycodes "1", "3" or "4") to the SalesJournal—which date, if you've followed our advice, will also correspond with what's stamped on the invoices, and with their physical sequence within your archive stack.¹²⁹

¹²⁶You might be looking for a completed sale involving someone named "OLSEN," for example, but realize that it might have formerly been entered, wrongly, as "OLSON." You could simply search under "OLS," but that will also yield entries for such names as "OLSHER" or "OLSWANG" (which may not be too serious a liability). To be more selective but still allow either "OLSEN" or "OLSON," specify "OLS*N" and the search will yield both (or even "OLSYN"). A similar tactic can be used if you're looking for a "GREENWAY" that was the consumer-beneficiary on a third-party payer job (such as for a home-warranty company), but you're not sure which company. If you've stuck with three-digit initials for your HVC abbreviations (see page 67), try searching on "****-GREENWAY." Whether the entry is "AHS-GREENWAY," "OLD-GREENWAY," or "FAM-GREENWAY," this target will find it.

¹²⁷The one difference is that you'll have as many invoices sharing an identical "page number" as you happened to add into the stack on a given day (perhaps between 10 and 40 for most small business). Thus, these so-called "page numbers" may get you only to a section in the archive (containing that many items), which you may then have to peruse individually to find the particular item you're looking for—an exercise that might add, on average, another several seconds to your total location time.

¹²⁸At first blush you might think the CstmrDbase would be an even better source for getting these dates, since its underlying JobRecords show the date of a job's recording to the SalesJournal, and its searches are conducted instantly (even though it's a much larger file, these searches are based on indexes, which the SalesJournal does not have). However, the JobRecord's dates show only an item's *initial* recording to the SalesJournal (after that, a JobRecord is archived and will no longer accept recording of new events). Thus, if the item was first recorded as a paid job, that date will, indeed, be just as useful as the same date found in the FundsJournal. If it was first recorded as a billed job, however, you'll have no way of telling from the JobRecord if and when the job was finally paid, and therefore moved (if at all) out of your A/R pile and into your archived invoice stack.

¹²⁹The same principle helps us locate even invoices in our AccountsReceivable stack (i.e., items that, if you checked the SalesJournal, would show a Paycode "2" entry because they were done but billed, but still no Paycode "3" because they have not been paid). Seeing this situation in the SalesJournal, we'd know first that our A/R stack was the place to go, rather than our invoice archive. On top of that, we could also be assisted in finding the item there quickly, if we've simply followed the same procedure there as suggested for the invoice archive. Specifically, add items into your A/R pile in the sequence of SalesJournal entry date, and keep them so-sequenced. Then, when you find the date of the Paycode "2" entry in your SalesJournal (using the SalesView form), you can again locate it rapidly simply by using the sequence of dates (albeit in this case the date should be stamped in a different location than when you're locating an item in your physical invoice archive, see page 188). For our A/R pile, we have specific sections for several of our largest clients, with each invoice pertaining to that client stacked in

One suggestion in regard to this invoice stack. Have strict rules about removing invoices from it. Don't let anyone ever remove such an invoice from your office, or even take it out of the stack for more than a few moments of quick examination, or to make a photocopy. And be religious about being sure it gets immediately back to its proper place. Otherwise, your only paper document having the customer's signature, and all the technician's handwritten documentation on the job, will be lost. You never know when it might be needed.

iv. The Reports Form.

While the Reports form has functions broader than merely reporting on sales (it can also produce Commission reports, Wage reports, and Aging of Accounts Receivable, for example), it bears mentioning here, for the sake of contextual discussion, that this is the specific form you'll use (press F11, or click on the corresponding MainMenu command button) when wanting to compile reports on sales. As you'll see from the form, you can do this on a daily, weekly, monthly or yearly basis (or as often as you wish, and covering whatever period you wish). You can print reports to either your screen or printer, and applicable to all sales, or only a particular department if wanted (see following). The reports include a plethora of data, along with useful ratios and other figures. There should be everything you need for thorough analysis, and to provide the particular figures that are needed for entry into your accounting software (see page 302), for compiling legally-required sales tax reports, and so on. ¹³⁰

v. Separating Sales Into Multiple Departments.

Some of our clients have wanted to separately track sales between different parts of their company's operation. One client, for example, has a division doing appliance service, another HVAC work, still another plumbing, and one involving over-the-counter parts sales. To know the separate profitability for each operation, they needed to know the specific sales of each. Thus, we have added a feature allowing this. It is purely optional: if you want to use it, you must enable the option from the 'Network' section of the Settings form (press Ctrl-F1 or use the MainMenu to access this form).

When enabled, you'll find that as you go to create a job from the Create Job/Sale form (see page 72), ServiceDesk will at such time require that you specify the department to which the job belongs. That assignment will then stay with the job, and never needs to be re-specified in any other context.

For the purpose of making this specification, there's a list box that will appear in the Create Job/Sale form (it appears only if you've enabled this feature from the Settings form). Initially, this box will be empty. To place names for each of your departments into the list, simply type in each name, and press Enter. You'll be required to use your password (this is to prevent a mere operator from altering the list), then each name will be part of the list from then on (unless deleted by you, which also requires your password).

Thus, after you've entered each of the department names into the list box, they'll be there from then on—for your operators to select the appropriate department as they create each job.

At present, any job's particular department-assignment is operationally substantive in only one context (though the assigned department will display from the JobRecord, and can be changed from there). When you produce a Sales report from the Reports form, if you have the Departmentalize feature enabled, you'll see a list box

the indicated order. Receivables for all remaining customers go into a remaining slot, also stacked in such sequence. Thus, after getting the date of our billed-but-not-yet-paid recording from the SalesView form, we go to the appropriate section of our A/R stack, and quickly locate the item within that section by using the found date.

¹³⁰ If you have to make different reports to varying municipalities, counties or states over which your territory spans, you have what we call the "Bad Sales Tax situation, a solution for which is discussed at page 326.

showing each of your departments. If you want your report to reflect sales only in a particular department, select that department (otherwise, the report will show all of your sales). Or you can select an option to print a departmental summary, that will tally the principle total figures for each of your departments. It's that simple.

Eventually, as we expand ServiceDesk to track specific job costing (or as we have particular requests), we'll likely expand the functions from which a job's department-assignment is referenced. For now, it's only in Sales reports where the matter is relevant.

B. Managing Accounts Receivable

There is still another area where many small businesses lose shocking sums of money. It's the matter of managing accounts receivable. We need, quite obviously, to police every item of money owed to us, to assure it's promptly paid, or that due and proper notice is sent when it becomes past due. But in most setups, this requires dedicated, ongoing, seemingly constant attention—and with the press of completing new work so demanding, it's common for office personnel to let this need slip. The best solution, obviously, is to make managing accounts receivable into a matter that involves only minimal labor, rather than much. As you might expect, this is precisely the solution ServiceDesk provides.

The primary file that's used to deal with Accounts Receivable is called, not surprisingly, the 'Accounts Receivable' file. As mentioned elsewhere, for every billed sale that you record via the SalesEnter system (see page 165), ServiceDesk creates a unique record, within this file, reflecting the sale and its expected payment. Thus, the file becomes a simple set of records reflecting every past sale on which payment is still expected.

ServiceDesk provides two forms for working with this file.

First is the *Accounts Receivable* form, which individually displays all the data from each record. It further provides search capabilities between records, various housekeeping and editing functions, and many other capabilities. To access it, press **F3** or click on the corresponding MainMenu command button. You'll see that most of its functions are quite self-explanatory.

Accounts Receivable

Hamilton, William
1046 Middlefield Road
Berkley, Ca 94708

Mr. Hamilton **N** Attn to: Qty Ltrrs sent **2**

Tech InvNmbr InvDate Rcrd #
CB **72308** **12/27/01** **103**

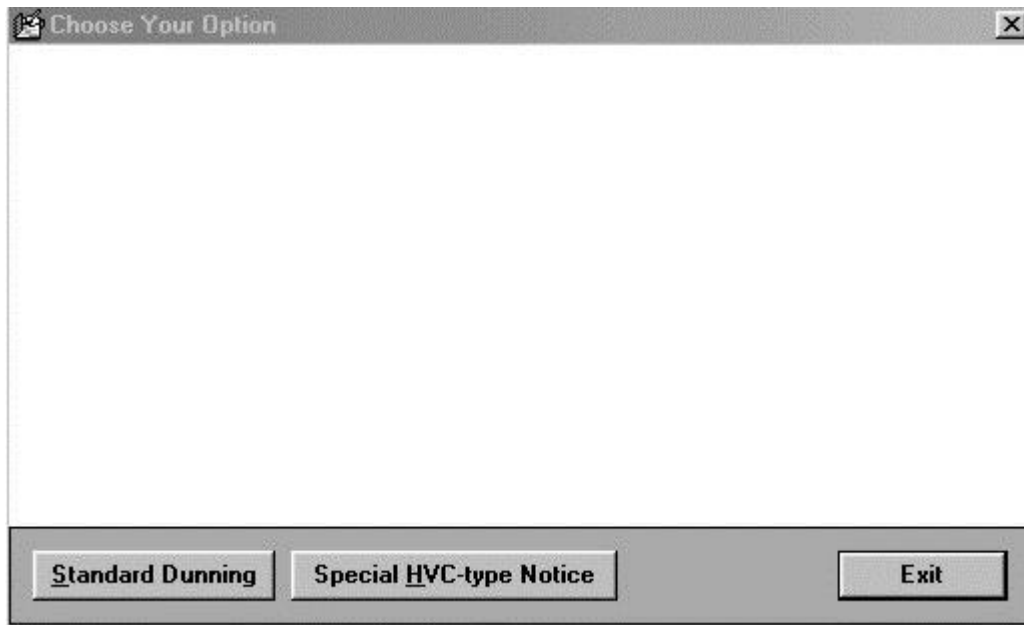
Write-off as bad debt

0 MrchSld
0 PartsSld
45 SrvcCall
0 Labor

45 Total
0 PdToDt

Name search **Top**
Invoice search **Up**
Purge Chaff **Exit** **Down** **Bottom**

The second form (accessed by pressing **Ctrl-F3**) addresses more of an operational need, for aside from keeping our AccountsReceivable records, reviewing and/or editing them as needed, and so on, our real concern is to be sure each is promptly paid—or that appropriate reminders are sent (in the form of dunning letters or statements) if they are not. For the function of sending such reminders, we use the *Dunning* form.



In regard to billing reminders (whether you prefer sending them in *letter* format or as more typical *statements*), you should note that here is another area where ServiceDesk discriminates between companies you've designated as 'HighVolume' types (see page 60) and all others. The reason is because, presumably, your HighVolume-type clients will probably owe on many receivables at one time, while regular clients will probably owe on just one (or certainly not more than a few). In addition, the preferred format of treatment is likely to be a little different for one type than for another.

Regardless, upon viewing the Dunning form (again, press **Ctrl-F3**), you'll see that your first option is to specify whether you want to do 'Standard Dunning' or 'Special HVC-type Notices'. The methodology for either procedure is almost self-explanatory, but we'll overview here regardless.

i. Standard Billing Reminders

We should start by clarifying a matter of terminology, and connected practice. Some companies are accustomed to sending out—as a way of reminding customers to pay what's owed—what may be thought of, more or less, as standard "statements". Others have used more of a letter-type format, with each letter explaining in more of a conversational manner what's owed (with these typically being referred to as "dunning letters"). As another difference, some companies send statements right up front, without waiting for items to become past due. Others just mail a copy of the invoice for initial billing, and don't send statements (or, of course, dunning letters either) until after that initial request becomes past due. You can do it any way you wish within ServiceDesk, though typically we think it's easiest to use the invoice-copy method (or, actually, one part of a multi-part form) for initial billing, and send a statement or dunning letter only later, when and if the item remains unpaid. Regardless, whether it's statement format or dunning letter, we'll refer to both here, generically, as "billing reminders."

With that said, you should begin by understanding that, at this time, ServiceDesk is not configured to produce *finished* billing reminders for your normal (i.e., non-HighVolume) clients. Instead, the idea is for it to

produce the raw data that can then be fed, typically, into the mail-merge (otherwise known as “form letter”) feature of a word processing program, such as Microsoft’s Word, for example, which will then take the data and place it into whatever finished context you may prefer. The reason for this design, obviously, is that it allows you nearly infinite flexibility in formulating your billing reminders in whatever manner you most prefer (in fact, you could even use contexts other than word processing for the purpose, such as Microsoft’s Excel spreadsheet program, for example).

If you have not previously been involved in creating merge documents, we’ll explain a little here. This is what big companies use to make it look like you’ve received a personal letter, when in fact your name and other information is just inserted at the right places within the text of a letter, in the address area on an envelope, and so on. The text might even vary, from place to place, depending on particularities in your own data. The reason they’re called “merge” documents is because of how it works. Essentially, there’s two documents that are *merged* together to make the set of output documents that’s personalized for each of maybe hundreds or thousands of individuals. One document (generally called the “Master” document), essentially contains the overall instruction set and common text. The other (generally called the “Data” document) contains the particular data for each of many individuals.

How it works is like this. The master document has the general text along with places, here and there, where essentially the internal instruction says “put the NAME from each individual here,” or “put the FIRST LINE OF THE ADDRESS here,” and so on. The data document, by contrast, has a paragraph of text items for each individual, with separate text items within that paragraph, each containing a particular element of information that’s relevant and available for insertion to the master document. When it’s all setup just right, the word processing program looks at the master document, then at the first paragraph of text from the data document, and inserts each element of text from that paragraph into appropriately-instructed places from the master document, sending the “merged” output to a new document, which you can then print. It then does the same from the next data paragraph (as pertaining to the next individual represented there), and so on through the entire set.

If you’ve not previously setup a word-processing program for this kind of feat, the task can at first seem daunting. That’s why we’ve set it up so you don’t have to. Instead, we’ve created a Word-based master document for you, that will produce very nice billing reminders.¹³¹ It’s designed to produce them in a format that’s something of a hybrid between a typical statement and dunning letter, hopefully carrying most of the advantages from each. It’s designed to make the letters vary in tone, urgency (and threat level), depending on how many previous reminders have been sent. Though you may need a magnifying glass to make them out, the following three items illustrate the kind of output that the provided Word master document will produce:

¹³¹ Actually, we’ve provided two Master documents, one to create the billing reminders themselves, and one to print addresses on the envelopes that you’ll need to mail them. The documents are entitled **DunningLetter.Doc** and **DunningEnvelope.Doc**. You should find them installed within the **c:\sd** folder of your computer, or in the **SpplmFis** folder on your installation CD. Load either into Word to view, use and/or edit.

<p>March 3, 2002</p> <p>Leanda Castro 28485 Barents Laguna Hills, Ca 92653</p> <p>Dear Mrs. Castro:</p> <p>This is just a friendly reminder. We're still waiting for payment from you on the following:</p> <table border="1"> <tr> <td>2/2/02</td> <td>73072</td> <td>\$45.00</td> </tr> </table> <p>As you may note, this item is now some 29 days old. If you would kindly forward this amount with due haste, I'd appreciate it immensely. Or, if there is any problem preventing your payment, please let us know.</p> <p>Sincerely,</p> <p>Name of Sending Person</p>	2/2/02	73072	\$45.00	<p>March 3, 2002</p> <p>Dan Streech 31721 Paseo Terraza San Juan Capistrano, CA 92675</p> <p>Dear Mr. Streech:</p> <p>This is another reminder. We're still waiting for payment from you on the following:</p> <table border="1"> <tr> <td>12/27/01</td> <td>72570</td> <td>\$108.28</td> </tr> </table> <p>As you may note, this item is now some 66 days old. At this point, I must insist that you forward this amount immediately—or, if there is some problem preventing your payment, that you let us know at once.</p> <p>Sincerely,</p> <p>Name of Sending Person</p>	12/27/01	72570	\$108.28	<p>March 3, 2002</p> <p>Gaye Bitcher 32327 Del Obispo St San Juan Capistrano, CA 92675</p> <p>Dear Mrs. Bitcher:</p> <p>This is another reminder. We're still waiting for payment from you on the following:</p> <table border="1"> <tr> <td>11/13/01</td> <td>71958</td> <td>\$30.00</td> </tr> <tr> <td>11/28/01</td> <td>72052</td> <td>\$1160.52</td> </tr> <tr> <td>12/20/01</td> <td>72303</td> <td>\$85.00</td> </tr> </table> <p>As you may note, the oldest of these is now some 104 days old. Together, the 3 items total \$1,335.52. At this point, I must insist that you forward this amount immediately—or, if there is some problem preventing your payment, that you let us know at once.</p> <p>In fact, this is the <u>last warning</u> I can give you. If the amount is not paid within the next week, I am under obligation to follow internal policy, and turn the matter over to collections.</p> <p>Please, do not let it come to this.</p> <p>Sincerely,</p> <p>Name of Sending Person</p>	11/13/01	71958	\$30.00	11/28/01	72052	\$1160.52	12/20/01	72303	\$85.00
2/2/02	73072	\$45.00															
12/27/01	72570	\$108.28															
11/13/01	71958	\$30.00															
11/28/01	72052	\$1160.52															
12/20/01	72303	\$85.00															

Please note that you'd want to print these on paper with your own letterhead, and at the bottom (from within the *Master* document, of course) you'd need to replace the text, where it says "Name of Sending Person" with a real person's name from your company. Please also note that you could change any of the other text (from the Master document) in any manner you might wish, though much of it is *hidden* and conditioned on particular circumstances (one receivable item versus more, how many previous reminders, etc.). Figuring out how to make the whole thing work when dealing with the hidden, instructional text is, again, not easy.

Of course, if you're up to the challenge, you may produce any kind of master document you wish (it is, in general, a *one-time* task, for typically you'll be wanting to use the same master document over and over again, month after month, year after year). We've provided one for you (one that produces the kind of reminders shown), simply to make it easy for those who don't wish to struggle with producing your own.

Regardless, once you have your preferred master document in place (again, for simplicity we recommend using ours), the task of producing your finished billing reminders, on a monthly or more often basis, is a very easy and simple one. It's done in three simple steps.

1. Use ServiceDesk's Dunning form to create the Data document.
2. Load the provided (or otherwise appropriately prepared) Master document into Microsoft Word (or other context if you prefer).
3. Load your printer with appropriate stationary, then select the appropriate *Merge* command from within Word (or other context if you've preferred).

In result of this simple sequence (should typically require less than 3 minutes), you can happily watch as a succession of perfect billing reminders stream from your printer—in whatever quantity as are at the time needed. Follow by loading a master document for *envelopes* (which we've also provided for you) into Word, and then watch a stream of perfectly-addressed envelopes stream from your printer.

In terms of the first step in the sequence (making the Data document), we'll provide a bit of guidance. Again, this is one of two processes for which the Dunning form exists.

Accounts to Send Dunning Letters to [pg 1 of 2]

1	68780	U-line Corp.	02/27/01	\$255.63	6	Y
2	68622	Asko Inc.	03/21/01	\$419.00	5	Y
3	69782	Karen Martin	04/27/01	\$45.00	5	Y
4	69297	Stephen Kimball	05/14/01	\$60.00	5	Y
5	70395	Mike Castillo	06/29/01	\$45.00	5	Y
6	70342	U-line Corp.	07/11/01	\$85.00	3	Y
7	71203	Del Ferguson	08/30/01	\$65.54	4	Y
8	71347	American Home Shield	09/13/01	\$140.00	4	N
9	71034	Kevin Mathews	09/14/01	\$111.75	4	Y
10	71330	U-line Corp.	09/17/01	\$85.00	4	Y
11	70823	Skin Therapy Day Spa	09/18/01	\$116.42	4	Y
12	71060	Dennis Demetre	09/19/01	\$425.00	4	Y
13	71381	Pedros Tacos	09/20/01	\$279.88	4	Y
14	70587	Asko Inc.	10/15/01	\$247.00	3	Y
15	70770	Leanda Castro	10/17/01	\$45.00	3	Y

Create File Special HVC-type Notice Exit

To use it for this purpose, begin by selecting its 'Standard Dunning' option, at which time you'll see a list of all non-HighVolume accounts that date from the preceding month or earlier. Basically, your task is to peruse the list (press **PgUp** or **PgDn** on your keyboard to change the portion of list in view), determining whether each item is one you want included in the data document it's about to produce. You'll notice a number toward the right end of each item. This indicates how many times you've sent reminders on it in the past. And at the very right end, there's a 'Y' or 'N'. Use this to indicate whether you want that item included in the data document the system is about to produce. You can toggle any item between 'Y' or 'N' status by clicking on it. The reason for this option is because you may have clients who, for one reason or another, you simply don't want to send reminders to.

Upon assuring that only those who you really wish to remind are so designated in the list, the next task is to click on the form's 'Create File' button. In response, the system will ask you to designate a name and location for the document it's about to create. Select the name and location that will be expected by your Master document.¹³² At this point, ServiceDesk will immediately create the Data document for you. Then, it's on to steps 2 and 3 (i.e., run Word, load the Master document that's wanted, and select the applicable merge command from therein). In result (and as stated), you should see a string of perfect documents streaming from your printer, ready for immediate mailing.

In sum (and particularly if you use the Word-based Master documents that we've provided), the process of regularly sending out billing reminders should be very, very easy. Of course, even then it requires some discipline to be sure you get around to actually running the process, at least on a monthly (or more frequent) basis. Be sure you exercise this discipline. You'll be rewarded with far fewer losses in unpaid accounts—and typically faster payments to boot.

ii. Special HighVolumeClient Notices

Ordinarily (and hopefully, at least) there's much less trouble collecting from the big institutional type of clients that you will have designated as 'HighVolume' types. Most pay reliably, within some predictable time frame, on almost all invoices. However, it still happens occasionally that a few invoices somehow fall through the cracks.

¹³²The provided Master documents are configured to find the needed Data document as **c:\sdl\DunningData.Doc**. If using the provided Masters as presently configured, you'll need to be sure that's the exact file path and name you specify when creating your Data document via ServiceDesk's Dunning form (as instructed in the text).

Some system is needed to regularly remind this type of client of the particular invoices which have so fallen. In this case, a somewhat different format seems preferable, and probably less formality is needed. For such reason, in this context we've designed it so that ServiceDesk will generate the reminders for you directly.

To use this feature, begin again by loading the 'Dunning' form (press Ctrl-F3), then select its 'Special HVC Notice' option. Immediately you'll see a list of each company you've designated (from the 'QuickEntries' form) as a 'HighVolume' type company (see page 60). Click on any company you're interested in, and immediately you'll see a report indicating what the receivable status is from that company (i.e., how many items and of what value over 30 days, over 60 days, etc.). If you wish to generate a reminder notice to this company, simply indicate when prompted. As you'll notice, you can specify any period of tardiness you wish to include in the notice.

Select a HighVolumeCompany (pg 1 of 1)

AMERICAN HOME SHIELD
FIRST AMERICAN
CHICAGO HOME WARRENTY
CROSS COUNTRY HOME SERVICES
OLD RE
BUYER
FIDELITY
EDISON
U-LINE
UNIVER
AON IN

ServiceDesk

Enter the cutoff (in terms of days since billed) for items you want included in this notice.

OK
Cancel

Standard Dunning Create Letter Exit

In result of the sequence, ServiceDesk will independently generate a nice little summary for the company involved, which you may then either mail or fax.

As you can see, we've made the process for sending these reminders *very* easy (easier than for normal billing reminders, for here there's no need to go through a merge-document type of procedure form a separate word-processing program). Still, if you don't have the discipline to go through it on a regular basis, it will do you little good. To stay right on top of things, you should create and send these reminder notices on at least a monthly basis. Please see that you do.

iii. Other Receivable Concerns

In case you're wondering how items are taken out of AccountsReceivable status upon being paid, we'll simply mention here that it's done, using a utility provided for the purpose, within the FundsForm (see page 157). You'll notice, in this regard, there's an item of information, displayed in the AccountsReceivable form and pertaining to each record, that indicates how much has been received on the sale in question. Whenever funds are checked in via the process just mentioned, this figure is updated by ServiceDesk for you. Of course, there may be occasions when you need to adjust the value manually. If so, it's easy to do so directly from the AccountsReceivable form (for the sake of security, this does require use of the owner/manager password).

As various AccountsReceivable records are in fact paid off (and their PaidToDate figures consequently indicate an amount equal to the sale), their records obviously don't need to exist within the file any longer (they are, after all, no longer AccountsReceivable). Thus again, we have a place in ServiceDesk where there's need, to run a routine with some regularity, that purges no longer needed records from the file. This is another of the events that will be done for you, on a nightly basis, if you have the Auto-Archive feature turned on (see page 209). Otherwise, on at least a monthly basis, you should click within the AccountsReceivable form on the button labeled '*Clean-out completed items*'. This will run a routine that removes all records from the file which have a PaidToDate amount equal to (or exceeding) the total sale. If this routine is not at least occasionally run, your AccountsReceivable file will grow ever larger, becoming filled over time mostly with items that have been paid and which need no longer to be there. Eventually, ServiceDesk will note the excessive size and alert you to run the routine.¹³³

Unfortunately, it's a sad reality that the path to retirement for some receivables is not by virtue of having been paid. Sometimes you'll find an item is simply uncollectible, or at least that further efforts to collect are not worthwhile. In such cases, you need a formalized method to "write off" the debt (i.e., to remove its item from the AccountsReceivable file, record the loss, etc.). This function is also addressed from the AccountsReceivable form. Simply bring up the particular record you're interested in, then click on the command button labeled '*Write Off as Bad Debt*'. In response, the form will fictitiously change the item's 'PaidToDate' value to equal the sale amount (this sets up the record to be purged the next time you run the 'Purge Chaff' routine), and it will simultaneously make a Paycode "4" entry into the SalesJournal (see page 169), reflecting the loss.¹³⁴ That entry (along with any similar ones) will then be the basis, when you later compile a SalesReport, of showing what you lost in bad debt during the period (as is necessary for entry into your accounting program).

iv. The Applications Journal

In introducing this section, we mentioned that the *primary* file for managing the various accounts on which you expect payment is the AccountsReceivable file (reviewed and managed via the AccountsReceivable form). This discussion would not be complete, however, if we did not mention another file and its associated form.

It so happens that on some receivables you'll receive a series of payments before the amount is paid in full (thus, you've got multiple payments on one bill). On others, you'll receive a just one check that applies to several different bills. From the accounts receivable file, the system keeps track only of the total that's been paid, at any given time, on any given receivable. There's no history there of how the payment came in (i.e., via what checks, on what date, and applied in what amount, to which receivables?). Sometimes, this latter information can be very useful (as when, for example, a client claims they made some particular payment against a certain claim, and you need to go back and see if you actually received the check, and if so how you actually applied it).

¹³³You might notice that in most other files where we run a routine to remove records that have completed their current purpose, the routine moves those records into some kind of archive, where they're still accessible should they be needed. With the 'Purge Chaff' routine in the AccountsReceivable form, in contrast, we essentially just *dump* the old records. The reason is because we've simply not seen much purpose in retaining a job's A/R record after it's finally paid.

¹³⁴Please note that in addition to this fundamental work, ServiceDesk will also invite you to create a *Special Situations Advisory*, thus "Red-Flagging" the involved customer as someone to watch out for in the future (see page 229). The other context that involves write-offs, by the way, is in the Funds form (see page 177). The difference, obviously, is that here you're dealing with money, owed to you, that was never even ostensibly paid by the customer. In that context, by contrast, you're dealing with funds that someone supposedly gave you, but which ultimately proved worthless.

Date	Payer's Name	ChkNmbr	ChkAmnt	InvNmbr	BalDue	AppldFrmChk	ActlCrtd	To/From-EDA
01/03/02	AHS	1696843	\$2,032.17	71941	\$72.00	\$72.00	\$72.00	\$0.00
01/03/02	AHS	1696843	\$2,032.17	71918	\$35.00	\$35.00	\$35.00	\$0.00
01/03/02	AHS	1696843	\$2,032.17	71837	\$245.76	\$245.76	\$245.76	\$0.00
01/03/02	AHS	1696843	\$2,032.17	71827	\$77.19	\$77.19	\$77.19	\$0.00
01/03/02	AHS	1696843	\$2,032.17	71551	\$253.73	\$253.73	\$253.73	\$0.00
01/03/02	AHS	1696843	\$2,032.17	71507	\$490.63	\$490.63	\$490.63	\$0.00
01/03/02	AHS	1696843	\$2,032.17	72010	\$68.13	\$68.13	\$68.13	\$0.00
01/04/02	FNW	101891	\$383.00	71899	\$463.00	\$383.00	\$463.00	(\$80.00)
01/04/02	FAM	517888	\$275.74	71945	\$275.74	\$275.74	\$275.74	\$0.00
01/04/02	DVORAK	1469	\$45.00	72329	\$45.00	\$45.00	\$45.00	\$0.00
01/07/02	SCHUMACH	4540	\$131.67	72432	\$131.67	\$131.67	\$131.67	\$0.00
01/07/02	FNW	201283	\$85.00	72135	\$40.00	\$40.00	\$40.00	\$0.00
01/07/02	FNW	201283	\$85.00	72163	\$45.00	\$45.00	\$45.00	\$0.00
01/09/02	JARVIS	15849	\$45.00	72365	\$45.00	\$45.00	\$45.00	\$0.00
01/14/02	AHS	1700952	\$354.69	72126	\$72.71	\$72.71	\$72.71	\$0.00
01/14/02	AHS	1700952	\$354.69	71329	\$281.98	\$281.98	\$281.98	\$0.00
01/16/02	KIPPEN	6785	\$136.44	70614	\$136.44	\$136.44	\$136.44	\$0.00
01/18/02	RANCHO M	6306	\$45.00	72304	\$45.00	\$45.00	\$45.00	\$0.00
01/21/02	FNW	202672	\$375.38	72222	\$121.29	\$121.29	\$121.29	\$0.00
01/21/02	FNW	202672	\$375.38	72164	\$254.09	\$254.09	\$254.09	\$0.00
01/21/02	FAM	521766	\$125.10	71284	\$0.00	(\$45.00)	\$0.00	(\$45.00)
01/21/02	FAM	521766	\$125.10	71924	\$170.10	\$170.10	\$170.10	\$0.00
01/22/02	VAUPEL P	5026	\$45.00	72250	\$45.00	\$45.00	\$45.00	\$0.00
01/22/02	AHS	1703153	\$983.88	72297	\$39.76	\$39.76	\$39.76	\$0.00
01/22/02	AHS	1703153	\$983.88	72232	\$35.00	\$35.00	\$35.00	\$0.00
01/22/02	AHS	1703153	\$983.88	72229	\$47.44	\$47.44	\$47.44	\$0.00
01/22/02	AHS	1703153	\$983.88	72215	\$12.00	\$12.00	\$12.00	\$0.00
01/22/02	AHS	1703153	\$983.88	72198	\$258.38	\$258.38	\$258.38	\$0.00
01/22/02	AHS	1703153	\$983.88	72186	\$24.00	\$24.00	\$24.00	\$0.00
01/22/02	AHS	1703153	\$983.88	72181	\$56.61	\$56.61	\$56.61	\$0.00
01/22/02	AHS	1703153	\$983.88	72154	\$87.24	\$87.24	\$87.24	\$0.00
01/22/02	AHS	1703153	\$983.88	72151	\$73.99	\$73.99	\$73.99	\$0.00
01/22/02	AHS	1703153	\$983.88	72119	\$120.76	\$120.76	\$120.76	\$0.00
01/22/02	AHS	1703153	\$983.88	72105	\$91.08	\$126.08	\$91.08	\$35.00
01/22/02	AHS	1703153	\$983.88	71718	\$102.62	\$102.62	\$102.62	\$0.00
01/25/02	MANOOGIA	1823	\$35.71	72191	\$35.71	\$35.71	\$35.71	\$0.00
02/23/02	AHS	1709471	\$1,500.05	70275	\$0.00	\$111.00	\$0.00	\$111.00
02/23/02	AHS	1709471	\$1,500.05	70575	\$497.73	\$468.11	\$468.11	\$0.00
02/23/02	AHS	1709471	\$1,500.05	72105	\$0.00	(\$35.00)	\$0.00	(\$35.00)
02/23/02	AHS	1709471	\$1,500.05	72107	\$49.96	\$49.96	\$49.96	\$0.00

Search

showEDA

Exit

The ApplicationsJournal (a separate file) is intended for this purpose. In one sense, you might say it's a bridge between the Funds form (where any payment on a receivable should be checked in, see page 157) and journal, and the AccountsReceivable context. Basically, each time you check in payment on a receivable, the system makes entries into this journal, for you, regarding how the fund was applied (i.e., how much on which receivables). Thus, you have a permanent record to refer back to, when needed.

To access and review this information, use ApplicationsJournal form, accessed by pressing **Alt-F9**. There are self-explanatory search and review features, but aside from that, there's essentially no work to be done within this form, for its purpose is informational only.

C. A Note Regarding Organization

The next chapter has two sections. Arguably, the first should instead be fitted as the *third* section in this chapter (indeed, right here in this very spot). The section is about *submitting electronic claims*, after all, which directly involves *post-completion management* (the precise focus of this chapter, so you'd think it should go here). But as it happens, claims submission is done within the *Finished-Forms interface*, which is the direct topic of the next chapter. For such reason, we place the discussion there.

Chapter 10

THE FINISHED-FORMS INTERFACE

In the earliest ServiceDesk incarnations (pre-2000), there was no Finished-Forms interface. In regard to invoice/ticket options, ServiceDesk had but one. It was for what we now call our “*up-front*” ticket. Machinery for that option is distinguished by the fact that final text as printed can never include more than *job-initiating* information (the form itself can have spaces for many other information items, but ServiceDesk cannot sensibly populate them) In other words, the up-front ticket machinery can manage the names of parties to the job, applicable addresses and telephone numbers, description of machine to be serviced, symptom, etc. But it cannot manage a description of work performed, materials used, and what the charges are. Mechanisms as associated with the up-front ticket simply do not include such facility. They were not designed for it.

Nor was any such facility *needed* by the first ServiceDesk users. Those earliest clients did not do OEM warranty work (hence no need for filing warranty claims), and did not do point-of-sale (POS) operations (hence no need for a counter ticket that includes a listing of items sold, prices and total, etc.). The simple up-front ticket sufficed for all needs. Printed for the techs before they went out each day, all further details (concerning work performed, materials used and fees) were *hand-written* by the techs on location at each job. In a nutshell, all ServiceDesk tickets in that era ended up as hybrids: partly machine-created, and partly hand-written.

The above is explained to contrast with this chapter’s topic. The Finished-Forms interface (Alt-F4) was created to provide a place where you can view and edit every detail about a ticket on-screen, then print, email or electronically transmit. By “every detail,” we mean not just the same *job-initiating* information as is available with the up-front ticket, but also details that come only with job-completion, such as description of work done, listing of materials used, and itemization of charges. Hence the title: Finished-Forms.

While the Finished-Forms interface is today so elaborate as to deserve an entire descriptive chapter (this one), it did not begin that way. Its first incarnation featured solely an on-screen/editable representation of the NARDA¹³⁵ form. Prior to this, warranty servicers were hand-writing onto actual paper NARDAs, which were in turn postal-mailed to the manufacturer for submission of each warranty claim. This hand-filling-in process was ultra-laborious, so we were asked to create a mechanism whereby ServiceDesk would instead *machine-print* applicable text into the NARDA’s spaces.

To fulfill the request, we made an on-screen representation of the NARDA, with editable boxes in each place where text goes on the paper NARDA equivalent. We further made mechanisms whereby information, as applicable to any particular job as present within ServiceDesk could be made to auto-fill to such boxes. And, of course, we made mechanisms via which such text can output to a printer.

¹³⁵ NARDA stands for North American Retail Dealers Association. The form in question was developed by that organization to provide a uniform claim format. Prior to its creation, each manufacturer had its own unique form on which a servicing company had to submit its claims. This vastly complicated the claims process for any company that was doing work for multiple manufacturers. The NARDA form provided a uniform submission interface, and eventually was accepted by all manufacturers in the U.S. appliance industry.

That, essentially, was the birth of our interface, and we did not immediately envision it as having broader application. Very quickly, though, it was realized that the very same on-screen-editing-and-then-printing capability would be handy for ticket formats other than the NARDA. Thus, we soon added two alternate-form interfaces.

The “*Custom*” form interface was designed generally to mirror the up-front ticket in format, but to of course (and in contrast) include full completion details, with on-screen editing, etc. (this is a form that, whether pre-printed or otherwise, requires a background image). The “*Generic*” form interface was designed to be, well, more generic in character, and with a design that requires no advance background (hence it is inherently suited for printing to previously blank paper).

With these alternate forms, it became easy to produce a completed ticket that was perfectly machine-done in its entirety. Thus (and as an example), if you needed to produce a very nice ticket for after-the-job mailing to a customer (no messy handwriting, no grease on the paper, etc.), it was now very simple to do so. Plus, we added an option to *email* the image, as opposed to printing first and then postal-mailing.

Before long we encountered our first clients involved in significant point-of-sale (POS) operations. Usually, it was a service company or dealer that also had a “parts counter,” conducting over-the-counter parts sales. They wanted to know how to manage this in ServiceDesk, and we realized our Finished-Forms interface was the best answer. By and by, we elaborated on its features to make them more amenable to effective POS functions, and eventually added a fourth form (the *POS* form) specifically designed for a streamlined POS sequence.

Finally, after launching our SD-Mobile system with its own unique, in-field ticket, we realized there was occasional need for the office to interface directly with (and in editable format) the ticket a tech created in the field. Hence, we added the Finished-Form’s *Mobile* ticket.

This is the history, briefly stated, of how the Finished-Forms interface arrived at its state today. We sometimes provide such a historical overview because, simply, it’s often easier to understand the current structure when it’s placed into an accurate developmental context.

With the above done, we’ll now discuss the two major areas of operation, within the Finished-Forms interface, that significantly need elaboration.

A. Electronic Claims Transmission

To the best of our knowledge, Maytag was the first company to pioneer electronic claims. They designed a file-format into which each claim must be fit. Much like a paper claim has boxes to fill-in, their electronic file format had specified “fields.” The notion was you make a file (or, better yet, your software does it for you) that puts appropriate text into each field within the file. You can add as many claims to the file as wanted, then must “upload” the file to a designated location. Early on, Maytag designated a particular electronic bulletin board as the place where such files were uploaded to. Later, as websites proliferated (and as other entities began accepting electronic claims), the uploading function shifted to website interfaces as particular to each such entity (each had its own specified file-format, too).

To make the transition within our Finished-Forms interface, from merely *printing* the contents of our on-screen NARDA to instead *saving* it to a file (and within a particularly-specified format), was quite easy — in principle, at least. It’s simply involved picking a different place and format for the output. Everything else stayed the same. For such reason, our Finished-Forms interface conveniently became the perfect platform for launching

electronic claims. Aside from the actual functionality, we simply had to add another action button: instead of solely “Print,” we added a “Transmit” button.

In principle, the entire concept is that simple. When you’re ready to claim on any warranty job, just let the system auto-fill its info to the on-screen NARDA, check for any needed edits, then click on the “Transmit” button, and follow further prompts. Such further prompts will ask you to identify to whom the claim is going (has to know to format correctly for that entity), and for some will ask you to choose among optional methods.

In regard to this latter, Rossware pioneered an advanced form of electronic claims transmission. To us, it seemed a little stupid that you should have to first save claim information to a file, and then afterward still have to manually *upload* it. With some of the processors (in particular, ServiceBench and ServicePower), we worked out “direct” methods, via which ServiceDesk can directly hand-off each claim— thereby avoiding any need for you to upload a file after the fact. It’s much more efficient and modern. Nevertheless, the old methods are still optionally available even for those processors, and are the only methods available for others.

The above four paragraphs provide the entire conceptual framework as needed to enable your successful entry into the hyper-efficient world of managing your claims electronically, and as direct-integrated with your management software. In principle, such understanding will allow you to immediately escape the inefficiency and drudgery of manually filling-in warranty claims on-line. Essentially, ServiceDesk is going to be accomplishing the fill-in there for you — though by putting all the claim information in a packet that is then processed (for fill-in there) by the entity to whom it’s handed off.

Aside from such general principles, we’ll now discuss a few specifics.

i. Client Setup

If you do warranty work, each involved manufacturer is a client. You should create an appropriate template for each in your QET interface [see page 57]. This is important, because when ServiceDesk fills-in that on-screen NARDA for you (as applicable to any given job) you want it to be filled-in as optimally and perfectly as possible (it means less for you to fix via manual editing). If you have an underlying QET template appropriately setup, ServiceDesk will be able to grab (and appropriately insert to the NARDA) information as appropriate to the underlying manufacturer (things like default labor amount, your servicer account number with the manufacturer, etc.).

It’s also important to assure ServiceDesk knows how to connect from a particular JobRecord to a particular client’s QET. To assure this, you need to know the method ServiceDesk uses when making the attempt. Quite simply, it looks in the Customer-Name box (very top) of the JobRecord as being loaded to the NARDA. It extracts what we call the *first-phrase* of text, which is defined as any such text as begins at the front of the line and continues until reaching a group of two or more spaces. It takes that text and goes looking in your QETs for a match, either to a QET name or to the two-or-three-letter abbreviation as there specified for same. If it finds a match to either, it goes “Aha, I have a match,” and figures info from that template should be used to guide auto-fill (for such boxes where it’s appropriate) to the on-screen NARDA.

Based on the above, please realize that for optimal (i.e., labor- and frustration-minimized) usage, it’s important to assure you’ve setup QETs appropriately, and that each applicable JobRecord is appropriately configured to connect to same.

ii. Alternate Process Scenarios

There are several paths for getting into the Finished-Forms interface.

As prior alluded, you can open the form directly via its quick-key shortcut (Alt-F4; access is also available via mouse-click in the MainMenu). Once you're in the interface, there's a simple box where you can *type* any ServiceDesk invoice/ticket number, then load into any of the optional form types.

As one potential work scenario, then, it's possible you could have a stack of paper tickets that need claims. With the stack on your desk and near your left hand, go to the Finished-Forms interface, pick NARDA as your form type, then type the first ticket number, and hit Enter for load (or click on the *Load* button, if preferred). ServiceDesk then loads information into the form, you scan and make sure all is appropriate to the claim (editing for needed changes, if any), then click on Transmit, and pick the entity/method of conveyance. With that first claim done, flip over the first sheet on your desk and proceed to the next — working in significantly rapid fashion through the entire stack, to complete your claims. (Don't forget: for any claims simply saved to a file, eventual upload is needed too).

That's one potential approach, and would be particularly suited if you're still using paper, and if there's a person who's directly responsible for the claims process but not directly involved with other processes that might otherwise be integrated.

For an example of the latter, suppose you've decided to organize your processes such that an office person is doing PostVisitReports (using the Type-II interface) on behalf of techs based on paper tickets they've turned in from the prior day's work. Suppose the same person is also authorized to handle claims submission. In this case, the person would want to have the PVR form's "*Link automatically to post-completion tasks*" box checked [see page 116]. Upon finishing the PVR in connection with a warranty job that was itself complete, she'd then auto-link to the Finished-Forms interface. In this context, there'd be no need to provide it with a ticket number, since it already would have been passed as part of the linkage. As in the other described case, even so, she'd pick NARDA as the form type, ServiceDesk would auto-fill, she'd proceed with the claim, then it would take her right back to the PVR Type-II form for the next PVR item.

In the *first* context as described, your operator is doing claim, claim, claim, all from within the operative home of the Finished-Form interface. In the second, she's doing PVR-then-claim, PVR-then-claim, all from within the operative home of the PVR Type-II interface. Depending on your organization and circumstances, one may be more efficient, or the other. At any rate, there's no need to strictly pick: you can go back and forth, at will.

Still another context would involve looking directly at a JobRecord on which you know a claim needs to be made. You can pick that form's "*Print Options*" button, and from actual options as then presented pick FinishedForms, from there NARDA, then make the claim. The main point is, there are many options, and what's most efficient depends on the circumstances.

As a final example, suppose you've implemented SD-Mobile with your techs, and you've gone rather paperless, so as a rule are not getting paper back from them. One method to facilitate appropriate Post-Completion management, in this scenario (both sales entries and claims submissions), is by using the JobsPerusal form (shortcut is Shift-F7). There, pick the "*Completed*" category, and rotate through each job in such status, doing such post-completion management as the situation demands. In this scenario, there's no need to worry about PostVisitReports (already done by the tech via Mobile), but you'll need at least to make a SalesJournal entry, plus a claim if applicable. If the latter, choose "*Print Options*" (as an applicable JobRecord is selected) and proceed exactly as otherwise described above — except please also note you can integrate to the SalesJournal entry from that FinishedForms/claims interface (further described shortly).

iii. Details on the Imported-data-Auto-Fill Function, and the Re-Load Alternative

Whenever a job is first “loaded” into the Finished-Forms interface, there’s a ton of underlying work done by the system. Besides looking for information pertinent to a particular warranty client that should be filled in, for example (see discussion above regarding use of QE-Templates), it looks in your inventory-control system to find any parts as used from inventory that should be filled in. It looks in your parts-process system to find any parts as special-ordered that should be filled in. It looks in your funds-control system to find any monies received, as should be filled in. It looks in your PVRs to find any tech date and start/end-times that should be filled in. And it looks in any JobRecord-attached UIS to find model, serial and purchase date to fill in.

In short, it looks in all the myriad places that ServiceDesk has collected information, as relevant to the job as you’ve used its multifarious mechanisms to most conveniently manage all connected processes. It ties them all together via automated fill-in to whatever FinishedForm you pick (where it’s a warranty claim, of course, you’ll pick the NARDA). Among other things, please realize this means you are going to achieve maximum benefit when you are fully using all those other system according to their intended design. Anything less, and your benefit will be reduced.

For most warranty processors (and if you’ve done all the above appropriately), the auto-filled text will be all (or in some cases *nearly* all) that’s needed for successful configuration of your claim. Common exceptions involve such matters as an extra mileage or extra labor charge, or where a manufacturer requires provision of special codes. Regardless, the general idea is that a claims operator should review what’s presented in the on-screen NARDA, and (prior to picking *Transmit*) add any such edits as might be needed for successful configuration.

In regard to editing what’s presented, you’d probably not like it if you put a bit of work into refining the initial auto-fill, and such work was not saved. For this reason, FinishedForms mechanisms will automatically save whatever edit state you have created, in any FinishedForm and as connected to any particular underlying ticket. It does this behind the scenes, and without even your awareness. The fact will be evident only later, if you go back to the Finished-Forms interface, and with a ticket number specified on which there was a prior such save. In such a case, ServiceDesk will convert to bold any radio button as associated with a form type on which there was a prior save:

Type of form to use

☒ **Generic** ☐ Mobile ☐ PQS

☐ Custom ☐ **Narda**

Invoice to Load

73041

Based on Generic and Narda showing in **bold**, we know in advance there were prior edits within those form types on this ticket

If you pick to load such a form type (i.e., one on which there was a prior edit-save for the ticket in question), ServiceDesk will not immediately do the “search-within-all-its-data-elements-for-fill-in-to-the-form” function. Instead, it will give you the option as to whether you want that (aka “a *fresh import*”) versus a *re-load* of the form in the state you last edited it.¹³⁶ Please understand that if you pick re-load, that’s exactly what you’re going to get. In other words, even if ServiceDesk has newer information than existed at the time of that last edit (e.g., new parts special-ordered), it’s not going to fill them in on a re-load. If you pick a re-load of prior edits, that’s exactly what you’ll get (text in exactly the state you last left it). On the other hand, if you pick *fresh import* it’s going to ignore your prior edits and do an entirely fresh import. This is a place where there’s no “best of both worlds.” It’s one or the other.

¹³⁶ As a matter of fact, you can pick multiple different saves of any form type as connected with a single underlying ticket number (just append the number, in the form’s own particular invoice number box, with a hyphen then other character, such as 73036-A, for example). If there are such multiples, you’ll be given a list and allowed to pick which to re-load, assuming that such re-load is your preference.

iv. File-Saving Details as Relevant to Batch Uploads

When you're using the newer, direct-transmission-of-claims methods, there's not much you can do wrong in regard to direct hand-off of the claim. The first time you do such a process from any station, you'll be required to provide your credentials (ServiceDesk needs them to talk directly on your behalf to either ServiceBench or ServicePower), and from that point forward ServiceDesk assures accurate hand-off of the claim info as assembled in your NARDA.

Where you're using the older methods, though (sometimes called "batch upload"), the burden remains on you to upload a claims file after you've assembled claims into it. Such older methods are all that's available (at least as of September '11) for processors other than ServiceBench and ServicePower, so there's a good chance you'll still need to be using them. We want to advise you here on the easiest/safest method of managing such files.

Whenever in ServiceDesk you go to save a claim to a file (you'll have gotten there from the FinishedForms NARDA by clicking on the Transmit button, then picking a non-direct-method option), ServiceDesk will open the standard Windows *Open-File* dialog box. It will default to a particular location and filename, the latter being specific to each processor. While you're perfectly free to pick a different location and name, we suggest (for simplicity) you accept the default. When you make the very first claim as applicable to a particular processor, the file will not already exist. ServiceDesk will create it as you put that first claim into it. When you go to do the next claim for the same processor (and if you accept the default offering to the same file), ServiceDesk will tell you the file already exists, and ask if you want to add your present claim into the file (append), versus replacing what's there with your present claim (replace)?

Here is the strategy we recommend:

As you're making claims during the course of any given day, work to build multiple claims into each file as applicable to each batch-upload processor you're working with. Suppose in the course of a day you will make three Dacor claims, for example. With the first such claim of the day, we suggest starting with a fresh Dacor claims file (thus, if one already existed when making the first Dacor claim, you'd choose to *replace* what was already in that file). With each such subsequent Dacor claim, choose to add to (or *append*) what's already there. In this manner, by the end of your work for that day you will have accumulated all of the day's claim work, for Dacor, in that particular file, and now you can upload it. Same with your Fisher Paykel file, and so on. The next morning start fresh in each relevant file, building it throughout the day for the evening's upload, and so on.

In regard to the above, please note we have not suggested you uniquely name a file for each day's work. Though there are some companies that do this, by our thinking that's too much of an annoyance, and there's no real need. We prefer the concept of using the same file set (one for each processor involved with batch claims) over and over. If you happened to mistakenly replace a file prior to uploading (i.e., Tuesday morning you're making claims and choose in the first instance to replace, without realizing you'd forgotten to upload the afternoon prior), it does not mean you've lost significant work. You can easily re-load the NARDA's as involved with each claim in the file that was not uploaded, and add them back into the current day's work. Not a big deal.

By the way, when talking about uploading those batch files, you'll note we're giving you no instructions for the actual upload. It depends on who you're uploading to. With all, you must log into the website of the entity involved (using the unique login credentials as provided to your company by the entity), then use mechanisms they there provide to accomplish the upload. More specific than that, we can't tell you. That end of it is entirely out of Rosswire's orbit.

v. Reviewing for Post-Submission Claims Rejection, and How to Deduce Requisite Configuration Details

Just because a claim is ultimately provided to a processing entity (whether via direct transmission or via batch upload) — it doesn't mean the claim is going to be accepted. In fact, you can count on some quantity being rejected, and for a variety of reasons. Thus, even though you're claiming electronically, it's critical to assure you've implemented procedures and practices to assure notice to yourself of those claims that are rejected, and reaction sufficient to assure the rejection is dealt with and corrected. Success in this aspect alone (at least if you do a significant amount of warranty work) can easily mean the difference between high profitability versus bankrupting loss.

The procedure as needed to bring notice to yourself of rejections will vary by the processor, and you'll need to seek understanding of same from each such processor.

Regardless, as you're studying rejections it will become apparent, in some instances, that particular elements of text were not filled into the particular on-screen NARDA boxes as needed to get them into the particular manufacturer's boxes as needed for a successful claim. When such occurs, you're going to wonder, which box of the NARDA did that item need to be in, it it to get it into the manufacturer's box I wanted it to end up in?

In some cases these queries can be difficult to answer. It's because, in all cases, there are at least two *translations* taking place:

1. Whenever ServiceDesk transmits a claim or places it into a file, it places claim data into the particular *claims-transmission format* that's specified by the selected entity. Each entity has a unique such format. It consists, essentially, of labeled boxes (i.e., "fields"), that must be formatted according to a very precise set of specifications. With some processors the specification includes a fairly large set of fields; for some it's smaller. Regardless, ServiceDesk has to pick which boxes from its on-screen NARDA should be placed into which boxes of the processing entity's specified claims-transmission format. It endeavors to pick as logically as possible, based on prior experience and on the names the entity itself has attached to each such box.
2. Once the processing entity receives the claim (and in the format as self-specified for the purpose), it (or, rather, its software system) must do a translation on its end. Specifically, if it's an entity like ServiceBench or ServicePower that has multiple manufacturer clients, each client has their own claim template. The entity must translate from its own claims transmission format into the respective manufacturer's format (in other words, must decide which boxes from its own transmission format get stuck into which boxes of the manufacturer's format). In fact, even where it's a single entity such as Dacor (managing its own claims only, via its own internally-specified transmission format), a translation is still required. Believe it or not, there is not a one-to-one, item-by-item and within the same-box-titles equivalency between fields in Dacor's transmission format and boxes in its on-line claim submission/review form.

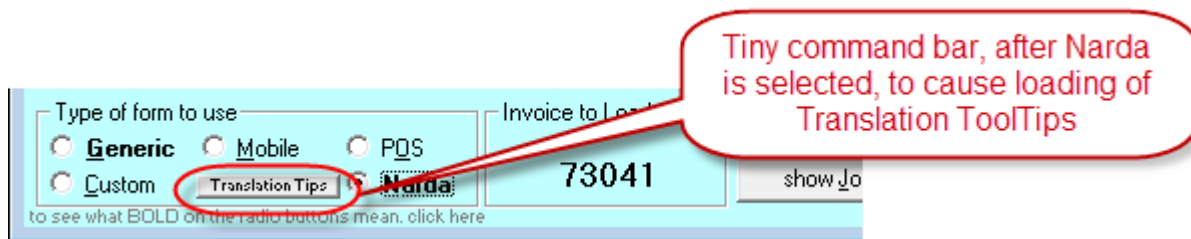
It's because of the above translation dynamics that, in some instances, you might find yourself in the conundrum of wondering: what box in Rossware's NARDA must "X" be in to show up in Box "Y" after the Rossware-assembled claim is handed off?

We have a conundrum too. It's that we don't know precisely how any entity makes the translation, from its own transmission format to any particular on-line claim review format. They don't share that with us. They give us specs for their transmission format only.

Given this, our best means of assisting you (for those rare instances where you face this conundrum) is by giving you an easy means to determine precisely what translation is taking place on our end (i.e., from on-screen NARDA to the processing-entity's claims transmission format). Our thinking is, you can see the names of transmission format boxes we're sticking stuff into, and logically deduce (or perhaps even by trial and error discover) which one the entity is in fact using for your needed purpose. If that fails, you can contact the entity, and ask them the particular name of the claims transmission box "X" needs to be in. Then, using our translation-exposure utility, you can directly see which of our NARDA boxes fills to that particular claims transmission field.

How does our "translation-exposure" utility work?

Quite simply, when our on-screen NARDA is loaded, a little command bar appears to the left of the NARDA-selection radio button.



When you click on that, you'll next be asked to pick the processing entity of interest. After you pick it, you get a little message that identifies any claims-transmission format boxes, as exist for that entity, that we're not sticking anything into at all. More importantly, each of the NARDA's boxes will become equipped with ToolTips (the little notes that pop up when you float your mousepointer over) that tell you the precise name, from the underlying entity's claims-transmission format, of the field contents of that box get stuck into.

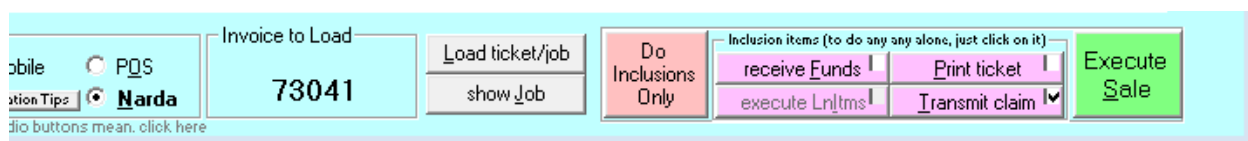
Thus, we fully expose the translation that happens on our end. If the need arises, you'll need to seek help from the processing entity for exposure of what's happening on theirs. Most typically, though, you'll find success easier to achieve. We're explaining these resources because they are ultimately available, if and when needed.

vi. Integrating with the Sales Entry Process

If it's time to make a warranty claim, it's very likely also time to make your entry to the SalesJournal. Certainly, you could segregate such entries (and for a whole stack of tickets) into their own separate batch processes, and depending on circumstances that *might* be the best strategy for you. For most operations, though, we think it's more likely you'll find it convenient to tie that entry right in with the claims process.

Your Finished-Forms interface has action buttons arranged in a fashion that makes it easy to integrate (tie-together) a variety of actions (more on this when discussing POS, in the next section). In particular regard to our present discussion, suppose you want each job's SalesJournal entry tied directly to making its electronic claim.

If you look at the button structure below, you can see there's a group of actions (violet-colored buttons) that can be included (or not) in the click to enter the Sale:



The specification (as to whether such actions will be included) is based on whether a little checkbox (in each such button's top-right-corner) is checked. Thus, if you wanted to tie the SalesJournal entry and claims submission directly together, you'd assure that the "Transmit Claim" button has its box checked (precisely as per above), and click on the Execute Sale button to accomplish both.

B. Point-of-Sale Operations

As described in this chapter's introduction, Rossware came a bit late into the POS world. As recently as September '11, in fact, we realized our POS mechanisms still needed significant upgrading (which, in fact, was done at such time). What will be described here is the state-of-the-art as was developed at such point in time.

In general, any of the FinishedForm types — except Mobile — could be used for POS functionality. Practically, however, it's impossible to imagine anyone would want to use the NARDA for the purpose. Most companies use either the POS form itself, or the Generic. Some use the two alternately, depending on circumstances. A few use the Custom.

Regardless of which form type is used, all (again, except Mobile) are designed to accommodate basic POS functions. In other words, you can pull items from inventory and sell them. You can create special-order requests. You can watch as the system self-totals items being sold, and automatically adds sales tax. You can collect items of money. You can make the SalesJournal entry. And, of course, you can print the ticket. Additionally, you can optionally tie/integrate the latter functions together. If you're a merchandise re-seller and use our SD-Dealer serialized inventory application, you can also integrate with it.

i. Alternate Approaches to Initiation of the POS Ticket

Prior to March '08, each and every POS transaction required initiation via exactly the same mechanisms as when creating a ServiceDesk JobRecord. In other words, you needed to put at least the customer's name into a Callsheet, do the Job/Sale transition to create a JobRecord, then (and on the basis of said JobRecord) go to the FinishedForm to do the actual transaction.

Around that time we had some new clients who were heavily into POS operations, and who rightly complained that such procedures were too cumbersome for situations where they simply wanted to do a rapid succession of simple sales, and did not even desire to track each purchaser's name. They requested a dedicated POS interface, one that would stay on the screen always, instantly ready to quickly conduct simple sales, and without simultaneous creation of a JobRecord (an instrument that is really designed, obviously, to manage performance of service).

This is when we created the POS form type, combined with a new mode of interaction within the Finished-Forms interface (we call it "Direct-POS"). Specifically when the POS form type is selected, the interactive mode changes to make it more amenable to the kind of abbreviated interaction our new clients wanted. It's a mode, specifically, that makes what is essentially a dedicated, POS-operations-only window, of a kind you can have "always-on" at any station that's dedicated to a part-sales counter. It's expressly designed for that circumstance, and particulars are described in the next sub-section.

The old method is still available, and remains preferred by some users (some find it optimum to switch back and forth, depending on circumstance). The old method, too, has been enhanced to make it a little more direct.

Specifically, if from a Callsheet you right-click on the Job/Sale button (as opposed to left-click) it will take you directly to the Finished-Forms interface, rather than down the standard “I’m creating a JobRecord” path (which it still does in the background; you’re just not stopped on the way for approval).¹³⁷

ii. Specifics When Using the Direct-POS Option

To make sure we’re clear, the Finished-Forms interface is placed into its dedicated, Direct-POS mode simply by bringing up the interface (Alt-F4), and picking POS as the form type. With that simple action, the interface is instantly posed for direct and abbreviated POS operations.

Among other differences (vis-à-vis other modes), please notice the box where otherwise you’d see a place for typing a target JobRecord’s InvoiceNumber. In this mode and instead, the same box invites you to type the ID of a Sales Person. This is, in fact, how any direct-POS transaction is initiated: with two keystrokes, by an operator, typing his two-letter abbreviation. By such means, the system simultaneously knows to begin a new transaction, and who is the operator conducting it.

Upon such initiation, you’ll see the POS form fills with beginning info, and places the cursor in the appropriate box for the operator to begin listing items being sold. At this point, your within-POS operations are very much as otherwise described in this chapter.

Underlying, though, is a major substantive difference.

With conventional POS-initiation (via entry to a Callsheet then Job/Sale-to-the-POS), a JobRecord is created for each transaction. With this method (and with one exception to be described), there is no JobRecord. There is just the POS ticket only. Rest assured, it can be searched for and reviewed later via its ticket number and specifically within the Finished-Forms interface window. However, that will be the only method. Unlike in the case of JobRecords, there will be no integration with ServiceDesk’s CstmrDbase index system (for as-you-type matches by name, address, and telephone, etc.).

The exception from absence of associated JobRecord occurs if you flag one or more of the line-items within your POS for ordering parts. In such a case, the system will demand creation of a JobRecord (and will likewise demand provision of at least the customer’s name, if not already provided) prior to proceeding with execution.

As another difference you will notice that when you first type in your two-letter abbreviation to initiate a sale, the POS form’s InvoiceNumber box auto-populates with the phrase “ToBePulled.” This signifies nothing is yet official. The system has not yet pulled an invoice number to assign. In fact, even as you begin filling-in boxes with items you’re intending to sell, nothing is recorded (and no invoice number is pulled) until you execute (by clicking on any of the operation-specific buttons). When you do, you’ll see that the “ToBePulled” text is replaced with an actual number. Simultaneously, the system saves a copy of your ticket, so there’s an instant and permanent record of the ticket associated with that invoice number. You could still abort the sale at this point, but the record will remain regardless.

In regard to the InvoiceNumber that’s pulled, you’ll notice that (unless parts are being ordered or the sale is being billed) the system makes it a negative number (i.e., puts a minus sign in front). This is so, internally,

¹³⁷ If you’ve activated ServiceDesk’s Departmentalization feature, we highly suggest you create a department called “Counter Sales.” Among other things, presence of a department of precisely that name will facilitate this right-click express-transition from Callsheet to the FinishedForms/POS interface. Specifically, when you do that right-click, the system will look to see if you have a department called “Counter Sales,” and if so will auto-assign the new JobRecord to said department. This avoids you needing to select, and allows the system to shuttle straight to the FinishedForms, with no stop along the way. If no such department exists, on the other hand (and if you have Departmentalization activated), it will have to make the stop.

ServiceDesk can distinguish the reference as one where there's no expectation for an accompanying JobRecord. To state it differently, the negative number denotes it as involving a Raw/Direct-POS, no-JobRecord situation.

Please keep the above in mind if you want to look up a ticket that was created via this method. To emphasize, the *only* place you can look up a bare, no-underlying-JobRecord POS ticket (i.e., negative InvoiceNumber) is directly from the FinishedForm interface — by typing in its invoice number there as a target in the target box. And, when you do, it's critical to include the leading minus sign. If you do not include it (and if it's an item with a minus that you're looking for), the system won't find the ticket.

Aside from the above, please notice that just as soon as you conclude a Direct-POS transaction, the interface goes right back into a mode waiting for input of a sales person's two-letter code, to begin the next transaction. It is thus a system always-ready to perform for a never-ending succession of such transactions.¹³⁸

One more detail concerns the option to customize the text as presented under the POS form's signature line. It's obvious you might want to add your own particular text there (e.g., "NO RETURNS ON ELECTRICAL PARTS"). To do so, simply create the text you want and save it in a plain-text file named PosDisclaimer.TXT. Place this into the \sd\netdata folder on your server. ServiceDesk will do the rest.

iii. POS Integrations with Inventory and Parts Ordering

Regardless of which FinishedForm type (and/or context) is used, each has a section for line-items that are being sold. Within each line-item, there are multiple boxes, or fields (the succession of several line-items, each with fields, forms a series of columns). The first field within any line-item is for quantity of items. The second is for the partnumber (or, in the case of merchandise sales, the model number) as involved. The third is for description, and fourth for per-item price.

In regard to the second field, whenever the Windows focus first shifts into it (typically when you click into or tab to it), you'll see a little grey selection window appear to its right:

Qty	Part #	Description	Per Item Price	Ext
1		Integrate input from . . . <input checked="" type="radio"/> Parts Inventory <input type="radio"/> SmartParts Listings <input type="radio"/> Serialized Inventory		
		Inventory location from which this POS part is pulled OF		

This is the *integrate-selection* window. Its purpose is to allow you to pick what source you'll be integrating with as you type a part or model number. As you can see from the above, there are three choices. The first is appropriate if you are selling parts inventory directly from stock. The second is the correct choice when you're intending to input an item to special-order. The third and last would be the choice if you're going to sell merchandise — specifically, serialized inventory — as managed by the SD-Dealer program.¹³⁹

¹³⁸ On the prior page we had a note regarding a special consideration as connected with Callsheet-initiated POS tickets, where you have ServiceDesk's Departmentalization feature turned on. There is also a special concern regarding Departmentalization where you're using Direct-POS. It is that (where Departmentalization is turned on) each sale must be assigned to a department, yet there is no interface within the Direct-POS interaction by which to pick the applicable department. The solution is the system auto-assigns all Direct-POS sales to a department called "Counter Sales." It's hard-coded. In fact, if you turn on Departmentalization but have not created such a department, yet conduct your first Direct-POS sale, the system will add that department for you.

¹³⁹ SD-Dealer is one of the supplemental programs that may be acquired to work with ServiceDesk, but is not actually part of it. It's specifically designed as a mechanism to manage serialized inventory, and is basic but elegant. If you are interested, please contact Rossware for details.

Based on which source you select, the system will display an as-you-type dropdown, containing matches that fit whatever you've typed with each keystroke. Thus, you can typically type but a few characters, before seeing the desired item. At such point, simply select it, and the system will do a full insertion for you.¹⁴⁰

This is how integrations work in regard to using the dropdown to aid in populating any particular line-item with appropriate text. But there's another, potentially more important function. As earlier alluded, we want to use the POS interface as a mechanism for actually pulling items as used from our inventory (i.e., if I used one widget from a stock of three, I need to have the system log such usage, and decrement its count of widgets down to two). We likewise want to use it as a mechanism for creating special-order parts requests, where that's a situation involved with our POS transaction. How does this occur?

In a nutshell, it's a two-step process.

Any such *operative* transactions (operative in the sense of *creating* transactions within our inventory or parts-process systems) begin by having an applicable line-item *flagged* for such an event. Flagging is shown by virtue of any such line-item being shifted in color, with a particular color standing for each kind of flagging. The flagging of a line-item is done for you when you select an item from the offered dropdown. Thus, if you select from the *Parts-Inventory* connected dropdown, the resulting line-item will "flag" as yellow (the color that designates flagging for potential pull from parts inventory). If you select from the *SmartParts-Listings* dropdown, the resulting line-item will "flag" as blue (the color for potential creation of a special-order part request). Finally, if you select from the *Serialized-Inventory* dropdown, the resulting line-item will "flag" as orange (the color for potential pull from SD-Dealer-managed merchandise inventory).

Parts Used	
2	514
1	WP20K10008
1	6XV540U
1	ACE124PT1

	FILL VALVE	19.17	38.34	
	THERMOSTAT	53.85	53.85	
	LCDTV, TOSHIBA, [AM403005382]	2299.95	2,299.95	
	AIRCOND/H/AT, WHIRLPOOL, [QCU2601236]	599.95	599.95	

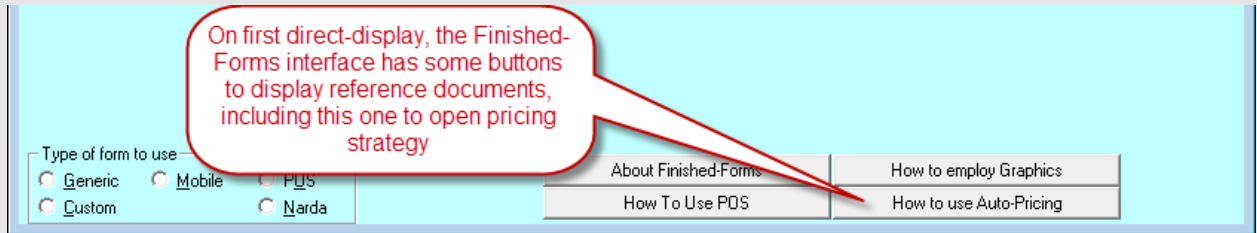
Yellow indicates flagged for pull to/from parts inventory

Blue is for create a special-order

Orange is for pull from Dealer Inventory

These are not the *only* mechanisms for flagging. They are simply handy mechanisms, if you happen to be pulling from the associated dropdown anyway. There is also a method for more *volitionally* flagging (or de-flagging, if that's the desired operation). If you want to volitionally manage the color flag, simply right-click in a line-item's description box. Doing so will produce another dropdown:

¹⁴⁰ You should note that, as rule, the inserted line is going to include sell-for pricing on the part. It raises the question: how does the system know what price to insert? This is actually a fairly complex topic, because we provide a lot of flexibility in how you can structure the underlying strategy. There is an entire document on the topic. You can access it via a button that's visible when you first direct-display the Finished-Forms interface:



Or (at least if within the PDF version of this manual), you may [click here](#). Please also note that, with each and every price insertion, the system attaches a ToolTip to the price box in question which explains that basis as used for the insertion. When you want to know the basis, just float your mousepointer over, and the ToolTip will pop up to explain.

1	ACE124PT1	AIRCOND/HEAT, WHIRLPOOL, [QCU2601236]	599.95	599.95
1	5918896	OTHER PART		

Do you want to:
☐ Delete this line item
 Or flag it for:
☐ No-action
☐ a Pull to/from Parts inventory
☐ a Pull from Serialized inventory
☐ Creation of a S/O Parts Request

This dropdown appears when you right-click in a line-item description box

As you can see, this dropdown includes a full set of flagging/de-flagging options, plus an option to delete the entire line-item, if that happens to be your preference.

Our overall point is you can use such mechanisms to accurately populate line-items within your POS form, so that it accurately reflects what you're selling to your customer, and how the interaction of such sales should relate with other elements as being managed within ServiceDesk. However, nothing actually operative happens until you tell it to.

This occurs during the next step, when you execute the POS transaction.

iv. Executing a POS Transaction

We prior referred to "flagging" line-items for action, because flagging is just that: it flags only, and does not do the underlying action for which the item is flagged.

To actually do the set of actions as flagged, there's a separate button on the form that must be clicked (whether virtually or directly). Not surprisingly, the button is labeled (with a bit implicit abbreviation) "execute Lnltsms:"

ticket/job w Job	Do Inclusions Only	Inclusion items (to do any any alone, just click on it)		Execute Sale
		receive Funds <input checked="" type="checkbox"/>	Print ticket <input checked="" type="checkbox"/>	
		execute Lnltsms <input checked="" type="checkbox"/>	Transmit claim <input type="checkbox"/>	

But you should notice, there are other buttons too, which are likewise associated with actually going forward with a transaction as prepared within and described by what you've setup within your POS form. Again, the button setup is structured so as to allow you to integrate multiple actions (according to preference) within a single click, or to click on any such action individually.

Quite simply, if you click on the "Execute Sale" button, you'll be offered all execution actions (whose checkboxes are checked) plus the SalesJournal entry itself (thereby allowing what is essentially single-click execution of all actions associated with the sale). If you click on the "Do Inclusions Only" button, you'll simply be offered execution of all the sub-actions (whose checkboxes are checked), but not a corresponding SalesJournal entry. This latter would be appropriate if you're ordering a part, in which case (as a formal accounting principle) a complete sale really should not be entered until the part is received and delivered to the customer.

Regardless, the general concept is you're going to finish a POS transaction by executing it, designing your execution to include whatever steps as should properly be involved.

In regard to "delivering" your part to a customer, a related topic is raised. Assume you prior used POS-execution (as above described) to, among other things, create the internal special-order request. Then via the

PartsProcess system, you ordered the part from your vendor, checked it in upon its arrival, then called your customer to indicate they should come in and pick it up. When the customer actually comes in for the pickup, what do you do? Simply, you'll want to bring up the JobRecord form (F7) as involved on the sale, choose its PrintOptions command, and from there pick FinishedForms. Choose to do a "*Fresh Import*" of information to the form, as this will bring in information regarding the special-order part as received. In particular, the part will show with double-caret symbols accompanying its part number, which signifies it has not yet been checked off as delivered to the customer. This check-off is needed as part of the cradle-to-grave parts management system. A simple double click accomplishes that check-off. This subject is further discussed within the chapter section that discusses PartsProcess management (see Page 133).

v. Accepting Returns

You sold a part. The customer comes back and wants to return it. Though there are a few ways the situation can be handled (including various schemes that involve re-use and editing of the prior ticket), all are subject to annoying quirks except the exact method we recommend here. We think you'll be happiest if you abide by the following prescription.

*Create a new ticket*¹⁴¹ for the new POS transaction you are now conducting.

In such regard, please understand that even though any return *relates* to the prior "ticket" on which the item was sold, as an accounting and operational matter it is nevertheless a *new* transaction, and from that standpoint deserves to be treated as such. As it happens, in fact, such treatment is not only superior as a *conceptual* fit. Turns out it's also better operationally.

In such regard, on your new ticket enter any item or items being returned, much as you'd enter if you were selling them. The major textual difference is that, in any return-item's quantity box, rather than providing a standard *positive* value, you'll provide a *negative* value instead. In other words, if you're accepting return of 1 widget, use a quantity of "-1".

Please also feel free on this new ticket to include items that you are newly selling to the customer. In other words, there is no problem with a single new ticket involving both items being accepted in return (negative quantities indicated) and items currently being sold (positive quantities indicated).

It's also true that if you want the POS mechanism to itself manage corresponding inventory movements for you (i.e., document movement of that widget back into inventory), any applicable line-items must be "flagged" for such movement (just as with a sale), and the movement must be "executed" as per discussion in the prior two sections. Regarding such movements, we have two caveats:

1. Presently, our POS return-acceptance machinery has automated ties only to *parts-inventory*, and *not* with the other two functions where it auto-ties on the "sale" side. In other words, while it auto-ties to the PartsProcess system on the sale side (i.e., it will *create* a request there when you *sell* a special-order part), it will not go find such a prior request and cock it for vendor-return if/when you accept the part back (meaning you'll need to attend to this separately). Similarly, while it auto-ties to Dealer/Serialized Inventory on the sale side (i.e., it will "pull" items from there when selling), it's

¹⁴¹ Our use of the word "ticket" here is deliberate. In general, "ticket" is a poor word for describing precise actions, because it can mean so many different things. In this particular context, though, we need a word with flexible meaning, and it's here in this footnote that we'll provide more precision. If you are using the direct-POS method (i.e., using the POS form directly, and with no underlying JobRecord), for that context a "ticket" would be defined as simply a new (completely new, fresh-initiated) POS invoice. If you are conducting the transaction on basis of a JobRecord, on the other hand, it means an entirely new JobRecord, with new invoice for that as generated via normal creation purposes, etc

not yet configured to “pop” such items back into dealer inventory if/when you accept return (meaning, again, you’ll need to do such work manually within its own operative area).¹⁴²

2. Since you are using a new ticket (with its own unique Invoice/Ticket Number), any inventory-system return (i.e., the kind the system *does* direct-manage for you) must know the ticket number as involved in the original sale. You will be auto-prompted to provide this number, directly within the line-item involved:

Items Sold				
-1	G31546	5 AMP FUSE [Rtrn from {provide ticket #}]	2.50	-2.50

The system self-appends a return-item's description with bracketed text, including a place for the ticket number on which it was originally purchased; just type to replace the prompting text

Items Sold				
-1	G31546	5 AMP FUSE [Rtrn from 73039]	2.50	-2.50

After typing the number, you end up with something that looks like this

Regardless of what’s auto-tied (or not) for physical item movements, financial/accounting elements will work exactly like in a regular sale POS sale, except they’ll (typically)¹⁴³ involve negative rather than positive values. In other words, as you invoke the “*receive Funds*” action, you’ll actually be giving money back to the customer (and simultaneously documenting the same). Applicable new insertions to your FundsJournal will simply be for negative (rather than positive) values. You can manage cash as being given back to the customer, or “plastic” credits, in the same direct, no-complications manner (in other words, proceed as if collecting, only with the negative rather than positive values; if you’re using our Virtual Terminal, it will know, with a negative value, to go into Credit mode). Similarly, as you invoke the SalesJournal entry, it will also work just the same as with a normal entry, except on returns it will also typically involve a negative value.

There are two potential rough-spots that may arise on the financial/money side of returns.

First is if you decide to refund money back to the customer by writing a company check. It’s rough only in the sense there’s no auto-integration — and can’t be — since ServiceDesk has nothing to do with managing your company’s checking account. If you decide to accomplish the refund in such manner, just do it separately, on your own, outside ServiceDesk. Go ahead and record the negative sale, but do not enter any movement of funds (it would not make sense, because the money-movement is one you’re managing separately). We had at least one

¹⁴²We’ll readily admit that for true integrated completeness, these added auto-ties should be present. It’s also true, though, that ServiceDesk is primarily a service-management system, as opposed to being the utmost/best-possible POS system. If not for the fact so many other developmental demands tug us toward other priorities, we’d address these elements of incompleteness immediately. We are leaving them for the time-being because, for now, there are higher demands on our developmental resources. This is particular true because: (a) in regard to special-order parts, we feel the only sensible policy is to not allow such returns in the first place; and (b) returns of items back into dealer inventory are likely a very infrequent event.

¹⁴³Exceptions will occur where you’re also selling new items on the same ticket, which sum to a greater value than items being accepted in return. In such cases, you’ll still be collecting money and entering a sale for positive (though reduced by the return amount) values.

client that expected ServiceDesk to at least compile a list of checks that needed to be written from this context. Sorry, it's not an element ServiceDesk manages.

Second is if you did your prior sale on credit (i.e., Paycode 2 with a still outstanding A/R). We have not at this time configured an integration that will reach into the prior A/R and auto-decrement its amounts (or eliminate it entirely, if you're doing a complete return) on the basis of your present return. So, suppose your present return ticket ends up at a negative value, and you have a present A/R for the customer where as much or more is owed; how do you handle that? Our suggestion is: (1) record your negative amount as a "paid" sale; (2) immediately open the prior A/R; and (3) add the negative total from your return ticket as a positive value to that prior A/R's PaidToDate field. It's a tiny bit of manual work, but should nevertheless end with an accounting-perfect result.

vi. Customizing the Finished-Forms Interface

We want to again distinguish FinishedForms from the up-front ticket-creation context. In the latter, we've invested intense programming capital to make an output that is almost infinitely (and very easily) customizable to user-preference (see page 269). FinishedForms are much less easily customizable, at least beyond a few easy elements.

Among the easy elements, we've already explained how you can customize text that fits under the signature in the POS form. In addition, it's also very easy to specify particular graphics you might desire, in lieu of plain text, for your company's header at the top of any particular form, or in some instances as the form background. There is a small document that explains how to setup and manipulate such graphics. You can access it via one of the instructional buttons that display when you first direct-display the FinishedForms interface (Alt-F4):



In the alternative (and at least in the PDF version of this manual), you can use this [link](#).

If the above measures do not provide sufficient customization — and in fact you require direct modification of an actual FinishedForm form layout — a much more intense investment will be required. If you're determined in such regard, we can make a truly-custom FinishedForm form layout for you, though there will be a significant fee (usually several hundred dollars, or more, depending on the degree of customization you seek). In the alternative, if you're intrepid and have a bit of programming bent, you can do it yourself. We have a complete instruction document for the purpose. It's contained within a folder on your installation CD, which also contains other elements you'll need. The folder is called "VbUtility." The instruction document, within, is called "Instructions.pdf."

Chapter 10

OTHER FUNCTIONS

We have now discussed the four major categories of ServiceDesk functionality: Call Management, Schedule Management, Job Management and Post-Completion Management. These are the core ServiceDesk functions, and the first three in particular involve processes where ServiceDesk innovates radically, providing revolutionary solutions like nothing else on the market.

This having been said, we now turn to a discussion of other, additional functions. Each of these work in close conjunction with those already discussed, but are either supplemental in nature or work so broadly as to have resisted inclusion into any single category.

Hint: If using the video tutorials, please read this chapter in conjunction with having watched Lesson # 6.

A. E-Mail

As many computer users already know, E-Mail is nothing more than the process of sending messages electronically (i.e., from one person's computer screen to another's), rather than on paper. It might seem that, in a relatively small office where everyone is within easy speaking distance, there would be little need, or utility, in E-Mail. Particularly when Callsheets can be used as a vehicle for passing small notes or requests (at least between office personnel), the idea of still another communicative method might seem redundant. However, even if you have only a boss and secretary working in the same room, and only one outside technician, you're likely to find that ServiceDesk E-Mail is one of your favorite tools.

Suppose you have a general announcement you want to make about an upcoming event or opportunity, or announcing a change in policy, or reminding of the importance of an existing policy, or congratulating for exceptional performance, or anything of the sort. You could post a paper note on a wall some place for everyone to read, but some might not see it, and besides, you have to compose and print or write the note, tape or pin it up, and so on. You could try to speak with each individual personally, but that's even more work. Instead, you can casually compose your message in E-Mail, and then with a single command instantly generate an electronic copy for each person in your organization. These copies will then appear for reading at each person's station (in regard to the tech's, each of their particular copies will appear as they individually check-in). And the copies will remain posted at these stations until deleted by their recipients, assuring that each has full opportunity to read and understand. Each has the further opportunity, moreover, to address his or her own E-Mail reply back to you (or to send their own original E-Mail if desired).

Of course, some announcements may be appropriate for your technicians only (i.e., they're not in regard to anything your office personnel need be concerned with). Maybe there's a new solution for a particular kind of repair, for example, and you'd like to share it with your technicians. No problem. ServiceDesk E-Mail can be addressed simply to "All-Technicians," thus creating a copy for each of them, and no copies for anyone else. Similarly, you can easily address a message to just office personnel (thus generating a copy for each of them), and not to your technicians.

More typically for E-Mail, there are obviously many times you need to communicate a matter to one individual. If that person is in the office with you at the moment the matter arises, it may be easiest to just talk. But if it's a technician who's presently out in the field, or an office person who's off-shift or sick while you have the subject in mind, it may be easier to compose a note. This allows you to clear your mind of the subject, knowing the information is safely in transit to the intended recipient. Even if you need to discuss something extensive and in-person with someone not presently in the office, you can clear your mind substantially by sending them an E-mail message requesting that, when they are next in the office, they bring up the subject for discussion with you.

Indeed, in many cases we have found there are questions we need to ask a technician regarding some job. It used to be that we'd make a mental or paper note to ask him when he was next in the office. But during the few minutes after he next arrived, we'd either be busy on the phone or simply forget, and he'd leave with the question still unasked. Sometimes this cycle repeated several times, the frustration level increasing with each. With E-Mail, you have an obvious and automatic solution.

There are two different ServiceDesk contexts from which E-Mail may be created or read: one is for office personnel, the other for technicians.

In the first context, ServiceDesk once again makes ancillary use, on behalf of office personnel, of the TechInterface form (we also use it, as you may recall, as a secondary venue for conducting CstmrDbase searches from the office, see page 209). To load this form and simultaneously request its E-Mail feature, press Ctrl-F12.¹⁴⁴ If there is any mail pending for you (a fact you should have already been made aware of, based on a flashing message in the MainMenu title bar, and a corresponding breep, breep sound), it will be displayed. If there is no mail pending, you'll be presented with a blank sheet of paper (at least the computer's best representation thereof), ready for composing your own E-Mail. In any case, it's easy to conduct whatever E-Mail business you may desire.

In the technician-use context, operations of E-Mail are pretty much the same. The difference is that the TechInterface form comes up automatically out of the Window/Tech-mode (that you'll presumably have that particular station set to), by the tech's simple press of any key. After he then provides his own TechInitials as requested by the form, the system searches for any mail addressed to him (including his copy of any mail that was addressed globally). Of course, it displays any mail so found, and he can reply or initiate any mail he wants as well. Much as in the office context, a technician should normally delete his own mail after reading it.

In regard to deleting your E-Mail, we found many users felt the need to keep a permanent record of all E-Mail activity. This enables the owner/manager to snoop on employee communications, if desired, but more importantly, allows you to check back to confirm some detail of past communication, resurrect a policy statement or general advisory (to possibly send out again), and even maintains some sense of history for your organization. In early versions of ServiceDesk, discarded E-Mails were, in fact, lost, but we've since amended the system, adding a new directory appropriately called 'OldMail' (look for it at 'c:\sd\oldmail'). Basically, when any E-Mail item is discarded, a copy is moved into this directory. The copy is named for the date on which it was discarded, followed

¹⁴⁴You may recall that, to load the TechInterface form for from-office, CstmrDbase-search-usage, the command was a *plain* F12. You'll notice (examine the indicated action in the menu bar command buttons as you hold down either the **Ctrl** or **Alt** keys), that if you want to load the TechInterface (again, for from-the-office-usage), but *without* simultaneously requesting either of these two functions, the command is **Alt-F12**.

by a an extension indicating the sequence of discards on that date (i.e., the second E-Mail discarded on 4/22/98 would have the following file name: '042298.002'). These files may be loaded into and read from any word processing program. If you want to re-use any portion of a letter, simply copy it into the Windows clipboard (see page 74), then paste it into your new E-Mail.

B. The UnitInfo System

The basic idea behind the UnitInfo system is simple. During any machine's lifetime, you may have occasion to work on it multiple times—and possibly even for different clients. The latter situation might arise, for example, where it was a manufacturer who first hired you to do warranty service (or a home-warranty company to do contract service), then later the consumer called you to do COD service. It might likewise arise if a house in which you'd done service was sold, and you were later called upon to do service (on the same built-ins) for the new owner. Regardless, you may have a more enduring relationship with a machine than with any of the particular clients who pay you to work on it. With such as the case, it would be silly if, each time you went back to work on the machine, you had to re-create the kind of specific information in its regard that's sometimes needed—such as, for example, model number, serial number, and perhaps purchase date and selling dealer. Indeed, it would be silly even if having to re-create this information when going back to work on the same machine for the same client.

Thus, we have what's called the UnitInfoSheet ("UIS"): a simple form you fill-out and which thereafter maintains the critical ID info that pertains to a particular machine. The beauty is that you create this UIS for a particular machine but once, and from then on, any job you do on that machine (regardless of for which client), you simply attach that same sheet to any new such job. Naturally, it brings all the pertinent machine info with it. You'll find this is very simple, and convenient.

The heart of the UnitInfo system (don't be surprised) is the *UnitInfo* form. You can display this by right-clicking on any Callsheet's 'Item(s) Make' box. As you can see, the form is quite simple—having spaces to fill-in, or display, each of the information items enumerated above. There are also a series of command buttons that are largely self-explanatory, but we'll describe them regardless.

Let's suppose you are an appliance servicer and receive a dispatch from Whirlpool to repair a refrigerator that was installed in the customer's home about one month ago. The dispatch includes all the relevant homeowner's information (i.e., name, address, etc.), and the kinds of specific-to-machine information that we're concerned with here. You fill-out a Callsheet, using a QuickEntry to insert Whirlpool's own name, address and telephone numbers (into the Callsheet's Customer-Info block), then type the homeowner's name, address and so on (into the Callsheet's Location-Info block). Then, of course, it's on into the Callsheet's Item(s) Type and Item(s) Make boxes to put info there. Now, however, rather than typing descriptions into these boxes, you may instead right-click on the Item(s) Make box. At this point you are presented with the UnitInfo form, where you may enter not only the Type and Make of item, but other relevant items of information too. Thus, you'll create a UnitInfoSheet, and attach it to the Callsheet. Then, when in another few moments you create a JobRecord from that Callsheet, the same attachment will carry through to that new JobRecord.

Of course, another possibility would be that this is a machine you've already serviced—in which case there'd be an existing record that already has the appropriate UIS attached. If that were the case, rather than creating a new UIS for the same machine (actually, once the system catches you entering the same serial number as exists in another UIS, it will not permit it), you'll want to simply attach the existing one. Because this is another possibility as you enter the form, ServiceDesk does not default to either format (i.e., creating a new UnitInfoSheet

versus attaching an existing one). Instead, the first two command buttons (proceeding along the bottom of the form) permit you to select from among these two, most common alternatives.

In particular, assuming again that you're presently setting up this warranty job for Whirlpool, we'll further assume that you've never worked on this particular machine. Thus, you need to create a new UnitInfoSheet in its connection. Since this is the more common situation, you'll notice that the command for the purpose is the one that already has the Windows focus as you entered the form. This means that you can simply hit Enter on your keyboard (*effectively* "clicking" the focused button) to select that function.

Upon so doing, you'll find your cursor is immediately positioned in the TypeDescription box. You might further notice this is a combination typing/list box. If this is your first use of the system, there will, as yet, be nothing in the list, so simply type-in the kind of description you want to regularly have used, within your company, for the kind of machine in question (most appliance servicers use "REFER" or "REFRIG" as an abbreviation for refrigerator, for example, but you can use whole terms or whatever suits you). Then hit Tab to move to the MakeDescription box. Here, of course (assuming our current example), you'll type "WHIRLPOOL", then Tab to the Model Number box, type as appropriate, and so on.

The system requires that you fill-in at least the first four boxes (figuring a UIS without at least that much information would be useless, while that much information in itself). Once this requirement is met, you'll note that two new command buttons are enabled: one to 'Save' the record, and another to 'Attach' it to your Callsheet (the latter action, as you might guess, incidentally saves the record too; the first is provided simply for those rare instances where you might want to save without incidentally also attaching). When ready, you may either click on the wanted command button or, for greater ease when wanting to proceed with the expected attachment, just press Enter.

Whenever you create a UIS that contains either a Make, Type or Dealer description that was not previously within its own respective list, you'll find the system demands entry of your password before allowing you to save the UIS. The reason is to enforce a bit of data integrity. In particular, the system will not allow use of a Make, Type or Dealer description that's not found, or that you're not willing to add into, its own respective list. This means when you pick something new for such a description, it's either got to be added into its own respective list, or you've got to relent, and pick something that's already in the list.

For a better understanding of why this is so, consider that mere adding of a new item *could* be made completely simple—except we've learned by experience that when free license is granted to add to these lists, they soon become clogged with an absurd panoply of varying descriptions. You might end up with both "RANGE" and "STOVE" in the Types list, for example (what's the difference?). You might likewise have "BUILT-IN RANGE" and "RANGE BUILT-IN." Under Makes, you might have "OKEEFE," "OKEEFE & MERRITT," "O&M" and "O & M." As you can imagine, it can soon become ridiculous, so a means is needed by which you can police these entries and be sure they conform to some reasonable scheme. That's why a user has to either use the same text as already exists in these lists, or use a password to add a new entry.

In terms of strategy for populating these lists, you can do it either on "ad-hoc, as needed" basis (i.e., simply create UISs as often as the need arises and, when a new UIS happens to involve a Make, Type or Dealer that was not previously in its respective list, use that occasion incidentally to add it to the list), or you can make a wholesale effort to populate the lists up-front. If this latter is your course, you can insert items to any of these lists simply by typing them in at the top, then hitting Enter on your keyboard to insert (you'll be prompted for password confirmation). If you want to delete an item, simply select it from the list by clicking on it with your mouse, then hit Delete on your keyboard. To edit the text of an item, similarly select it, then hit Ctrl-Enter on your keyboard (Hint:

there's a tip sheet summarizing these methods available in the form itself; look for white text in the form on which you can click to bring the tip sheet up).¹⁴⁵

Anyway (and returning to the subject of simply creating and attaching a UIS to your job-initiating Callsheet), suppose you've just done so. At this point, ServiceDesk returns you to the Callsheet. Here, you'll see it's entered a notation into the Callsheet's Item(s) Make box that references the "UIS" attachment. At the Callsheet level, this is ServiceDesk's only means of registering the attachment, so (assuming you want the attachment to hold), do not mess with this notation.

The next step occurs when you create a job from this Callsheet. At that time ServiceDesk registers the attachment (as indicated by the above-described notation), and in consequence takes a series of background steps while creating the job. Most important, it makes an entry into the UnitInfo Database that ties the job you've just created to the same UIS as was connected to its initiating Callsheet.

You can easily see the fact of this connection from the JobRecord. Simply hit F7 to bring up the JobsCurrent form. Since this is a job you just created, it will be the most recent in the JobsCurrent file, and therefore will automatically be the one displayed. The fact of a UIS connection is indicated here by two facts (in contrast, remember that a *Callsheet*-to-UIS connection was encoded solely by textual notation in the Callsheet's 'Item(s) Make' box). First, on any UIS-connected JobRecord, there is a line/shadow drawn under the Item(s) Type and Item(s) Make boxes. This provides quick, at-a-glance indication of the fact. Second, there's a, small radio button in the 'Potential Actions' section of the JobsCurrent form. If the JobRecord in question has no UIS attached, the caption on this button will read "Create UIS" (which, incidentally, suggests the fact that if you haven't already created an attachment when initiating the job from a Callsheet, you may nevertheless do it later from the JobRecord itself). Alternatively, if the JobRecord already does have a UIS connection, the caption on this button will reference the ID # of the UIS in question.

In either case (as you might guess), you can access the UnitInfo form by clicking on this button (alternatively, you can right-click in the JobRecord's 'Item(s) Make' box (just as you can from the equivalent box in a Callsheet). If there was not a previous attachment, you can create one. Or, if displaying an existing attachment, you can edit it, change to a different attachment, sever (i.e., break) the connection, and so on.

Still another context in which a UnitInfo sheet might be created and/or attached to a job (assuming it was not previously done) is when making the PostVisitReport. Whether using the old dialog or new fill-in-the-blanks method, provision is made for creating (or attaching to an existing) UIS as part of the process. Please note also that, if you sometimes work on multiple machines under the same ticket, you can attach up to five different UISs to a single JobRecord (this multiple-attach capability does not exist at the Callsheet level).

One advantage of having a UIS attached is that the system no longer needs to query for make, model and serial number when you're ordering non-stock parts (it already has that data, after all). Another is that if it's a warranty job (or other situation where Model and Serial data is required for payment), the system has the information, and can fill it into the final claim form for you.

Another incidental takeoff from the UnitInfo system concerns entry of information into Callsheets themselves. Once you've added a few listings into the UnitInfo's 'Type of Item' or 'Make of Item' listboxes, you'll notice a new entry aid as you're entering a Type or Make in the a Callsheet's own equivalent boxes. Now drop-down lists will appear as you type, and you can select the wanted ItemType or ItemMake from within such a list,

¹⁴⁵The file that contains all this UnitInfo data is in Microsoft Access format. If you want to make any separate use of it, or edit it from that context, you are completely free to do so. Running Microsoft Access, you'll simply need to find and open `\\sd\\NetData\\UnitInfo.mdb` from your system's FileServer drive.

rather than having to type it out in its entirety (you do have to have this feature turned on from within the Settings form, but that's the default, so you'll probably find it's already on and you don't have to worry about it).

In addition to the above benefits, the UnitInfo form provides a couple of its own, independent search-and-review type capabilities.

In particular, if you want to quickly review all the jobs you've done on a particular machine, just bring up the UIS as is relevant to that machine (either link to it from an applicable JobRecord, or independently bring up the UnitInfo form and locate the applicable UIS via an Serial Number lookup (Click on the button labeled '*Find an Existing Item*' and proceed per prompts). Then click on the button that's labeled '*Show All Linked Jobs*'. The system will show you a listing of all such jobs. You can click on any item and instantly see the full corresponding JobRecord.

Similarly, if you want to quickly review all the jobs you've done on any particular model number (regardless of the particular machine involved), you simply need to again bring up the UnitInfo form (from *any* context). Again, click on the button labeled '*Find and Existing Entry*', and follow the prompts. In this case you'll be shown a listing for all jobs that involved whatever model you indicate, and again you can select from any item in the list to instantly see the full corresponding JobRecord.

C. Making Reports

There is one primary form that's used to create many of various reports that may needed, on a periodic basis, regarding activity in your company. It's the *Reports* form, accessed by pressing **F11**. Two such categories or report involve sales and accounts receivable, which were separately discussed (in terms of making of this form for reporting) in their own sections, so we'll not further discuss those here. Instead, we'll focus on several other types of reporting which also employ the Reports Form. At present, there are a number of different reports available, and we intend to add others in the future. If you have any specific requests, please let us know.

i. Salary, Wage and Commission Reports

Whether you pay your employees on a salary, commission or wage basis, it's nice to have help in tabulating each individual's earnings. ServiceDesk makes it easy, even almost automatic.

Your first step, in setting up ServiceDesk to calculate employee earnings, is to designate the type of earnings, and rate, for each person. For this purpose, use the RateOfEarnings form, accessed from the 'File' section of the MainMenu or by pressing Alt-F2. Operation within this form is most self-explanatory. You should see a listing for each person that's been entered, either as a 'Station Name' (i.e., it's an office person), or in the 'List of TechNames', from within the Settings form. Simply click on the person whose type and rate of earnings you want to set (or review), then specify (or note) the settings accordingly. In regard to those technicians for whom you specify earnings based on commission, you can set independent rates for each category of sale (i.e., you can pay a particular percentage on Merchandise sold, something different on Parts, and so on for ServiceCalls and separate Labor).

The next step is to compile the data to which the various payment types and rates will be applied. In the case of those employees paid hourly, this obviously means you need to keep track of their hours on the job. There is no reason you must do this from within ServiceDesk (a traditional, mechanical TimeClock still works very well),

but for office personnel working at their own designated stations, ServiceDesk's built-in TimeClock feature is very convenient. Each such person can easily clock-in or out, simply by pressing F2 from his or her own station. ServiceDesk carefully logs each such event. Similarly, technicians can easily clock themselves, in or out of work, using the same feature—only in this context as accessed from their TechInterface form (a button to access the feature will appear whenever a tech, who's been designated for hourly pay, logs himself into the form; the button remains invisible to other technicians).

For those technicians paid on a commission basis, the relevant data (that which will underlie their pay) is already compiled, independent of any commission-calculating need. That data, of course, is precisely what's found in your SalesJournal (i.e., it shows every sale by every tech, broken into the categories of Merchandise, Parts, SCalls and Labor)—already compiled for purposes of plain record keeping, accounting, and so on.

To create an actual salary, wage or commission report, select the desired feature from your Reports form (to access, press F11 or click on the indicated item in your MainMenu). You may note that, for purposes of selecting a report-type from within the form, if it's either a periodic salary or hourly wage report that you want, you must select the one option labeled 'Employee Wages' (ServiceDesk will make the distinction, between the two types, based on what you have specifically designated in your RateOfEarnings form). In this regard, you may note that, while there is no direct requirement for you to keep track of hours worked by a salaried employee (i.e., total earnings remain the same regardless of hours), we think it useful to maintain such information regardless. A salary-type report in ServiceDesk will, moreover, look for a history of hours logged by the employee, and report on those hours just the same as if he or she were paid hourly. The difference is that, rather than calculating a variable pay total based on hours worked, it will instead indicate the fixed salary rate, then calculate and report on what the salaried employee effectively earned per hour. We find it useful to track such information.

If you pay any technicians both a wage/salary plus commissions, please note that your Reports form does not offer a combination report. Instead, if you want to show the hours an employee has logged on the job (and maybe calculate a portion of pay on such basis), do a wage/salary report. Then, do a separate Commission report to show and calculate that portion of pay.

At present, ServiceDesk will not calculate and report on withholdings (basically, it only shows total pay for the period, and the underlying basis therefor). We formerly had plans to add this feature, but given the variations in payroll taxes from place to place (and the complicated means for calculating), we presently think the best plan is to have ServiceDesk make the basic earning reports for you, then you may use a dedicated payroll package (available at any office supply), or an outside professional payroll service, to do the rest.

ii. Completion Analysis

A major concern for any service-call-performing company is to complete each job in as few trips as possible. Many times, when owners of such companies gather, you'll find them discussing the rate of "first time completions" each has managed to maintain within his or her company. This is the report that will calculate that figure for you. Actually, it has several breakdowns, and provides figures not just company-wide, but as to each technician as well. The latter figure is very useful, of course, for spotting which technicians are weak, in terms of completing jobs on the first trip (or second at most), and helping them improve.

There are couple of details you may want to know in regard to how these reports are compiled. Basically, the system looks exclusively at jobs that have already been moved into the JobsArchived file (as many back as you wish to tabulate for). It examines the 'History' section of each such job, and deduces how many trips were made by looking for key words that so indicate. Specifically, it looks for the beginning of the kind of entry that's made each time there is a PostVisitReport on a job (i.e., something like '11/17/99 8:44 DS there 16 Tues 1.30 to 3.05, . . .'). It

counts the number of such entries within each reviewed item's history, and scores that as the number of trips involved. When segregating numbers between the various techs, it's the tech who's on record as making the first trip who gets assigned all numbers pertaining to the job in question.

iii. QOS Analysis

For those companies that do a lot of work for Home-Warranty type clients, there's a constant concern for keeping certain averages within prescribed limits. The reason is obvious: the client is itself watching these averages, and if your figures are not good enough, they may dispatch far fewer jobs to you. So you'd like to know how you're doing, and if maybe you're veering off in the wrong direction. Most importantly you want to know this *before they do*. This report (QOS stands for "Quality of Service") is designed to provide you with that information. Specifically, at present, it provides figures regarding Average Ticket and Recall Rate, separately calculated for each of your clients that are designated as 'HighVolume' types.

Again, there are some details you may want to understand. As with Completion Analysis, this report looks exclusively to your JobsArchived file for its data. And, similarly, it reads the History section of each record that is reviewed there (again, you specify the quantity of records back that you wish to tabulate) to get its information.

In specific regard to calculating the Recall Rate, you should know that the system's sole means, for determining whether to classify a job as recall or not, is by looking within the job's Complaint/Request section (i.e., essentially the same text, unless later changed, as was typed into the equivalent box on the job's originating Callsheet). The system looks within the text there for the word "RECALL". If it finds the word, it counts the job as being in such category. Otherwise, it does not.

This obviously means that, if you are to have accurate tabulations in such category within this report, you must be consistent in assuring that this word ("RECALL") is placed in all job orders that belong in such category. Generally, we recommend that with any job that a Home-Warranty type client would classify as a "Possible Recall," you place these same words (or something similar, such as "POSS RECALL") in the Complaint/Request section. We know, many times such jobs prove to have been new, unconnected problems, but most home-warranty companies simply use that classification (or, rather, that of "Possible Recall") regardless, it it's simply for any job on the same appliance within 30-days of a previous repair. If you use the same rule in your company, and be sure that at least the critical part of the term ("RECALL") ends up in the Complaint/Request section of each such job's JobRecord, you'll have accurate reports in regard to how the home-warranty companies, at least, classify the matter.

iv. Tech Time Analysis

Still another major concern for any company that's concerned about quality service is to monitor how well each technician is doing in terms of arriving within prescribed time limits at the customer's home. And, to partially gauge his efficiency, there's also an interest in knowing how long he's spending, on average, after having arrived.

Information regarding these matters is provided by the Tech Time Analysis feature of the Reports form. Simply select that option from the form and, as with the related reports, indicate how many records back (within the archived job record) you wish to search. As you'll see, the resulting report provides a treasure trove of information regarding the matters above-referenced, and related ones.

Again, bear in mind that, as with the cousin reporting methods above-described, this report actually reads through job histories in order to glean the information that it compiles. This means the fundamental information has to be there in the first place, based on accurate PostVisitReports having faithfully been made by you and your

people. And it means that it takes a while for the report to be generated (reading through those histories takes time, even for a computer) if you select a very large number.

v. Callback Reports

As a business owner, you probably have a pretty good feel for which of your techs have relatively higher recall rates, and which are lower. But it's probably a seat-of-the-pants feel, and you're not certain if it's entirely reliable. And, more important than that, even if you feel certain, you realize that if you bring one of the guys into your office and tell him it's your perception that his recall rate is too high, and that he needs to work on it, the whole argument is not likely to carry a lot of weight. If, on the hand, you can show him a report—with solid, computer-derived figures—one that shows his recall rate is twice as high as the next worse tech, you know for certain it's gonna make an impression. Thus, you need a means of coming up with such a report.

Problem is, figuring recalls is not an altogether straightforward matter. As we all know, often customers will call in, insisting that a job must be a recall, and even if within mere days of an earlier repair, it's not uncommon to find a completely independent problem has developed, one for which the tech on the earlier job could not possibly bear proper responsibility. By all rights, such a situation should not be charged against the technician. If, however, we were to accurately refrain from classifying jobs as recalls in this situation, while catching ones that genuinely are, and classifying them as such, it would require that a fair-minded (not to mention technically knowledgeable) manager review every potential recall situation, and render a judgment as to whether it should properly be classified as a recall or not. Not only would this consume excessive time and resources; it would also engender dispute and argument from the technicians in various cases, rancor and what not. All in all, to try to render a judgment in each instance and attach it for the sake of statistics, is probably not a good idea.

What's needed, then, is a different strategy, one by which we can come up with at least comparative rates of recall by which to compare the techs (even if inflated by *apparent* recalls that in fact were not), and do so without any actual judgment having to be made in each instance.

We've come up with two strategies by which this can be done. You may use either or both. We have the two strategies because, depending on how you conduct your business, one may work for you while the other does not, or vice versa.

The first strategy requires no separate effort on your part whatsoever, so long as it is your normal practice to create UnitInfoSheets in connection with most every job.

The second requires that text within the JobRecord be setup in a way that ServiceDesk is able to read the text and thus recognize the job as a recall (or at least possible recall) on the basis of the text itself. Since computers don't have any true IQ, you have to arrange the text in a particular manner to help ServiceDesk in this "reading" process. Specifically, you should do one of two things: either (a) manually type the word "RECALL" ("RECALL" is also okay) within the Description section of any job that potentially is a recall;¹⁴⁶ or (b) insert the info-set to your Callsheet from the previous job using the "oriented-for-recall" method of insertion (see page 67).¹⁴⁷

¹⁴⁶ In other words, assuming that Mrs. Jones calls in, telling you the tech was there last week, fixed the dryer, and now it's not heating again., your operator could type something into the Callsheet's Description box like the following:

GB REPLACED ELEMENT LAST WEEK, NOT HEATING AGAIN, POSSIBLE RECALL

It doesn't matter where the key word is placed within such text, or even whether it's separated from other text as a distinct word (***RECALL*** would be okay, in other words). If you're using this method, however, you should have a consistent policy as to when your call-takers are expected to insert the word (such as on any occasion when the previous job was within 30 days, for example), and do a little policing, to assure they're following the policy with reasonable consistency.

¹⁴⁷ In this case ServiceDesk inserts the info-set from a previous job to a Callsheet (when you use the Ctrl-RtClick method of insertion from the previous job's CstmrDbase reference), but adds some text to denote the recall, specifically at the right-end of the CustomerName line it inserts

As you might note, the first strategy might well be preferable, since it doesn't require you to do anything aside from your normal work—assuming that in fact you're normally using UnitInfoSheets. The second strategy is provided as a substitute method for those operators who, for one reason or another, find themselves not using the UnitInfoSheets with the regularity and consistency that's required to make the report method, that relies upon them, work effectively.

Regardless of which strategy you choose, to create a report simply go to the Reports form (**F11**), choose 'Performance Analysis', then select the 'Callback/Recall Rates' option. You're then offered the choice of which method to use in compiling your report. If you've consistently been UnitInfoSheets to your jobs, the first method should give you pretty good numbers. If you've been placing the word "RECALL" into the Description section of jobs that were potential recalls, the second method should give you pretty good numbers. If you've done both, try both reports and see how they compare. Hopefully (and ideally), you should find the numbers are pretty darned similar.

One suggestion. When using the resulting numbers with your techs, you'll likely need to educate them on the fact that you realize there's some proportion of the numbers generated that do not represent true recalls. You'll have to educate them to the fact that it's just not practical to make and tally judgments on a job-by-job basis, but that all techs should, presumably, suffer from the same rate of "false" recalls (i.e., being called back on the same machine within 30 days even when the first repair was done perfectly). Thus, even if the figures are, say, 10 percent too high for all techs, it's the comparison between techs that matters—and if one tech is significantly higher than the others, he is just plain that—since any inflation from false recalls should affect all equally. Hopefully, your techs will not be too dense to get this (I once had one who was, or at least pretended to be).

As a final concern, for the sake of the technicians' feelings (and if you're using the key-word method), it's probably best to always use the phrase "POSSIBLE RECALL" rather than just "RECALL" in the text. The reason is most techs take a lot of pride in their work, and can quickly feel a lot of resentment upon seeing the word "recall" when not feeling it's justified. Indeed, you may need to educate them to the fact that it's simply policy for statistics purposes to place the phrase there in all circumstances of going back on the same machine within 30 days (or whatever rule you've made), and does not yet reflect (at the time of creating the ticket) any judgment whatsoever concerning whether the former work was done correctly.

If, incidentally, you are concerned about your customers seeing the phrase and concluding on its basis that they simply should not be charged the second time around, I think, if anything, the opposite is more likely to occur. By placing the phrase within the ticket, you're demonstrating to the customer your complete willingness to be up-front on the issue. If the technician finds there's a new, unrelated problem, the existence of that phrase on the ticket proves the company had been ready and willing to consider it a recall—which makes it more credible when and if the tech happens to find otherwise.

D. Miscellaneous Printing

As you've no doubt noted, there's many different contexts for printing information, out onto hard-copy, from within ServiceDesk. Primarily, these various means and methods are designed to print a specific kind of report or other image onto paper – and in every such case there are contextual commands (and generally a command button) for such purpose.

a phrase such as "C/B # 17943-5". In addition to looking for the word "RECALL" in the description line, ServiceDesk's Recall-Report,-Key-Word-Method also looks for that particular phrase. Thus, if either is found, it knows to interpret the job as such.

Still, you may note that we have not created provision for purpose-designed printouts in every context. In some cases (in terms of information that's available for viewing on-screen), we've judged that your need for a purpose-designed printout is likely to be either extraordinarily rare, slight, or both. Since it involves significant work to create the underlying code-machinery for each purpose-designed printout, we've not created purpose-designed reports in these instances.

However, we've not left you helpless. Should you occasionally the odd need for a printout in any case where it's not otherwise provided, there's a very effective solution. From virtually any context in ServiceDesk, if you want a printout of what's on the *active form*, simply press **Ctrl-P**. If the active form is one with freely displayed text in it (i.e., text that prints directly on the form itself, not within a text box), ServiceDesk will print the text that's displayed. If, on the other hand, it's a form that is simply composed of various controls (including text boxes), ServiceDesk will print an image of the form, including its controls and their contents.

As still another possibility, if you'd ever like to print an image of the *entire ServiceDesk screen*, that's very easy too. Use the same command as mentioned above (Ctrl-P), only before you do so press the **PrntScrn** key on your keyboard. Particularly if using a laser printer, you'll find that a very nice image prints out for you.

Also in regard to this last application, there may be occasions where you want to print what's on the screen from another application (i.e., one not even related to ServiceDesk), and there's no provision *there* for doing so. No problem. ServiceDesk will be happy to help you out here as well. Again, just press the PrntScrn button on your keyboard. This places the image in the Windows clipboard. Go to ServiceDesk, and press Ctrl-P. If it's an entire screen image in the clipboard (no matter what screen image was captured when you pressed the button), ServiceDesk will print it.

Think of this Ctrl-P based print method as being for *miscellaneous* printing, wherever there is otherwise no specific provision for a hard-copy printout of what's visible on-screen. If you remember the feature is available, it will come in very handy at times.

E. Making a Customer Mailing List

While we have long believed the best method for generating customer goodwill is by providing superb service at a good price, we have found there are servicers who believe it's profitable to do even more. Specifically, some have made it their habit to follow up every job with a "Thank You" note mailed to the customer, not to mention annual holiday greetings, and so on. We think this is a lot of expense, and that customers might be happier thinking their payments did not provide so much profit as to pay for such friendliness. However, since several buyers have requested a utility for generating mailing lists from within ServiceDesk, we have chosen to provide it.

Actually, and all fun aside, some companies use mailing lists for direct profit-making activity, sending out mailings that offer things like air conditioning tune-ups in the spring, or winterization of plumbing in the fall, for example. It's our understanding that such direct-marketing campaigns—to customers you already know—can be most successful.

In truth, we initially resisted creating a Mail List-making utility because there are so many potential complications.

For one thing, there are many customers you will have done jobs for repeatedly, yet certainly, you don't want to send multiple pieces of mail to them, at least not within the same mailing. You therefore need some means of assuring that subsequent jobs to the same customer are equated within the computer system, so that it will

produce only one entry for that customer within your mailing list. This might sound simple, but computers are not that smart. If a single letter of a name spelling is different (or a different first name used, or a less complete name, or whatever), it will look to a simple computer algorithm like a different customer, and in result (absent some solution) you'll get multiple references within your mailing list, in spite of the fact that it's the same fundamental customer and household involved. This can result in much wasted money in your mailings, not to mention making you look less than efficient to your customer.

Another problem is that entries in a mailing list eventually become old and outdated, suffer inaccuracies that need correction, and so on. A mailing list may also take up hard drive space that could better be used for other purposes, and require effort on your part to manage, correct, update, and so on.

In creating a Mail List-making utility in ServiceDesk, we were determined to create no additional overhead in terms of input and maintenance, and to make a reasonably accurate list, with absolutely no redundancies, and no separate file storage requirements. We are pleased with the result, and hope you will be too.

To create your mailing list, load the MakeMailingList form from the ServiceDesk 'File' menu. Press the 'MakeList' command button, and ServiceDesk will do the rest. Repeat this routine as often as you want to update the list to include jobs that have been completed since the last time you created it. The list itself is created de novo each time, based on entries in your JobsArchived file (remembering this is your source, you should not expect that jobs will be referenced that are not yet complete and archived out of the JobsCurrent file), in combination with your CstmrDbase indexes (thus, if these are not current at the station from which you run the routine, results may be inconsistent).

For a particular job's customer name and address to make it into the list, it must be the most recent job for that customer in the JobsArchived file, and must include full mailing information, including state and zip, etc. (this means that if you want all your customers to potentially be included in a mailing list, you should select the option within the SettingsForm that will consistently provide for such insertions—see page 225). In determining whether any two jobs should be equated as belonging to the same customer, ServiceDesk compares the name, address and telephone numbers of every job in your JobsArchived file. If any one of these three fields is the same between a set of jobs, ServiceDesk concludes that all such jobs must be from the same customer, and, therefore, allows a resulting Mail List entry for only one of them (again, the most recent).

The resulting output file (file name and location specified by you) consists of customer name, first line of the address (i.e., street number and street name), second line of address (i.e., city, state and zip), and date the source job was initiated. It is in comma-delimited, Ascii format, which should work fine as a source for creating form letters or mailing labels from most word processors. You should further be able to specify, from within your word processor, a range of dates (from the ServiceDesk-made list) that you wish to include in your mailing. Thus, you might specify the creation of mailing labels only for those mail list entries that include dates between, say, January 1, 1996 and the present—or whatever you choose.

In general, you should find the list is remarkably accurate. However, it certainly will not correct for misspellings in the originating documents, and there are a few other limitations. It is necessary in creating the list, for example, for ServiceDesk to convert the all upper case format of each source JobRecord into the upper and lower case that is suitable for a more formal mailing. ServiceDesk is fairly effective in this regard, but unfortunately, lacks the brains to realize that the "O" in "P.O." (as in P.O. Box) needs to be capitalized, or that the "M" in McMurtry should be, and so on. Thus, there will be some imperfections in a few of your Mail List entries, but certainly, not so many (at least as created by ServiceDesk) as to be significantly embarrassing.

Since it is in simple Ascii format, you can easily view your Mail List (and edit it if wanted) from any word processor. Simply load the file name you specified when creating it.

F. Finding Various Customer Records—A Summary

Because there are so many available, we want to provide a synopsis here of the various kind of customer records you can look up, and of the contexts in which you may do so.

First among the records, of course, are the Callsheets. These are created, obviously, when a job (or often merely a telephone call) comes in. When their task is complete, they are moved and stored permanently within the CallsheetArchive. From this context past call records can be viewed page-by-page, or of course you may conduct specific searches for any particular name segment (see page 80). Sometimes it's very handy, even, to use this as a kind of telephone book. You need a number for so and so, and remember you must have fielded a call or two from them in the past. Just do a quick search in the CallsheetArchive and, wala, there's the information.

More typically, of course, you'll be interested in looking up information that directly describes past jobs—for which purpose JobRecords are obviously much more useful than Callsheets. Current JobRecords can be viewed record-by-record in the JobsCurrent form, archived ones in the JobsArchived form. Specific searches may be conducted from either form, but it's typically much better to search in both records simultaneously using the CstmrDbase utility. As explained elsewhere (see page 283), the underlying data for this utility consists of records from both contexts, and it uses *indexes* to rapidly search among and locate the particular records wanted. There are several contexts in which searches may be conducted from among these indexes. We'll summarize them here.

First, there's the *auto-CstmrDbase-Search* utility, which (so long as it's turned on) functions automatically whenever you're typing into a name, address or telephone-number box of a Callsheet (see page 66). A search is likewise conducted whenever your cursor is in such location and you also press F1. Second and closely related, there's the JobsCurrent form's built-in CstmrDbase-Search, which allows you to press F1 (from within the JobsCurrent form) to search simultaneously on all of an existing job's relevant fields (see page 106). Alternatively, you can put your cursor in a particular field and press F1 to search specifically on it. Third, you'll very frequently be using the convenient CstmrDbase-Search utility that's provided as part of the TechInterface form.

Given that latter form's name, you might think its CstmrDbase search feature would be oriented more toward technicians than office personnel, but this is not the case (in fact, given the form's expanded functions it ought to have a different name, but we haven't thought of one yet). At any rate, whenever you want to lookup something in the CstmrDbase and you're not otherwise in a Callsheet or similar context in which the search text either exists already or is something you must type-in, there, regardless, the easiest method is to just hit **F12**. This loads the TechInterface form and selects its CstmrDbase search option. Thus your only remaining step (after hitting that one key) is to begin typing in your search target (no need even to specify whether it's a name, address or telephone-number you're looking for). As the list of matching items pops up, you can click on any one to view it fully (or more easily hit F1 to enter the list and display the first item, then use the cursor keys to move up or down viewing others). Also note that from here you can directly print the history of any selected item (see page 74), a feature that's not so conveniently available from other CstmrDbase contexts.

While CstmrDbase-oriented searches are extremely powerful, you should remember their limitations. Consider, for example, that any JobRecords added since the last indexing event will not appear in a CstmrDbase search. These must be found by direct searches in the JobsCurrent form (however, if you use the Auto-Archive function, there will never be more than a day's worth of jobs in such category). And there are *kinds* of searches that cannot be run from any of the various CstmrDbase contexts, but can be from elsewhere. If wanting to locate a job by P.O. Number, for example, you'll have to conduct a search directly from the JobsCurrent form (for current jobs) or JobsArchived form (for archived jobs). Or if wanting to locate by street name, a utility for the purpose exists only within the JobsArchived form (thus, it's impossible to search from among current jobs by street name).

If you consider the path of job progression in ServiceDesk, you'll realize that when a job or mere call first comes in, there is first a Callsheet created, which we've already discussed as forming the potential for subsequent lookup. Then, when an actual job is created, there's the JobRecord. We've just discussed the potential for lookup in those. Now what else happens? What other kinds records are created that we might have occasion to lookup?

Well, there's the process of ordering non-stock parts. These incidents leave specific records in the PartsProcess system. Maybe you want to know, for example, what kind of part you ordered for so-and-so two years ago, who you ordered it from, and for how much (or how many times you've ordered that part, what's the range of prices paid, etc.). Use the PartsProcess Archive to quickly look it up and find out.

Then there's the process of using, restocking and reordering the parts you do stock. All records regarding such history are kept in the InventoryJournal that's part of the InventoryControl system. To lookup and review this kind of data, use the InventoryControl form.

What about funds collected from a customer? There's a couple of files applicable here, the FundsJournal and the ApplicationsJournal. Depending on context, either or both may be used when wanting to lookup the relevant history regarding payments from any particular customer, or in connection with any particular job (and of your related deposit activity, etc.).

Amazingly, all these records may be created before a job is even completed. And with that event, even more records are produced. Most importantly, there's a record that contains an entry describing each and every completed sale (i.e., the SalesJournal). Many times, you may want to lookup a specific such record. Use the SalesView form. Of course, if there were still amounts outstanding when a sale was completed, there will also be an AccountsReceivable record in its connection (at least until paid). Use the AccountsReceivable form to lookup and review these.

The primary point here is that there are a multitude of methods which allow you to quickly put your finger on the particular record that has the information you want. We urge you to become familiar with and use all of them.

G. The Source Of Jobs Survey

This is a feature that, even if ServiceDesk did *nothing else* for you, should easily repay the purchase cost, and many times over. With use of this feature you can easily generate iron-clad, scientific data regarding the type of jobs you are doing (i.e., new, recall, etc.), where you're getting them from (i.e., repeat customers, referral, OEM or home-warranty types, etc.), and of those customers that found you in the Yellow Pages, which ads they used and in what quantity.

You'll notice this feature can be turned 'on' or 'off' from the Settings form. The reason is because, simply, it is not necessary to conduct your survey year-round. In fact, there are important reasons not to. At some point in the year you've got old telephone books being retired and new ones delivered. Conduct a survey during this period and you'll have mixed results, failing to show clearly what the performance is in any particular book. You'll want to conduct your survey, therefore, during a non-transitional time, when the set of books you're interested in is not being changed. Typically, two months of survey time is all that's necessary to produce solid, statistically significant results.

When you have this feature turned on, an important event occurs just as you finish entering a customer's appointment in any Callsheet. At this moment, a survey form will pop into your Callsheet, prompting you to ask the customer a brief series of questions. The survey is designed to invoke in this manner, and at this time, for three

reasons: first, it is really only those customers who've scheduled jobs that you're interested in surveying, and this method assures exactly that; second, it delays the survey until after you've taken all the other information, made the appointment, and secured a service commitment from your customer (thus, it's generally too late for them to be annoyed and call someone else); third, you're asking the questions while they're still on the line and have the ad they used in front of them (thus, there's no need for investing the time to call them back, and you're able to secure real information).

There are actually only a few questions in the survey, and often you can fill in the answers without even querying your customer, depending on the circumstances. As in other forms, you may indicate the appropriate response from among listed alternatives either by clicking with your mouse or by moving over the item with your cursor and pressing Enter or the spacebar. The questions are also structured so that if a particular response logically makes the next inquiry unnecessary, that inquiry simply does not appear. On average, the survey should consume less than 30 seconds.

Of course, even 30 seconds can be too much when you've got three other calls on hold, and perhaps you'll have some customers who'd rather not be surveyed regardless, meaning there must be a provision for your operator to skip the survey when needed (she can merely hit Esc). Yet, there is a design concern here, for if there was anything systematic about which kinds of calls tend to go un-surveyed, it would skew the reported results, making them invalid. In fact, by its very nature the SourceOfJobs survey is easier to complete in some cases than others (if the job's a recall, for example, or from an OEM or home-warranty company, there are only one or two questions needing answered, and no inquiries are needed from the customer—meaning that an operator will tend to complete the survey much more often in such cases).

To correct for such bias among initial completions, we must assure that every job makes it at least partially into the survey, eventually at least, and with at least the information that's essential for adjustment purposes. Thus, in all cases when the normal survey has been avoided, before it creates a JobRecord ServiceDesk will demand completion of a mini-survey, a briefer query that requires no questions from the customer. This can easily be done after the flurry of calls is over, and corrects for any bias in the kinds of calls initially not surveyed. In fact, by tabulating the total number of incoming orders (whether fully surveyed or not), it allows ServiceDesk to infer (on the basis of the total figures and the answers among those fully surveyed) how many total callers would have fit into each of the answering categories, as though all had actually been fully surveyed.

To setup the survey, you must first create a list of your various Yellow Page ads (see the Appendix). To view the data that's been gathered, at any point either while the survey is in progress or afterward, press **Ctrl-F11**, which will load the *SourceOfJobs* form, showing several successive pages (use the PgUp and PgDn keys) of up-to-the-minute, tabulated results. To print the data, press Alt-P from anywhere within the form.

The reason this feature is so valuable, obviously, is because it will quickly reveal which of your ads work, and which don't. Thus, you can in the future spend your advertising dollars much more effectively, perhaps investing more in those locations where you now know it's effective, and perhaps eliminating entirely your spending in locations where, you now know, it's not effective. Likely, you will be surprised to discover how totally ineffective some of your past expenditures have been. In our area we've found that only two (out of approximately ten available books) have any significant advertising effect. This, obviously, is very valuable information.

H. Red-Flagging Problem (or other Special) Customers

Unless you're very lucky, you'll occasionally have customers that cheat you, abuse you, or otherwise act in such manner that you want to put their name on a list that, essentially, says watch out for this person.

Or, you may have an opposite situation: VIP customers that you need to pay special attention to. Or, there may be any of several other potential circumstances where, when a particular comes up, it's beneficial for an operation in the system to be reminded of some particularity concerning them.

It's for any of these situations that ServiceDesk is equipped with its own "*Special Situations Advisory*."¹⁴⁸ This is simply a place for keeping track of the identity of these customers, of recording what they did to you (if it was a bad customer), and what your treatment of them should be in the future (i.e., whether refusing service, demanding up-front payment in cash, treating them extra special, etc.). And, it's a system that automatically flags any future effort to service such people, whether you're looking out for the fact of not.

The primary basis for this system is found in what we call the "*SSA form*" (for "*Special Situations Advisory*"). To access this form, press **Alt-F11**. You'll find, when you do this, the system makes certain assumptions. In particular, if you invoke the action from a Callsheet *with text in it* or from a JobRecord, the system assumes you want to make a new SSA-Record having to do with the customer as found in that source context. Thus, it displays the SSA form with such text pre-loaded for you, so to make a new record all you must do is add any applicable additional notes. If **Alt-F11** is invoked from any other context, by contrast, the system will assume you're simply wanting to review existing SSA records, and so will load in that fashion.

Problem-Customer Advisory (Item 1 of 1)

HENDOZA, MANUEAL

9 CALABRIA LN [862C4]

FOOTHILL RANCH

393-7343

Created 12/09/99

REC'D \$45 BANKCARD THAT WOULD NEVER CLEAR, FAILED TO MAKE IT GOOD

☐ Do not service again

☒ Service only w/Manager Approval

☐ Consider Special Circmtns

☐ Other

Save Delete Exit

In addition to these contexts for loading the form, there are two others that will volunteer themselves on the basis of other actions, whether you're thinking about the need or not. In particular, if you're in the AccountsReceivable form and writing-off a bad debt (see page 179), or if you're in the Funds form and writing-off an uncollectible item of money (see page 162), the system will essentially figure to itself "Hey, if you're losing out on this money, there's a pretty good chance you're going to want to "red-flag" that customer against future service. Accordingly, the system checks to see if the person's name is already in the SSA-List. If not, it asks if you'd like to add it. If so, it again assists you in the process by displaying the form with as much text as possible pre-loaded for you.

You may conclude, therefore, that entries are added to the SSA-List either quasi-automatically (when engaging in separate actions that are deemed as probably applicable) or as otherwise voluntarily initiated by you. The form itself is self-explanatory, as you'll see upon viewing it. You'll simply save an individual record there for each problem (or other special situation) person, describing what's important in their particular regard.

So, now that you understand where these records are kept, and how they are added there, the next question is: How is the fact of such a record brought to your attention, in a buzzers ringing, lights flashing kind of

¹⁴⁸Point of history: Until December of '09, this was called the "*Special Situations Advisory*." However, we found that a number of people were using it for a broader purpose (as now described here), and we decided it should be re-named to accurately reflect such a broader purpose.

fashion—whenever, perhaps some months or years after having created a SSA-Record, you’re again about to take a job order from the person involved?

The answer is that entries from the SSA-List are made into part of the CstmrDbase system.¹⁴⁹ The result of this is that, in any of the various contexts in which a listing of past jobs appears (such as when typing a matching name, address or telephone number into a Callsheet, for example), references to applicable SSA-Entries will also appear—in the very same CstmrDbase list as you’re already perusing for other purposes. Thus, it’s very likely you’re going to see the SSA-reference (they’re distinguished in the list by a series of “#” signs in the reference line) if it’s applicable to any new job you may be proposing to create.

And, in this case if you click on the reference, rather than showing an entire JobRecord (not applicable, because this references does not refer to a JobRecord), the system will instead display the the particular SSA-Entry involved, thus allowing you to instantly review the basis for your earlier complaint against the person, or other special notation (like other aspects of the CstmrDbase system, searches will also connect on the basis of matching address, telephone numbers and email address).

PCA-Item

Call Sheet # 70

MENDOZA

Customer Name, Address and Phone numbers

Location Name, Address and Phone numbers

Item(s) Type Item(s) Make Date and Time

Problem to be solved, and/or Description of Customer's Request

Active Desk

My own

Another's

Current

Hibernate

delete

Job/Sale

others On

Originated:

GR 11:20 am

Mon 11/26/01

More info?

Empty

MENDOZA, CHERYL 24391 NUGGET FA 425-9355

MENDOZA, JAVIER 23212 #7 ORANGE 380-1564

MENDOZA, JAVIER 23212 #7 ORANGE 380-1564

MENDOZA, MANUEA 9 CALABRIA LN [393-7343

MENDOZA, MANUEA#####9 CALABRIA LN [#####393-7343

MENDOZA, RICHA 21191 LARCHMONT 380-4335

MENDOZA, RICHA 21191 LARCHMONT 380-4335

MENDOZA, RICHA 21191 LARCHMONT 380-4335

MENDOZA, RICK 24332 DONNER CT 360-7901

MENDOZA, ROBERT 2928 BONANZA [9 361-5660

MENDOZA, ROD 24972 HIDDEN HI 495-9585

MENDOZA, ROD 24972-8 HIDDEN 495-9585

MENDOZA, SHERYL 24391 NUGGET FA 425-9355

¹⁴⁹ Please note, in this regard, that just as JobRecords will not appear in any CstmrDbase search until after an “indexing” routine is run (either as part of a WIP-Archiving event [preferably done automatically on a nightly basis] or as explicitly performed via the ‘MakeNewIndex’ function in the JobsArchived form), neither will references to these PCA-records. Thus, assuming you’re using the Auto-Archive feature (see page 239), you can expect to automatically see references to a newly-added PCA-Entry by the next day—but won’t on the same day that the item is added, unless you manually invoke one of those two routines that will update the indexes.

Page 213

Chapter 11

UTILITIES

Like most any advanced software system, your ServiceDesk package includes more than just one program. Indeed, upon installing ServiceDesk you should have noticed that its resulting Windows Program Group includes several program icons: aside from ServiceDesk itself, there's also *SD-Backup*, *SD-Tools*, *SD-Manual* and *Zips*. The purpose of this chapter is explain the functions of these other, auxiliary programs. In addition, we'll describe a few features that, while they are *part of* the main ServiceDesk program, may nevertheless be bundled in the same category as these other utilities in that their purpose is to manage operation of the program itself (as distinguished from aspects of the program which operate functions of your business, as discussed in preceding sections).

Hint: If using the video tutorials, please read this chapter in conjunction with having watched Lesson # 7.

A. SD-Backup

Because it manages so much of your service operation, you'll quickly find yourself becoming dependent on ServiceDesk in a degree that might be considered dangerous—dangerous in the sense that if you happened suddenly to suffer a catastrophic loss of data (i.e., if the fileserver's hard disk were to suddenly crash, with no backup), you'd be lost, severely handicapped in the ability to continue functioning with your normal degree of panache and flair.

As any seasoned computer user knows, there are many systems on the market designed to provide a periodic backup of critical data. Typically, such systems are setup to do a nightly backup of files onto a magnetic tape or writable CD. Certainly, you might consider using such a system to regularly backup your ServiceDesk files. However, besides involving substantial investment in hardware, these systems suffer other limitations which make them potentially less than adequate for the needs of a ServiceDesk environment.

The first problem is that a nightly backup will not save data until after the business day has ended. So, what happens if your fileserver's hard disk crashes around lunch time? Obviously, you will have lost data pertaining to the entire morning's work. This includes all the Callsheets regarding incoming calls that morning, all the JobRecords regarding jobs created and dispatched, all the PostVisitReports from techs regarding jobs done the preceding day, all the entries regarding parts used, replenished, and/or ordered, all the entries regarding sales completed, even the list of jobs scheduled in what was its current state (with additions, deletions and changes since the previous night). Obviously, there's a lot of important data to lose from such a

short period of time, and the task of attempting to reconstruct it (and attempting to continue uninterrupted operation in the meantime) might be formidable.

The second potential problem is that, to save media, most backup systems are setup to write new backups over old. Thus, Wednesday night's backup may write over Tuesday's, and so on. To see why this might cause trouble, suppose one of your personnel, working in the Windows File Manager, inadvertently corrupts one of the ServiceDesk files (like the one containing a record of all your past sales, for example). Suppose this isn't noticed right away. You all go home at night, with your ServiceDesk directory containing a corrupted copy of the file, and your backup containing the only remaining good version. Then, precisely when programmed to at midnight, your backup system turns on and copies the corrupted file, writing it over the one good backup. Now, suddenly, you have no remaining good version of the file.

SD-Backup is designed to address both these problems, for any system or network that has any additional hard drive, besides the primary server, available. Installed on the secondary drive (typically from a satellite station in the network, but potentially from a second drive within the primary machine), SD-Backup will make copies of ServiceDesk files once per hour, once per day, once per week, once per month, and once per year. Each of these categories of backup will write into its own sub-directory, over-writing previous backups within its own class, but never over the backup from another. Thus, the most you'll lose from a fileserver hard disk crash (which, by definition, you'll certainly know of immediately) will be one hour's worth of activity (on average, the loss will be only 30 minutes). And, in the event of file corruption that's not detected for at least an hour but less than a day, you'll still have the daily backup to fall back onto (losing up to 24 hours worth of activity in regard to that one file, but no more). If the corruption is not detected until even the previously good daily backup is copied over, you'll still have the uncorrupted weekly backup to use (losing no more than a week's worth of activity on that one file), and so on.

It goes without saying that absolute data security would be even better, but in the real world, complete security for moment-by-moment activity does not yet exist. The SD-Backup system provides the next best thing. Of course, it has to be running to do anything at all. To run it, simply click on the SD-Backup icon from the station where you want your backups located. Then, leave it running in the background, full-time, permanently. Thus, you'll assure continual protection for your files. ¹⁵⁰

¹⁵⁰ To assure that SD-Backup is reliably re-started, each time your computer boots up, we suggest you setup your Windows installation to make it do so automatically. That way you never have to worry about it. Turn the computer on, and with nothing more you can be confident SD-Backup was started, is running, and is doing its task. The method for doing so is different, depending on whether you're running in a Win98 or equivalent version of Windows, versus an XP or equivalent version.

For the Win98 and equivalent series, your basic strategy will be to place a copy of the SD-Backup icon into the *Windows StartUp* folder. There are several ways you could approach this, but probably easiest is to first **right-click** on any vacant portion of the **Windows Taskbar**, then click on **Properties**, then select the **Start Menu Programs** tab. In the form which now appears, click on **Add**. In the 'Command Line', type in '**c:\sd\sdbackup.exe**', then click on **Next**. Now, in the 'Start' hierarchy that's shown, locate the folder called '**Startup**', and select it, then click on **Next** again, then on **Finish**.

In XP or equivalent version of Windows, the above procedure is no longer possible. Instead, you can use either of two methods:

WinXP Method 1: Go to the Windows **Control Panel**, and there select the **Scheduled Tasks** icon. Then click on the **Add Scheduled Task** icon that's presented, then follow the Wizard prompts to the point where it asks you to "Select the program you want Windows to run." At that point, click on the **Browse** button, and locate **SdBackup.Exe** within your **c:\sd** folder. Select it, then click on the **Open** button. At this point you'll be presented with a box that will present a list of options as to when Windows will run the program for you (i.e., whether daily, weekly, monthly, when you log on, etc.). Select the option labeled "**When my computer starts**", then finish the Wizard, and you'll be done.

WinXP Method 2: Begin by placing a copy of the SD-Backup shortcut into your Windows Clipboard. To do this, click on **Start** from the Windows Taskbar, select **Programs**, then **Rossware Computing**. Look for the item labeled **SD-Backup**, right-click on it, and from the pop-up menu select **Copy**. Now you've got a copy of that shortcut in the Windows clipboard, and it just needs to be copied into the Windows Startup folder, which you can find (using the My Computer utility) at the following location: **C:\Documents and Settings\All Users\Start Menu\Programs\Startup**. Upon finding that folder, simply right-click on it, then from the pop-up menu select **Paste**.

Regardless of your Windows version (and which method used), we highly recommend setting up for this automatic startup of SD-Backup. Don't rely on your memory. You're likely to forget, and then sometime when your server crashes (or something else goes

We should mention, incidentally, that while there is no strict rule requiring that you use SD-Backup, you'd be most foolish to risk running without it. You'll become so dependent on ServiceDesk, security of its data becomes paramount. As any experienced computer user can tell you, "things happen." SD-Backup is effortless, silent, and very gentle on computer resources. Please see that it's properly running, and that it's kept so.

In the event you actually suffer data corruption or loss, your next task, obviously, will be to secure recovery from your backups. This, essentially, involves locating the uncorrupted backups, and copying them back over, into the appropriate directory, on your primary FileServer drive (or onto a newly appointed FileServer if your former one has catastrophically crashed).

One of the considerations, in this regard, is that before you copy any particular backed-up files over formerly working ones (or into a newly appointed FileServer drive), you may want to first review them for accuracy, lack of corruption, state of currency or not, and so on. ServiceDesk has a terrific built-in feature, for this purpose, called ViewBackups.¹⁵¹ Access it from the File section of the MainMenu, or by pressing Alt-1. As the name implies, its function is to allow you to view any ServiceDesk data in any of the various (i.e., hourly, daily, weekly, monthly or yearly) backup locations. You are required simply to select the drive and directory for the backups you want to see—after which each ServiceDesk form behaves just as it normally would, except now they are each accessing data from the selected backup directory, to allow you a full review of any file you are interested in. The one difference, you'll notice, is that the Menu bar flashes endlessly in this mode, and in an abnormal color, all to continually remind that you're viewing backups, and not regular ServiceDesk files. To return back to regular file access, either select ViewBackups again (MainMenu or press Alt-1), or assent to changing back when ServiceDesk so suggests (as it does once each minute).

Once you've determined that any particular (or all) normal operating files need to be replaced by one or more backups, the next task is to copy the desired backup (or backups) into the appropriate location. At present, ServiceDesk has no built-in utility for this purpose—meaning you must use Windows Explorer, FileManager, or DOS commands to accomplish the task.

Remember that all the primary operating files, for ServiceDesk, are kept in the \sd\netdata folder of your FileServer drive (see Chapter 11.B: Directory and File Structure). Any other files (in other \sd\ sub-directories) can either be reproduced with some ease, or are relatively unimportant (except the backups themselves). It is, specifically, the contents of the \sd\netdata folder that SD-Backups will be making regular copies of. If a particular operating file becomes corrupted, it is that file, in this location, that must be replaced (i.e., copied back over with a good backup). Use whatever Windows utility you're most comfortable with.

B. SD-Tools

The purpose of this utility is to provide a venue for performing tasks that do not need addressed as part of ServiceDesk's daily operation. Creating an Invoice-Format-Instruction file, for example (see page 278), is something you'll generally need to do but once. Creating your list of Yellow Page ads is a once-yearly task.

wrong), and you need the backup, you'll find it hasn't been running in weeks. Make it start automatically from the particular computer where you want it to run, and (so long as that computer is running) you'll never have to worry about it.

¹⁵¹This feature is also handy, occasionally, even when you *haven't* had a file corruption or loss. For whatever reason, occasion may arise when you want to see what some file (or the data that goes in a file) looked like yesterday, last week, or whatever. It's easy to find out with this utility.

SD-Tools is designed to assist you in tasks such as these, and need be run only when you are performing them (as specifically addressed in the Appendix). For all other times, SD-Tools is a program you'll leave dormant.

C. Zips

Actually, in providing this utility, we were mostly having fun. We'd secured the data and created machinery, internally, to quickly connect city names with any zip code you might happen to throw out (an ability essential in creating your StreetList). It occurred to us that some users might find it convenient to have this ability at hand, and its reverse: the ability to throw out a city name and instantly see all zip codes applicable to it. So, we made a simple application that does this for you. Just click on its icon; operation is self-explanatory. The one drawback: as mentioned elsewhere, in some cases the city name, that's connected to a zip code, is not very accurate. Blame the Post Office for that.

D. The On-Screen Manual

Your ServiceDesk package includes an electronic version of this manual. It's provided in Adobe's .PDF format, so is viewable on-screen via use of Adobe's Acrobat Reader (more on that later). The file containing the manual (appropriately called 'sdmanual.pdf') is installed in the 'c:\sd' folder, and could easily be run/accessed via Windows Explorer (or similar means) if wanted. More easily, there's an icon for it in the RossWare program group. Just click on the icon and (providing your system already has Acrobat Reader installed), the manual will display for you.

One advantage of the on-screen version over the paper manual is that you can click on any page number reference (within the text or table of contents, at least) and instantly be transported to that page. Another plus is that you can always have the most current manual-type information available, even if your original paper manual has long-since fallen out of date (don't worry, the original paper manual is always up-to-date when first shipped)—as it most certainly will (and probably with some rapidity), for we are constantly improving the ServiceDesk system, adding new features, streamlining others, creating better methods here, shortcuts there, and so on. Naturally, as we make changes in the actual program, we simultaneously make changes in the manual to reflect the fact. Plus, we often make independent changes in the manual simply to improve *it*, independent of any underlying program changes.

For these reasons, it is important that you make a concrete effort, over time, to assure that the manual you are using is reasonably up-to-date. That's where the on-screen manual comes *most* into play, for it's easier to keep this one up-to-date (electrons being comparatively cheap) than to keep replacing your paper version, over and over again, and with much rapidity. Any time you receive an update via CD, it will include an updated on-screen manual, which will install as part of the process. If you're updating over the Internet (via email or website), we'll typically include the current manual as an optional, separate download. Regardless, please make an effort to assure that whatever your source of reading is, it's not excessively out of date.

In this regard, we realize you may, like us, prefer the immediacy of actually flipping through pages in a physical book, when wanting to review information. Thus, while the idea of a perfectly up-to-date electronic

manual may sound well and good, it's still the paper version you may find yourself wanting to use. So how can you do this, and yet still be reading something (say a year from now) that's not grown badly out of date? There are a few potential solutions. (1) You can always print-out particularly wanted excerpts from the *current* on-screen manual. (2) You can even print the entire current manual (from the on-screen version) and have it bound into book form at your local Kinkos or similar operation (or take in the CD, which will always include the current manual [at least current for whatever version is on the CD], and have them do both printing and binding for you). (3) We are happy to provide a new paper manual (updated for all the most recent changes, of course) as often as you want, but for a reasonable fee (typically the fee will be greater, probably, than the cost of doing one yourself at the local Kinkos). Just check with us for current rates.

Regardless, if you're to use the on-screen manual at all, it is (as mentioned), necessary to have Adobe's Acrobat Reader installed on your machine. In that regard, we'll here provide a small explanation. Acrobat is a utility developed by Adobe to allow for universal document exchange. Basically, any document can be converted to their .PDF format (which we do to our manual before providing it as part of your package), then viewed correctly (at least most of the time) from any other context simply by using Reader. The format is now so common that most computers end up installing it, for one purpose or another, before long. Thus, your computer probably already has it, and there'll be positively no need for you to concern yourself with these last two paragraphs of description.

If, on the other hand, your computer does not yet have the Acrobat Reader, you'll need to install in order to use the On-screen ServiceDesk manual. For this purpose, we've provided the necessary files on your ServiceDesk installation CD. Just click on Start in your Windows TaskBar, then Run, then Browse, then select your CD drive (usually it's D:). Once you have the contents of the installation CD displayed, click on the folder within that's labeled "Acrobat". Within that folder, you'll see a file named "Acrd4enu.Exe". Simply select that file and run it. It's the utility that will install Acrobat Reader for you.

E. The MainMenu

Though not actually a utility per se, the ServiceDesk MainMenu is discussed in this separate section because, in experienced day-to-day operation, you should almost never need to use it. For novice users, however, it may be most helpful, providing a map to functions that are not otherwise apparent.

Most specifically, you may note that the Menu provides access to almost all non-form-specific functions, without the need to know the Quick-Key combination that would otherwise (and more easily) access the function. Plus, corresponding with each such function's listing in the menu is a reminder of the Quick-Key that might, more conveniently, be used instead. Thus, you can gradually learn which Quick-Keys to use, by reminding yourself what they were each time when, as a beginner, you access something from the menu.

In addition, you'll note there's an entire section in the Menu labeled "Command Summary." This provides an easy, on-line synopsis of the most common commands you're likely to need, under several categories. Some of the items (particularly those under the category labeled "Callsheet Controls") even offer tutorial-type information. Thus, the menu (and particularly this section) can act as a constant, on-line help guide. Also remember that particular sections of this summary are available contextually, from within the particular domains to which they apply.

Aside from the menu per se, there is, of course, its row of 12 command buttons, designed for rapid access without having to reach into actual menu listings. As noted elsewhere, these command buttons may be accessed by a simple left-click of your mouse. But more efficiently, they serve as labels, indicating the function for each of your keyboard's 12 function keys, arrayed in corresponding fashion at the top of your keyboard. Thus, if you want to look at your DispatchMap, and notice that the fifth button on your menu's command button bar is labeled for such purpose, it follows that you can correspondingly hit the fifth function key on your keyboard—for the same purpose. The result will be the same, whether you click on the command button or hit the corresponding function key. It doesn't matter—except that, over time, you'll find it's usually faster when you can keep both hands on the keyboard.

Also, in regard to those menu/command-buttons (and your keyboard's corresponding function keys), you may notice there are only twelve. Yet, we've found the need for more than 12 buttons for bringing up various forms. Accordingly, we've expanded the utility of some button/keys by adding a Ctrl or Alt prefix for differed functions (e.g., you may click/press a plain F7 to load the JobsCurrent form, or Ctrl-F7 to load the JobsArchived form). To see the altered function for each button/key under Ctrl or Alt combinations, simply hold down Ctrl or Alt and watch the labels change. (You'll notice that not all the button/keys have changed assignments; we simply haven't yet found need for all of 36 potential combinations).

Another feature of the menu is its title bar, which displays your company name, with current weekday and time on one side, and date on the other. It flashes alternatively to indicate the status of Callsheets needing serviced at your desk (if any), along with the status of any mail pending for your station (if any), etc.

Note that the MainMenu functions, for the most part, in the same manner as you've grown accustomed to in other Windows applications. The exception is that we've not provided any QuickKeys to access its various headings. The reason: we've used virtually the entire alphabet for QuickKey operations that are more important. This left few keys available as candidates for non-mouse Menu access (and of those that are still available, few would have worked well mnemonically). In consequence, access to our Menu is by mouse only.

F. The Auto-Archive Feature

As you may have noticed, there are a number of files in ServiceDesk that need to be *archived* from time to time. Essentially, archiving is analogous to what happens in your file cabinet as one year ends and the next begins. If your office is like most, you probably have one or more drawers where, throughout the course of a year, you assemble all the various bank statements, paid bills, contracts you've entered, and so on. When a new year starts, you probably close out that old drawer and open a new one. Eventually, the contents of that older drawer will be moved to a box or boxes and stored in a dusty room somewhere. That, essentially, is archiving. The general idea is that you want to keep your *current* work area exactly that. Ancient records may occasionally need accessed, but the current area is made much less weighty (and accessible) if the ancient stuff is at least kept in a separate place.

It's no different with several of the files that ServiceDesk uses, except we do the archiving process more often than once a year. A current area can be kept much lighter (i.e., less to look at, easier to find what you want, more instantaneous access by your computer, etc.) if old stuff is occasionally put elsewhere.

There are five primary contexts in which this process occurs in ServiceDesk.¹⁵²

Most obvious are the Callsheets. The Callsheet pages (how ever many you have going at a given time) are intended to reflect current work in managing calls, service inquiries, and so on. When a Callsheet has completed it's purpose, it's marked accordingly (either as having been the basis for creation of a 'Job/Sale', as 'Otherwise Done', or for 'Delete' status). Conceivably, we might have designed the system to instantly remove a Callsheet from the current area (and into the Archive for all but the last category) whenever it's status is so changed. We did not, however, for removing an item from the Callsheet lineup changes the pagination of everything that comes after. It's a significant event, and for such reason we thought it best for the process be done on an occasional, batch basis (i.e., letting multiple "done" items accumulate as work progresses, then periodically, and only when ready, removing all such items at once).

Similarly, the JobsCurrent file is intended to contain records pertaining only to jobs that are in-progress. We want completed jobs moved into the JobsArchived file. But that movement is a very serious event, because CstmrDbase indexes must be updated at the same time, copied to each station, and so on. Then there's the ScheduleList, where again the intent is for the current file to contain only current appointments, and for appointments from the past to be regularly moved to archive. This event is not so significant in terms of what must be internally managed, yet by nature should be done on a batch basis, for it's by the passing of a day that an entire set of appointments (those from the previous day) suddenly become ready for removal. Additionally, there's your PartsProcess and AccountsReceivable files,¹⁵³ where the same or similar considerations hold true.

As you've learned on previous reading, there are methods to *manually* invoke each of these various archive procedures, on an individual basis. Prior to Version 3.8, that's exactly what we expected you to do, as often as might be deemed beneficial. In a typical operation (say one having five techs), we expected that Callsheets should probably be archived a few times each day (press Alt-R), the ScheduleList once at the beginning of each day, the JobsCurrent file at least once a week, and the PartsProcess and AccountsReceivable files perhaps once a month. The problem was, most users considered the duty of doing this something of a nuisance, and in fact often failed to invoke the procedures as often as they should have. So, we created a solution: the Auto-Archive utility.

Using the solution is simplicity itself. Simply "*turn it on*" from within the Settings form. Do this from whichever computer it is that you want to have performing the procedure (not from any others), and leave that computer turned on (with ServiceDesk running) overnight. At precisely 12:30 am¹⁵⁴ Monday through Friday,

¹⁵² A sixth context involves a kind of quasi-archiving. It's in the FundsJournal (see page 178). The difference is that there we do not use two separate files (i.e., one for current data and a separate file for old). Instead, we use a moveable marker within a single file (called the "CurrentAreaPartition"). It's purpose is to allow the system to know how far back, into the file, a search can be limited and still be certain of retrieving all non-checked entries. Like archiving in general, movement of this marker is an "event" –type procedure, because a significant search must be conducted to assure accuracy in the process. However, the system self-invites this process on detecting significant need (a quick check being made each time you load the Funds form). Of course, if you permit the procedure and yet have allowed old, non-checked entries to accumulate in the file, the system will be unable to move the CurrentAreaPartition forward. If so, it will inform you, and suggest that you amend the situation. Given such structure as this, we deemed it best not to include this particular, quasi-archiving procedure among those that are invoked automatically during the Auto-Archive process.

¹⁵³ Actually (and in contrast to the other records here discussed), old AccountsReceivable entries are currently not saved in any separate file. Instead, the process simply *deletes* those entries that have been either paid or written off. We have never encountered a need to review such records in archived form (particularly since the ApplicationsJournal, see page 198, records payments received on any receivable). If you have such a need and would like the old records to be genuinely "archived" (rather than simply thrown out), please let us know.

¹⁵⁴ You might notice that, with this automatic procedure added to a few others, your computers have a busy nighttime lineup of work to perform. At 12:00 am on any business day, the *SD-Backup* utility should run its procedure for the 'daily' category of backup (see page 233). At 12:30 am, there's this *Auto-Archive* procedure. At 1:00 am, one of your stations should run the *WipAlert* procedure (see page 134), and at 1:30 am, perhaps another the *WipAlert-Supervisor* procedure (see page 170). Be sure to leave the applicable machines turned on, with ServiceDesk or SD-Backup running (as the case may be), so these procedures can indeed be run (unless, in the

the system on that computer will self-initiate archive procedures in each of the five files discussed. It will further copy updated CstmrDbase indexes to each of the other stations (for this purpose, obviously, you must leave them running also and connected in the network, but it does not matter if ServiceDesk is running on them at the time).

That's all you've got to do. No fuss, no muss. With so little effort, all these files will be daily cleansed of old stuff, and your CstmrDbase indexes kept up-to-date.¹⁵⁵ The only limitation is that, for Callsheets, a regular nightly archive may be less frequent than you want. If so, continue to manually invoke the procedure (for Callsheets only, the QuickKey is Alt-A) as often as wanted throughout the day (at least you'll start with a freshly archived Callsheet work area in the morning, based on the automatic procedure having occurred overnight).

If, on the other hand, you're against leaving your machines running overnight¹⁵⁶ (as is needed for the Auto-Archive procedure to implement *automatically*), there is a "next-best thing" that allows you to turn your machines off overnight—while still keeping the burden of archiving minimal. Basically, you can *manually* invoke the same meta-process that would have occurred, overnight, had you had the system setup for it. Under 'File Functions' in the MainMenu (there is no QuickKey for this), you'll see an item labeled "Invoke Auto-Archive." This features does, obviously, exactly what its name suggests. Specifically, it immediately invokes the same process that would have occurred automatically, overnight, had you had the system setup to do so.

If you do turn your machines off overnight, but will nevertheless make it a practice to religiously click on this item once each morning (before the rest of your work begins), you'll have the same benefit as if using the full-auto system, and with *scarcely* more work. The primary downside is that, to fully keep your files fresh, you must force yourself to remember to do this each business day (whereas, with the full-auto system, there's no such need).¹⁵⁷

alternative, you want to manually invoke the auto-archive procedure after turning on the machines upon arriving in the morning, as described in text following).

¹⁵⁵ If there are any stations in the network that do not receive updated copies (the station that does the Auto-Archive will be waiting, when you arrive in the morning, with a report regarding all copied updates), you'll need to figure out why. For such purpose, well here explain the requirements that must be met in order for the archiving station to recognize and send copies to any other. (1) The other station must be running and connected in the network at the time the Auto-Archive procedure takes place (i.e., 12:30 am on the morning of any business day). (2) The other station must possess the appropriate "c:\sd\cstmrdb" folder on its hard drive. (3) If ServiceDesk is running on that other station at 12:30 am, it must not be a version prior to 3.8. (4) From within Windows, the *archiving* station must have "mapped" the other stations hard drive with a local drive-letter designation (see page 338) you may confirm the last requirement by checking, from the archiving station, to see which drives are offered in the 'FileServer' designation box of the Settings form. All other stations, that you want to have copied with CstmrDbase updates, should be found there.

¹⁵⁶ While opinions vary, there is a significant plurality of computer gurus who believe it is *better* for your computer's health (particularly its hard drive) to be left running, at least most of the time, especially if it is actively used on a near daily basis. If you have its energy-saving options set appropriately (so that various functions "hibernate" after some period of non-use), the electrical draw from doing so is quite minimal. Plus, there are other tasks (automatic hard drive maintenance and so on) that can be facilitated much more conveniently if this is your practice.

¹⁵⁷ Even so, there's another potential use of this "manual-invoke" feature. It happens sometimes in our office that, though we intend to always let the full Auto-Archive function take care of us (thus we leave our machines running overnight, etc.), occasionally, we'll inadvertently go home at night, having left without ServiceDesk running on the machine where the Auto-Archive feature is turned on. Thus, we arrive the next morning only to find the process was not done. No worry. Just click on 'File Functions' in the MainMenu, then on 'Invoke Auto-Archive', then watch as the procedure goes through its tasks—exactly as we'd *intended* for it to do overnight.

G. The Settings Form

As mentioned in the chapter on installation (see page 13), the Settings Form displays automatically when you first run ServiceDesk after it's initial installation, requiring some minimal settings before ServiceDesk operation can proceed. After that, you can access it from the 'File' section of the main menu, or by pressing Ctrl-F1.

Basically, the form is used to set the primary operating parameters that are specifically customized by you for ServiceDesk operation. It's divided into two sections: one for 'Local Settings' and the other 'System-Wide Settings.' In regard to the latter, we have sometimes used the term "Network Settings," which is something of a misnomer, for settings here are relevant whether you have a network or not. The real distinction is that, while the local section refers to parameters that are particularized for the specific station at which these values are being set, the system-wide (or "network") section refers to parameters that will apply to the entire ServiceDesk operation, regardless of whether it consists of just the one station or of many in a network. If you have multiple stations, changes made in the local section will affect just the one station where it's done. Changes in the system-wide section will alter the values for the entire system—regardless of the station at which they are done.

In the initial setup process, you probably did not take the time to set all the relevant values to where they need ultimately to be when you're actually using ServiceDesk to operate your business. Let's take the time now to explain each parameter, and what you can do with it.

i. Local Settings

In the upper-right corner of this section of the Settings form, you'll notice two boxes, side-by-side. The first of these is labeled 'Drive Designation for the ServiceDesk Network FileServer'. If you are running ServiceDesk from just one station, the setting here should probably remain at the default that is offered: 'Drive C:'. Of course, if wanted (even on a single station system), you may designate a different drive if available (although you'll have to add the basic ServiceDesk directories to it). More importantly, if you will be running ServiceDesk simultaneously from multiple computer stations (i.e., you really are networking), please understand, the setting in this box is key to success in the process.

In general (and so far as ServiceDesk is concerned), the networking scheme is very simple. As one component, you must separately install ServiceDesk on each computer where it will be used (you do not have to do this immediately, however, and certainly may add ServiceDesk to any new computer as convenience and need dictate). As the other component, you must decide on which particular computer you want ServiceDesk to maintain its common data files.

Basically, all stations must read, as it were, from a common book. In other words, if you have an operator at one station who adds a job to the schedule, this fact must immediately be evident from all other stations (i.e., as someone at another station looks at the schedule, he or she must immediately see it with the new addition). This task is accomplished by keeping all the basic operating files for ServiceDesk (i.e., the ScheduleList, JobRecords, SalesJournal, and scores of other operating/data files) in one place—or, more specifically, on one particular computer's hard drive. This makes it so that, when changes are made to such files (regardless of from which station), it's the one information set that is modified. And, of course (and by the same token), it's this one information set that is read by any other computer in the system. Thus, and by such means, all stations are enabled, as stated, to "read from a common book."

When referring to this commonly-accessed drive, we think of it conceptually (and refer to it here) as the "FileServer." Understand, however, that aside from the fact that it's where we've decided to keep those common-to-the-entire-system ServiceDesk data files, it's just an ordinary drive (i.e., no more than the basic hard drive on one of your computers).¹⁵⁸

In ServiceDesk, so far as this scheme is concerned, there is one simple fact each station must know: where it is supposed to find those common data files (i.e., which drive is the FileServer). And that, incidentally, is the precise and only purpose for this little box in the Settings form (i.e., the one labeled '*Drive Designation for the ServiceDesk Network FileServer*'). This is where you must inform each station of where to find that "common book."

You may note, in such regard, that this is a *local* setting, because each station must separately know where to look, for the wanted drive, on the basis of its own Windows-based perspective. And, importantly, you should realize that, depending on how the networking has been setup within Windows, the designation for this drive may vary from one station to another.

If you are wanting to run ServiceDesk within a network, it's very simple—at least so long as the Windows work (of setting up the underlying networked drive access) has already been accomplished. Basically, just click on the down arrow in this box (the one labeled '*Drive Designation for the ServiceDesk Network FileServer*'), then identify and select the networked drive you'll be using for the common ServiceDesk data files (i.e., the FileServer). Then, click on the 'Save Local Values' button.

Do this at each station in which you'll be running ServiceDesk, so that all will ultimately look to, read from and write to that one common drive (which, from the FileServer itself, incidentally, will likely be designated as its own Drive C:). Really, it's that simple.

At least, we should say, it's that simple so far as ServiceDesk is concerned. But remember, we are assuming that work within Windows, to setup access to the networked drive, has already been done. Unfortunately, you may find that the underlying work there has not been done. If it has not, you'll find that, as you click on that box's down arrow to display available drives (with the intent of selecting some particular drive from another computer for the FileServer), that other drive does not appear. In fact, unless you've already setup other applications for networking in such manner, you probably will not see the needed listing. This means some Windows work is needed. If so, it will be the one matter—a Windows matter—that may complicate your ServiceDesk networking process.¹⁵⁹

In regard to further ServiceDesk setup, we'll now move our attention one box to the right. Now, we're in the upper-right corner of the of the Settings form 'local' section. Here, you'll see a box labeled 'Name of person using this station'. Why is this name wanted? One thing you'll find, in ServiceDesk, is that as various tasks are performed, ServiceDesk documents who it was that performed them (and typically the date and time

¹⁵⁸Or, perhaps, not entirely ordinary. Certainly, you may logically decide to use your *fastest* computer and drive, or *largest*, as FileServer. You may even decide to use a machine that's dedicated solely to the task. Aside from possible effects on speed and reliability, however, it doesn't matter to ServiceDesk—except that its basic directories, at minimum, must be installed on any drive that's designated as FileServer.

¹⁵⁹To help you with such matters, we've provided a helpful Technical Reference in the Appendix (see page 338). In a nutshell here, you should understand that our '*Drive Designation*' box (within the ServiceDesk Settings form) obtains its list of available drives *from* Windows. More specifically, it will display (and allow you to select from) only those drives that are first *mapped*, from Windows, with appropriate *Drive Letter Designations*. You should understand, this "mapping" process is *not* an automatic consequence of merely installing network cards and getting them talking. On the contrary, it's something that must be done from each computer that wishes, as directed solely from within an application, to read from another's hard drive. Until the process is done in a particular machine, there will be no reference to that other drive in this ServiceDesk drop-down list (again, see the Technical Reference beginning at page 338 for instructions on how this is done).

as well). Basically (and with some exception), if work is done at a particular station, ServiceDesk assumes it was done by the person who is designated, here in this box, as the person assigned to the station. Plus, if any mail is sent to a particular person, or a Callsheet's transferred to that person, and so on, ServiceDesk goes by the name designated here to determine whether such matters should be displayed at such person's desk. Thus, it's important to put the actual name here of the primary person who'll be using the station (or desk) in question. By convention, we put just first and last name (in normal order, no commas), in upper and lower case (i.e., do it like **"John Smith"**).¹⁶⁰

Now, below these two selection boxes in the top-right corner of the form, you'll see an options frame that allows you to select the appropriate CommPort for your computer's modem. In most installations, this will be CommPort 2, but it might, potentially, be any other. If the setting is not correct here, ServiceDesk will be unable to access your modem for autodialing and similar functions.

Finally, arranged in a long column (and somewhat separated groupings) at the left of the local settings section, you'll see a list of features that may, optionally, be turned on or not.

In the top grouping we have first an item labeled '*Enable auto-setting of Capslock*'. Set this if you want ServiceDesk to manage placing your keyboard into capslock, or not, as appropriate for each context (recommended). Second, is an item that reads '*Include full mailing address with all street name insertions*'. Use this feature if you're planning on generating mailing lists for all your customers. The full mailing addresses (i.e., including zip and state) will need to be in each Callsheet, so as to ultimately be included in a mail list. Third, is an item labeled '*Include full month reference with each appointment insertion*'. Set this to true if you want *all* inserted appointment notations to include a number for both month and day-of-month (as in "12/31 THU") versus simply the day-of-month (as in "31 THU"). Fifth, is an item reading '*Sound periodic chimes to alert of past-due Callsheets*.' Pretty self-explanatory.

The second grouping involves three options related to the DispatchMap.

The first of these, for example, is labeled '*Show mini-map in dispatch map*'. If it takes a lot of panning to view your entire dispatch map, you may want this feature turned on, to help maintain a sense of which portion of the map you are viewing at a given moment (you can turn it on temporarily, to try it, at the very least). Mostly, however, we find the little insert is distracting, and prefer to keep it turned off.

Next is a feature labeled '*Show 5-mile range lines within dispatch map*'. Again, you can certainly turn the feature on to try it, and if you like it, keep it turned on (particularly, if you need with some frequency to know distances from your headquarters). We find there is little distraction in the range lines, and like to keep them turned on.

Finally, in regard to the DispatchMap, there is an option labeled '*Show individual mileage estimates in dispatch map*'. Our purpose: we found some of our dispatchers were not paying sufficient attention to making the routes for each tech as efficient as possible (i.e., minimum travel between jobs). So, we invented a utility that instantly estimates the number of miles for each route (or instantly recalculates with each routing change). This provides a "score," as it were, by which the dispatcher can grade how well he or she has done—and, we hope, gives some motivation for doing better. We find the extra data is somewhat distracting on the map

¹⁶⁰ We recognize that occasionally a person different from the normal user may want to sit temporarily at a station and do some work there. When this happens, you can easily make a temporary change in the Station's Name, without changing this somewhat more permanent designation in the Settings form. To use this temporary *ChangeDesk* option, just press **Alt-K** (think of it as "change-desk"), then specify the temporary user as prompted. ServiceDesk will document any work as being done by that temporary user until you consent to changing back to the regularly-specified person (ServiceDesk will interrupt, once every ten minutes, to invite such a change back).

(roster-wide estimates are given at the top of the map regardless), but may nevertheless be worth the clutter. If wanted, you can choose a middle ground: keep the option turned off, but, when wanting to momentarily view mileage estimates while viewing the DispatchMap, simply press and hold down your keyboard's 'M' key.

Next there's an option all by itself, labeled '*Switch this station into TechMode immediately upon startup*'. The purpose: if you have a station that you've setup pretty much for exclusive use by your technicians (i.e., in reporting on their jobs, etc.), it may be helpful to have ServiceDesk switch into the TechWindow mode immediately upon starting up. By turning 'on' this feature, you can have such a station do so. Of course, you can still switch it out again using the semi-secret key code (see page 116).

Finally, there's a group of three options. These are somewhat special in that, given what they do, there'd be no purpose in having any *one* of them turned-on from more than one station in your system (of course, there is labeling there to let you know this).

The first two of these relate to the WipAlert system as described at page 118. First is a feature labeled '*Send WipAlerts (Callsheet notes re stale JobRecords) to this station*'. Turn this on at one the particular station where you want such alerts sent (probably your primary secretary's desk). Do not turn it on elsewhere. Second, is an item labeled '*WipAlert-Supervisor (informs if job more than 2 days past alert)*'. Turn this on at the one particular station where the owner or supervisor works, assuming this person is separate from the primary operator, and also assuming this person wants to be informed if the ordinary WipAlerts (see above) are not being promptly attended to.

The third item in this last group reads '*Do nightly Archive/Cleanup of files*'. This is where you activate (again, it should be done on one computer only) the Auto-Archive feature, as described beginning at page 209.

After you have all settings as wanted within the local section, click on the 'Save Local Settings' command button. Return to the form to change any feature you want, as often and whenever wanted.

ii. System-Wide Settings

In the first box, labeled '*List of Station Names*', you should (as the label implies) create a listing of the names at each station in your network. Use the same simple name format as for the local station name (i.e., first and last, normal order and upper and lower case, as in "John Smith"), and be sure that after typing each name, you hit Enter to make it insert into the list (if you have just one station, list just the one name).¹⁶¹

In the next box, labeled '*List of Technicians*', you should (again as the label implies) create and maintain a listing of the names for each of your techs. Again, use the same simple name format as described above (i.e., first and last, upper and lower case and in normal order, as in "John Smith"), and make sure you hit Enter to insert the typed name into your list.

One caveat, in respect to these names: ServiceDesk uses initials in several contexts for the names of both techs and station operators. If you have two people with the same initials (i.e., a John Dillon and a

¹⁶¹ While in most cases you'll want to identify each station with a particular person's name (the primary person working at such station), and include each such personal name within this list, in some cases you may want give a station a name that's *not* a person. In our office, for example, we have a separate computer, located in the store room and not otherwise used by any office personnel, which we use as our FileServer. In the SettingsForm, this station is named "Server Station". Similarly, we had another station that, until recently, was used exclusively by our technicians in reporting on their jobs. We gave this station a name, within the SettingsForm, of "Tech Station" (that is until we hired an additional part-time secretary who now uses this station, and whose name it now bears).

Jennifer Dougherty), difficulties could arise. If such happens, you might use a middle name for one of the two persons in lieu of their first, a nick name, or use some similar expedient to avoid identical initials.

To the right, in this purple 'net-wide' section, you'll see another array (similar to those in the 'local' section) of features that may, optionally, be turned either on or off.

First among these is the '*Departmentalize Sales*' feature; check it to true if you want to categorize your sales among different departments (see page 172).

Next is a feature labeled '*Enable Source of Jobs Survey*.' Basically, you'll want to turn this feature 'on' during the particular period (or periods) when you're conducting a survey (see appendix for directions on setting up), and leave it off at all other times. When it's on, your operators will be prompted to ask the caller a series of questions the time they schedule service.

Next is an item labeled '*Mandatory 10-digit Dialing*.' The purpose here: in some areas of the country, even local calls require use of area code when dialing. If this feature is turned 'on', ServiceDesk will add the local area code when auto-dialing any number that does not otherwise indicate an area code. Conversely, if the feature is not turned 'on', yet a number-to-be-dialed shows your own local area code (the latter is itself determined by the voice telephone number listed for your company), ServiceDesk will delete the area code when dialing.

Next is an item labeled '*Allow for divergent sales tax rates depending on locality, type, etc.*' This is for those few clients whose territories encompass multiple jurisdictions that each have different sales tax rates (or which require separate sales tax reports, etc.). For such users, it obviously would not work to specify one particular set of rates for all sales company-wide (see next paragraph). If this is your situation, check this option and read the technical section beginning at page 299.

Finally, in terms of net-wide features that may be turned on or off, there's a feature labeled '*Immediate Call-in Option*.' This is to enable the approach, as described at page 112, where you elect to have your techs call in both on arriving and departing from every job, plus you take a complete PostVisitReport from each with the second call-in.

Following each of these feature settings, in the net-wide section of the Settings form, are several self-explanatory boxes. Some are for inserting your company's current address and telephone numbers (used in some of the documents ServiceDesk creates). One is for your bank account number (ServiceDesk includes this on the deposit slip it prepares for you), and two others for the materials tax and labor tax rate applicable to your sales (ServiceDesk uses these in checking totals on your completed sales, etc., assuming you haven't selected the option described in the last paragraph). Of course, there's also a box for entering the next invoice number in your sequence (defaults to start at invoice number '00001', but should be changed to start at whatever number fits with the sequence you already have in use).

A final box is labeled '*Area codes for which '1' prefix should not be inserted when auto-dialing*.' The reason: there are many cities where, in recent years, area codes have multiplied uncontrollably. In some of these, it is both allowed and preferred (some say it's cheaper) to dial the nearby area codes without first using a '1'. If yours is such an area (or if you have mandatory 10-digit dialing yet the '1' prefix is not necessary when dialing your own area code, you may list all such codes in this box. On the other hand, if you do not have mandatory 10-digit dialing, there is no need to list your own area code here. ServiceDesk will assume (without separate listing here) that any number which has the same area code, as the one listed for your voice telephone number, does not need a '1' first.

Much like in the local settings, when you're done with the System-Wide Settings, click on the command button labeled 'Save System-Wide Settings'.

You can enter the Settings form, as often as wanted, to change any parameters as need or preference may arise. Don't be afraid of it. It's simple and easy.

H. Security Settings and Passwords

As mentioned elsewhere, one of our purposes in ServiceDesk is to enhance the security of your operations by making it impossible, for example, for funds to disappear without you knowing it, or for paid sales to be credited without corresponding funds being collected, and so on. If an unscrupulous operator were granted unrestricted access to your files, obviously, these purposes could be defeated. Thus, at various places and for particular kinds of operation, we require use of a password—a key to the lock, if you will, that should be possessed only by those you trust for the particular operations involved.

Besides security against intended skullduggery, another reason we occasionally require a password is to keep those who may not know precisely what they are doing (but who nevertheless intend well) out of sensitive operations. You may, for example, want to do something in terms of entering sales, payments, or whatever, that's out of the norm, to correct for a previous improper entry, or similar purpose. You know what you're doing, so it's fine to proceed, but if someone without a real understanding attempts the operation, it may produce an unforeseen result. Thus, we require a password to keep those who are well-intentioned, but perhaps not perfectly knowledgeable, from messing things up—which means that, in considering who to entrust with a password, you must account not only for whether you've got absolute confidence in their integrity, but in their competence in dealing with sensitive operations as well.

For passwords to be effective, obviously, their secrecy must be carefully controlled. It's also important that you can specify which locks any particular password is allowed to open, and which not. All these needs are addressed in the *Security Settings* form.

To access this form, you can hit the quick-key combo **Shift-F11**, or (if you prefer mouse action) click on '*File Functions*' in the Main Menu, select '*About ServiceDesk*', then click on the '*Internal Security*' button. If this is the first time you've accessed this form, you'll need to set your Master Password. Think of the Master Password as being like a master key. It can unlock anything. As such, it's probably a password that should be known only to the owner, and perhaps to a most trusted, top-tier manager.

With Master Password out of the way, you can choose which particular areas of ServiceDesk you'd like to have password protected. To show this display, click on the '*Actions*' radio button (Alt-A). The resulting list shows all the areas of ServiceDesk that have password protection available. To enforce password protection on any item, *double-click* to toggle "Password required?" to TRUE. Double-click again to toggle it back to false.

Just as a large building may have a master key that opens all doors (with lesser keys that open only particular doors), the *Security Settings* form allows you to create lesser passwords, that can open only the particular doors you indicate. To enter this function, choose the '*Users*' radio button (Alt-U). Here you may create (and manage) as many lesser passwords as you wish. Each must be listed on its own line in the first column. The idea is that it's like creating a key that can open particular doors, and you'll give that key to the

person who you want to have access to those doors. Make as many keys as you need here—and, certainly, keep track of to whom you are giving them. That is, in fact, what the second column in this display is for: to help you keep track. Generally, that second column does not have any other function (i.e., it's meaningless to ServiceDesk).

After creating any permission key, highlight it and select the '*Permissions*' radio button (Alt-P) to select which items in ServiceDesk this key accesses. You can *double-click* on items here (much as in the 'Actions' display) to toggle whether the password will (or will not) provide access to the indicated action. If a password is compromised you can change the password in the first column without resetting the permissions you granted (Alt-P). Also, you may return (as often as wanted) to modify the permissions that are granted to any particular password (i.e., the “doors” which that password can unlock).

There is one exception to the fact that, generally, the second column in the Users page is only to help you keep track of who has which key. If you have techs logging into TechWindow mode and want make it impossible for another tech to log-in under a particular tech's ID, create a password for the tech and in the second type his entire name (exactly as entered in the settings form). Now, only that password (or the Master one) will work for a tech mode log-in under that tech's ID.

In the *Security Settings* form you can also *reset* your master password (it's good to do this periodically to enhance security). Select the '*Set Master Password*' button at the bottom of the screen. You'll be prompted for your current password before you can enter a new one. You will then re-enter your password for accuracy. ServiceDesk will keep a log of when your master password was changed.

One note: if you feel absolutely no need for the security of a password, you can set (or leave, if you've never otherwise set) the password as nothing (i.e., do not type in anything for your new password, and just hit enter to save the nothing typed). Then, in actual operations, whenever ServiceDesk asks for a password, you can simply hit Enter (easier than actually typing something in), to enter your password of nothing.

Chapter 12

REVIEW AND SYNOPSIS

Because there's a lot of information in the foregoing chapters, it may help to have a summary regarding certain elements that might otherwise seem lost in the miasma of generalities.

A. Command Summary

In this section we provide a written summary of all the various commands available from within ServiceDesk. Each is accompanied with a short explanation of its function and purpose. In general, we think this will be handy as a method of review, to remind yourself of what you learned during reading in the text, and to go through kind of a rehearsal regarding the types of tools that are available.

However, please bear in mind that the format here is *not* the most ideal when you want simply to remind yourself of the particular command that's needed—when you know there's such a command and simply can't remember what it is. For that kind of purpose, there's a much better summary, provided right from *within* ServiceDesk itself. Just click in the MainMenu under the item that's there labeled “Command Summary.” There's a nice categorization, with commands in a compact, easy-to-find format (in contrast to what's provided in this section). In fact, probably you'll want to read what's here provided just once. Subsequently, it will be much easier to use the on-screen summary.

In particular, in some of the contexts where you may need reminding most, you can easily bring up the particular section from the Command Summary that's applicable to the context you're in. From within the DispatchMap, for example, you can simply right-click in any empty spot, and the command summary as there applicable will instantly appear. From a Callsheet, just right-click on the large label area. From the Parts Process form, right-click on its large label area. Or, from the ScheduleList form, right-click on its label area.

In terms of commands generally, it may help to understand that within ServiceDesk they exist at two levels: general and specific.

i. General Commands, as Used to Load Various Forms

General commands can be executed from anywhere (i.e., they work regardless of what particular ServiceDesk venue you happen to be in). Many of these involve the function keys, which correspond with the MainMenu's command buttons. They are used mostly for calling various forms, with the exception of the first, as follows:

F1	<i>CstmrFind</i>	Invokes the CstmrDbase search if you are in any of the suitable Callsheet fields and do not already have previous search results displayed. If you do have previous search results displayed, it moves your focus into the displayed list (also does same in TechInterface-DbaseSearch context). If you're already in the displayed list, it moves you back into the Callsheet. This command also sets up for a CstmrDbase search when executed from the TechInterface form.
F1	<i>CllshtRtrn</i>	If you are in any form other than a Callsheet or the TechInterface form, pressing this key will bring you instantly back to the Callsheets, without unloading the current form. Useful when you're in the middle of something in another form, the phone rings, and you instantly need a Callsheet to begin taking information from the caller.
Ctrl-F1	<i>Settings</i>	Loads the Settings form. This is where you'll set all the specific parameters that pertain to your own operation of ServiceDesk.
Alt-F1	<i>ViewBckps</i>	Allows you to view backup files from each of the various ServiceDesk forms, as though they were the primary files. In this way you can inspect such backups before copying them back over a possibly corrupted primary.
F2	<i>TimeClock</i>	Loads the CheckIn/CheckOut form, from which employees may log into or off of the job. Creates a TimeClock-type record.
Ctrl-F2	<i>CS-Archve</i>	Loads the CallsheetArchive interface, which allows you to view and perform searches among all previously archived Callsheets.
Alt-F2	<i>RtOfEarngs</i>	Loads and displays the RateOfEarnings form, from which you can specify or change the earning basis and rate for each employee in your operation.
F3	<i>AcntsRcvbl</i>	Loads the Accounts Receivable form, from which you can view and edit each of your Accounts Receivable records.
Ctrl-F3	<i>A/R Dnng</i>	Loads the DunningList form, from which you can generate a list for use with your word processor to create form billing reminders for past due accounts, or produce reminders notices for HighVolume-type clients.
Alt-F3	<i>MakeMail</i>	Loads the MakeMailingList form, from which you can generate mailing lists including each of your past customers.
SHIFT-F3	<i>SalesView</i>	Loads the SalesView form, from which you can review your sales record, search for specific records, and edit them.
Alt-F4	<i>PrintClaims</i>	Loads the Finished-Form interface, from which you can edit and transmit a completed invoice/claims document.
F5	<i>SchdMap</i>	Loads the DispatchMap, displaying all of your scheduled jobs in graphic format according to their relative geographic locations (as superimposed over a map of your service territory), sequence, and assigned tech. Allows

for switching assignments between techs, confirming dispatch, auto-dialing to reach tech's at location, etc.

Ctrl-F5	<i>Streets</i>	Loads the StreetList form, used for manually adding new streets to your street list, and for sorting into your list those streets names added in the course of creating jobs. Also copies updated files to other stations.
Alt-F5	<i>SchdArchv</i>	Loads the ScheduleArchive form, which displays and provides for searching among all past items in your Archived ScheduleList.
F6	<i>SchdList</i>	Loads the ScheduleList form, which displays the same data set as the dispatch map, but in a purely textual format that allows sorting, textual editing, archiving, etc.
F7	<i>JobsCrrnt</i>	Loads the JobsCurrent form, which displays the history and current status of every job "in progress" (i.e., of every job that's been created from a Callsheet, and which has not yet been entered into the SalesJournal as having been completed). Allows for record search, auto-dialing of customer, subsequent scheduling, appending notes, editing, and archiving.
Ctrl-F7	<i>JobsArchvd</i>	Loads the JobsArchived form, displaying the history of all previously completed jobs that have been moved into the JobArchived file. Allows for specific record search and for creating CustomerDBase indexes from scratch, should the ongoing indexes for some reason be corrupted. Also shows unrestricted display of each item's History, and allows for certain kinds of searches (e.g., looking for a job by virtue of its street name) not available in other contexts.
Alt-F7	<i>JobRprts</i>	Loads the PostVisitReport form directly, as opposed to the indirect loading that's done via the TechInterface context. You'd use this route if an office person was making the PostVisitReport on behalf of the technician, as opposed to the technician doing it himself.
Shift-F7	<i>JobPerusal</i>	Loads the JobPerusal form, used for reviewing a particular status category of record from the JobsCurrent file.
F8	<i>PartOrders</i>	Loads the PartsProcess form, which displays pertinent information regarding all special-order requests, allows generation of faxed order/inquiries to suppliers, entry and recording of new information as it develops, and for searching of specific items, separate pricing by manager, etc.
Ctrl-F8	<i>PartsArchv</i>	Loads the PartsProcess form in its Archive-viewing mode. This allows viewing of, and searching among, all archived PartsRequest and PartsProcess records.
Alt-F8	<i>PartRqsts</i>	Loads the PartsRequest form, which displays each special order request individually, allowing for expanded notes, archiving of all PartsProcess files, etc.

F9	<i>SlsEnter</i>	Loads the SalesEnter form, providing for entry of all your completed sales.
Ctrl-F9	<i>FundsJrnl</i>	Loads the Funds form, which accesses the file containing a record for every item of money that comes into your business. Allows for specific search or selective display, deposit generation, cross-checking, managing receipt of funds received on accounts payable, confirmation of deposits, etc.
Alt-F9	<i>AppsJrnl</i>	Loads the Applications form, which displays the Applications Journal, a listing that shows how every post-completion payment (i.e., customer mails you a check on a billed job) was applied (i.e., to which invoices). It's also from this form that you can view any particular client's EDA (i.e., Errors and Discrepancies) account.
F10	<i>InventoryControl</i>	Loads the InventoryControl form, used to track and manage actual items of stocking inventory. Facilitates ordering of deficient stock, filling needs on trucks, displaying existing quantities and locations, tracking usage, etc.
Ctrl-F10	<i>MasterPartsPlan</i>	Loads the MasterPartsPlan form, providing the interface for you to create and/or edit your MasterPartsPlan—essentially the list of every type of item (including prices, etc.) you intend to maintain in your stocking inventory.
F11	<i>PrdcRpts</i>	Loads the Reports form, from which you can display or print periodic reports on Sales, Wages or Salary Earned by employees, and Commissions.
Ctrl-F11	<i>JobSource</i>	Loads the Job Source Survey form, which allows you to view results (interim or final) from your job source survey process.
Alt-F11	<i>RedFlag</i>	Loads the SSA form (Special Situations Advisory), while simultaneously inserting text applicable to the customer if done from a context suitable for such (such as a JobRecord, for example).
F12	<i>CstmrDase</i>	Loads the TechInterface form, while simultaneously invoking its CstmrDbase search feature. Thus, after hitting F12 you can immediately begin typing your search target. Once it's found, you may also, from this context, print the entire JobRecord to hard copy.
Ctrl-F12	<i>E-Mail</i>	Loads the TechInterface form while simultaneously invoking its E-Mail function.
Alt-F12	<i>TechIntrfc</i>	Loads the TechInterface form without simultaneously invoking the above functions (CstmrDbase search or E-Mail).

ii. General Commands, as Used for Purposes Other than Loading Forms

Other general commands are focused more toward specific functions than toward loading operational forms, and do not use the function keys. These are as follows:

Ctrl-P	<i>MscInns-Print</i>	Useable from virtually any form, this will print either an image of the form itself, or of text that's printed to the form. If you've placed an image of the
---------------	----------------------	--

entire screen into the Windows Clipboard beforehand (press PrntScrn on your keyboard), it will print that entire screen image.

Alt-X	<i>End program</i>	Terminates operation of ServiceDesk. Don't worry about doing this accidentally, ServiceDesk asks for confirmation before proceeding to shut itself down, and warns if you've got any unsaved Callsheet data. You can, in the alternative, use the MainMenu (select File Functions, then Exit) for shutdown.
Rt-Clk, #	<i>AutoDial</i>	When performed on any telephone number field (Callsheets, Archived Callsheets, JobRecords, Dispatch Map, etc.), causes computer's modem to dial number (for an explanation of alternative methods, see page 301).
Lft-Clk	<i>AutoAlpha</i>	When performed in either the Cstmr or Lctn labels of a Callsheet, allows a user-specified dial-up, including those described by alpha-characters.
Ctrl/Shft-Lft	<i>AlarmSilencer</i>	Each click grants one minute of reprieve the from Callsheet Overdue Alarm (also the mail pending alarm). In a sense this is a general command, for it can be accessed from any form, but it pertains most specifically to Callsheets, so is listed here.
Alt-W	<i>techWindow</i>	Converts the station into a Window/TechMode format, meaning the available interface is entirely different, no longer setup for general use by office personnel, and instead transformed into the limited mode designed for direct interface with Technicians. Remains locked in this mode until the secret exit command is entered (see main text).
Alt-K	<i>ChangeDesk</i>	Allows you to temporarily change the name that a particular station assumes is its operator (as opposed to changing the station name, typically on a more permanent basis, from the Settings form).
Ctrl-S	<i>SlideShow</i>	Initiates the SlideShow demo. Once it's started, you can press the Spacebar to pause, or Esc to terminate the show (actual termination may be delayed for up to a minute).

iii. General Commands, but Applicable Within Specific Form Contexts

Besides these general commands, there are also several for navigating around within almost any form:

PgDn	<i>PageDown</i>	Functional in almost all forms to perform the obvious.
PgUp	<i>PageUp</i>	Functional in almost all forms to perform the obvious.
Ctrl-PgDn	<i>LastPage</i>	Moves to the last page of data in a given form or form context.
Ctrl-PgUp	<i>FirstPage</i>	Moves to the first page of data in a given form or form context.
L-Clk/txt	<i>Edit</i>	Allows editing in most circumstances.

Esc	<i>DprtPrsnt</i>	Extremely useful, handy, and easy. Functional almost everywhere to take yourself from the present context into that preceding, in a manner you'd intuitively expect. Same as clicking on a forms 'Exit' button when so equipped.
Enter	<i>Execute</i>	Please, be sure to use a simple stroke of this key, INSTEAD OF THE MOUSE, in all cases where permitted (almost any where ServiceDesk can assume that such a stroke means you want to proceed). It saves a small amount of effort, and time, each time.
Alt-P	<i>Print</i>	Though mentioned separately in a few specific contexts, the command functions also in many others. If you're in any location where you think you should be able to print something, try it. Even if there's no specific command or option button listed for the purpose, it's possible that form-specific printing is available.

iv. General Commands, as Windows Editing Tools

Also, you should be aware that each of the standard Windows editing functions work within ServiceDesk, as follows:

Ctrl-X	<i>Cut</i>	Removes highlighted text into the Windows clipboard.
Ctrl-C	<i>Copy</i>	Copies highlighted text into the Windows clipboard.
Ctrl-V	<i>Paste</i>	Inserts Windows clipboard contents at cursor location.
Ctrl-Z	<i>Undo</i>	Restores text, in your current box only, to its pre-edit form.

Now moving our focus to Specific commands, you should understand these pertain to any of the particular ServiceDesk forms in which you may happen to be working—meaning they are context sensitive. An identical key sequence may produce one result within one form, and an entirely different result in another (or perhaps no result at all).

v. Specific Commands: Callsheets

The most obvious and extensive of such commands are those pertaining to Callsheets. Several are simply alternatives to clicking on a provided button (remember: you can right click on either of the large label areas in a Callsheet (e.g., the label that reads “Customer Name, Address and Phone Numbers”), and an on-screen command summary will instantly appear):

Alt-S	<i>SwitchDesks</i>	Same as clicking on whichever button is non-selected in the 'DeskAssigned' section of the Callsheet. If the Callsheet is presently assigned elsewhere, this command switches its assignment to your desk. If it's already assigned to your desk, the command switches it to the other desk if there are only two in your network, or presents you with a list of other desks from which to select if there are more than two.
--------------	--------------------	---

Alt-C	<i>makeCurrent</i>	Switches the Callsheet into 'Current' status. Same as clicking on option button of same name.
Alt-H	<i>Hibernate</i>	Loads the Hibernate form, from which you can put the Callsheet to sleep for a specified period. Same as clicking on option button of same name.
Alt-L	<i>deLete</i>	Marks a Callsheet for deletion (which will occur when the Callsheet Archive routine is next run). Same as clicking on button of same name.
Alt-J	<i>Job/Sale</i>	Loads the Create Job/Sale form, which allows you to review and/or change various parameters, then prints a job invoice. The same process readies the Callsheet for movement into the Callsheet Archive (which occurs when the Archiving routine is next run). Same as clicking on button of same name.
Alt-D	<i>othrwsDone</i>	Marks a Callsheet as having completed its purpose through events other than creation of a job/sale. Readies the Callsheet for movement into the CallsheetArchive (which occurs when the Archiving routine is next run). Same as clicking on button of same name.
Alt-M	<i>MoreInfo</i>	Loads the Callsheet's MoreInfo form, which displays any MoreInfo notes previously created in its connection, and inserts an automatic time and date stamp for any note you may presently wish to add. Same as clicking on button of same name.

Other Callsheet functions have no button on the Callsheet itself, and can only be accessed by direct command:

Alt-A	<i>csArchive</i>	Invokes a routine to archive completed Callsheets (i.e., ones marked PrntToInv or othrwsDn), and to remove those marked for deletion.
Alt-R	<i>RcvMssgs</i>	Loads the Communications form, which facilitates electronic downloading of messages from your answering service directly into each item's own awaiting Callsheet.
Alt-U	<i>Undo</i>	Removes all edits created since the last Callsheet save (i.e., restores the text that existed when you last entered the present Callsheet).
Alt-E	<i>Erase</i>	Erases all text from a Callsheet (after first requesting your confirmation, of course).
Alt-F	<i>FlipFlop</i>	Moves the Callsheet's 'LocationInfo' text into the CustomerInfo section, and vice versa.
Alt-T	<i>TelFlpFlp</i>	Moves the telephone number of the box you're in into the other box of the same section, and vice versa.
Alt-O	<i>Originate</i>	Resets a Callsheet's origin info to the present desk, date and time.

Esc	<i>InvokeSave</i>	Useful, when pressed from Callsheet, for invoking an immediate save when, you don't otherwise have any reason for leaving the Callsheet (thus invoking a save naturally).
Ctl-Enter	<i>DnClSht</i>	Moves to the next following Callsheet.
Ctl-Bkspc	<i>UpClSht</i>	Moves to the preceding Callsheet.
Alt-Enter	<i>FndActvDn</i>	Moves to the next following Callsheet that is active to your desk.
Alt-Bckspc	<i>FndActvUp</i>	Moves to the next preceding Callsheet that is active to your desk.
Alt-Q	<i>QuickEntry</i>	Loads the QuickEntries form, from which you can create, edit and enter into your Callsheet the entire data set for frequent customers.
Alt-n	<i>QuickInsert</i>	Inserts QuickEntry templates 1-9 (from the QuickEntries form) directly, without having to load the QuickEntries form.
Alt-l	<i>InsertEmail</i>	Causes system to check Windows clipboard for text from an email dispatch. System will appropriately insert such text to Callsheet if found there.
SpcBar	<i>DisplayItem</i>	Causes the selected item from a displayed CstmrDbase list to show fully, with all JobRecord details. Left-clicking on the list item will produce the same result.
Enter	<i>InsertItem</i>	In a displayed-from-the-Callsheet CstmrDbase list, causes pertinent name and address info, from the selected item, to insert into your Callsheet (right-clicking on the item will produce the same result; insertion is into whichever Callsheet area—Customer or Location—that is active at the time). In a displayed-from-Callsheet street list, it causes insertion of the selected StreetName, together with city, map reference, etc. (left-clicking on the item will produce the same result).
Shift-Enter	<i>ModifiedInsrt</i>	Modifies the CstmrDbase insertion, described above, to include only the name and telephone number (shift-right-click produces the same result). Modifies the StreetList insertion to include street, city and zip, with no map reference (i.e., mailing format; right-click produces the same result).
R-Clk/addr	<i>LocatItem</i>	When performed on either of a Callsheet's address lines, loads DispatchMap and displays corresponding location thereon, with customer's name flagged in red (for an explanation of alternative methods, see page 301).
R-Clk/--	<i>ShowJob</i>	When performed in the CustomerName box of a Callsheet that includes a WipAlert message, will instantly call up the JobsCurrent form loaded with the particular record that caused the WipAlert (alternatively, you may double-left-click directly in the CustomerName box, or hit F1)

Though not commands per se, you should also consider actions that, from within the Callsheet, invoke certain other elements of program execution. When typing in a street address, for example, you are automatically invoking a search for matching street names from your own StreetList, and will prompt a display when appropriate matches are found. Similarly, if your Auto-CstmrDbaseSearch feature is turned on, you'll be invoking similar searches, but within your CstmrDbase, any time you are typing into any of the Callsheet's name, address, or telephone number fields. Finally, as you enter an appointment in the appropriate Callsheet box, you'll invoke the SourceOfJobs survey (providing the Survey feature is turned on), prompting you to ask the customer a series of questions regarding what prompted his or her call.

There are a few forms in ServiceDesk where, by design, we've felt that available screen space could be used much more effectively for display of information than for command or option buttons that remind you of the tools available. This does not mean, of course, that such forms do not offer many such tools. It just means you have to know (in a sense memorize) what those tools are, and which particular key and mouse combinations access them (even in the absence of command and option buttons that remind you of their presence).

vi. Specific Commands: the DispatchMap

More than any others, our DispatchMap is such a form, boasting many, many tools, and yet not a single command or option button (or even a displayed instruction) to remind you of what they are, or of how to use them. It just happens, in the DispatchMap, that you need all the available space to see things more important. Remember, however, that you can right-click on any empty space within the DispatchMap, and instantly the portion of the Command Summary from the MainMenu as applicable to the DispatchMap. Anyway, the more explanatory summary follows:

CrsrKeys	<i>PanMap</i>	The up, down, left and right cursor keys will move your viewing window to different areas of your map, depending on which portions of your map are viewable from a single window.
PgDn	<i>NextDay</i>	Displays the schedule for the particular day which follows whatever day is currently displayed.
PgUp	<i>PreviousDay</i>	Displays the schedule for the day preceding whatever day is currently displayed.
Ctrl-PgDn	<i>DnToPrsnt</i>	Moves immediately to display the present day, providing you've been viewing day's schedules anywhere in the past.
Ctrl-PgUp	<i>UpToPrsnt</i>	Moves immediately to display the present day, providing you've been viewing day's schedules anywhere in the future.
'C'	<i>Calendar</i>	Causes display of the Month Calendar, to facilitate more rapid selection of a day distant from the present.
Alt-P	<i>PrintSchedule</i>	Prints a copy of the displayed day's schedule.
Alt-S	<i>Auto-Sort</i>	Invokes the same Auto-sort routine as may be performed from within the ScheduleList.

'D'	<i>DensityGraph</i>	Causes display of the Appointment Density Graph. Press again or press Esc to remove display.
'P'	<i>Re-load data</i>	Causes a re-load of data for the day presently displayed. This may be useful, for example, if you suspect another user may have made changes and you'd like to update your display to show those changes, particularly before initiating additional changes of your own (which would, in fact, be disallowed if you failed to update first).
Hld-SpcBar	<i>ShowOvrView</i>	Causes a reduced-size map to display, which allows you to see the entire area, with abbreviated job entries, on one screen.
Lft-clk	<i>ScheduleItem</i>	With the mouse pointer on the red-representation of the customer's location in an <i>item-locate</i> situation, a left-click will invoke the scheduling process, causing a list to display with proffered time-frames, etc. This is called On-Map scheduling.
Rt-clk	<i>ScheduleItem</i>	With the mouse pointer on the red-representation of the customer's location in an <i>item-locate</i> situation, a Right-click will cause the same little calendar to come up as does pressing 'C' from your keyboard, except here, after selecting a date, the system will immediately present the Time-Picker for you.
Lft-clk	<i>RemoteDsptch</i>	With the mouse pointer on a the tech's name at top of his list of jobs, this will allow you to print out (or transmit via your computer's internal fax) all the dispatch data regarding the jobs presently assigned to the technician for the day in question.
Sh-Alt/Rt-clk	<i>Alpha-Dispatch</i>	With the mouse pointer on any job's list representation, this action will allow printing or internal fax transmission of that single job, or in the alternative will insert its data into the Windows Clipboard, for subsequent transmission to the tech via alpha-numeric paging.
Lft-clk	<i>AssgnTech</i>	With the mouse pointer on a job's map representation, this action presents a list that allows you to assign the job to a particular tech, or to change its existing assignment. To make it even easier, don't release the mouse button until you've highlighted the tech wanted.
Ctrl/Lft-clk	<i>SwtchStatus</i>	With the mouse pointer on a job's map representation, this changes its AssignmentStatus from 'definite' to 'tentative', or vice versa.
Rt-clk	<i>SwtchStatus</i>	With the mouse pointer on a job's map representation, this changes from which you can re-sequence a job within the particular tech's list.
L-Clk/Drg	<i>Re-Sequence</i>	With the mouse button held down on a job's TechList representation, you may drag the item either up or down in the list, one position at a time. (May also re-sequence from the ScheduleList form: just click on an item, to enclose it in editing boxes, then move it up or down in the list using cursor keys).

Shft-LftClk	<i>CnfrmDsptch</i>	With the mouse pointer on a job's TechList representation, this checks the job off as having actually been given to the tech indicated. A second click reverts status back.
Ctrl-LftClk	<i>TechDone</i>	With the mouse pointer on a job's TechList representation, this checks off the fact that the technician's completed his work there (and presumably is headed for the next location). A second click reverts status back.
Alt-LftClk	<i>CnfrmRprt</i>	With the mouse pointer on a job's TechList representation, this checks off an indication that the tech has made his PostVisitReport on the job. A second click reverts the status back, but typically, this feature will not be used in such context (use it only for corrections, for the check-off occurs automatically, for you, when the technician actually does his PostVisitReport.
Shft-RtClk	<i>ShowText</i>	With mouse pointer on either the job's list or graphic representation, loads the ScheduleList form selects the appointment reference involved and places it in edit mode.
Ctrl-RtClk	<i>ShowJob</i>	With mouse pointer on either the job's list or graphic representation, pulls its JobRecord and displays it for you.
Alt-RtClk	<i>ShowPVR-1</i>	With mouse pointer on either the job's list or graphic representation, loads the PostVisitReport-DialogMethod form, appropriately prepped (providing the 'Immediate call-in' option is selected from the Settings form) to take a PostVisitReport on the job in question.
Sh-Alt/RtClk	<i>ShowPVR-2</i>	With mouse pointer on either the job's list or graphic representation, loads the PostVisitReport-Fill-in-the-BlanksMethod form, appropriately prepped (providing the 'Immediate call-in' option is selected from the Settings form) to take a PostVisitReport on the job in question.
LftClk-Title	<i>Send Schdl</i>	When you left-click on a tech's name (i.e., the underlined title over a tech's list of jobs), allows you to print (via internal fax if wanted) entire schedule info for the tech.
RtClk-Title	<i>ShowOneRoute</i>	When you right-click on a tech's name (i.e., the underlined title over a tech's list of jobs), invokes the DispatchMap's ShowThisTechsRouteOnly mode, which means route-lines for all other techs are hidden, making it much easier (if multiple route-lines overlap) to discern the particular tech's route in question. Hit Esc to exit the mode.

vii. Specific Commands, the PartsProcess Form

The PartsProcess form is another place where you may invoke several specific commands that are not obvious on the face of the form (remember, you can right-click on the label area in this form for a concise on-screen summary). These are as follows:

Rt-Click	<i>New-InfoBand</i>	With the mouse pointer on an existing inquiry/order band (in the right two-thirds), this creates a new daughter band, in which you can enter info regarding an additional inquiry/order pertaining to the same originating parts request.
Rt-Click	<i>ShowRequest</i>	With the mouse pointer on an existing inquiry/order band (in the left one-third), this displays the PartsRequest form, loaded with an unabbreviated display of the originating request itself.
Shft/R-Clk	<i>Delete-InfoBand</i>	Again with the mouse pointer on an existing inquiry/order band, this calls for that band's deletion.
Ctrl/R-Clk	<i>Send/Add Rqst</i>	Adds the clicked-on item into whatever inquiry/order document you are presently preparing for transmission to a supplier. If you have not yet designated any such items, it sets up the process, making the clicked-on item the first to be listed in such document.
Lft-Click	<i>Edit-Item</i>	Encloses the inquiry/order band in editing boxes, allowing you to add, delete or change its data at will.
Lft-Click	<i>ShowRequest</i>	With editing boxes already loaded and your mouse pointer in the InvoiceNumber box, does the same thing as described three items above (thus, you have this function for when editing boxes are displayed, the other for when they are not).
spacebar	<i>RotateStatus</i>	With your cursor in the OrderStatus editing box, this causes the indicated status to rotate between "P&A Only," "Ship if I/S" and "Ship or B/O." With it in the Instruction editing box, it causes the instruction to rotate between "Definite," "Tentative" and "Declined."
"NOW"	<i>InsertDate</i>	When you type these letters in a DateConfirmed or DateReceived box, ServiceDesk will instantly replace them with the actual present date.
"nD"	<i>InsertDate</i>	When you type a number ("n") followed by "D" in a DateExpected box, ServiceDesk will instantly replace your entry with the date pertaining to that number of days hence.

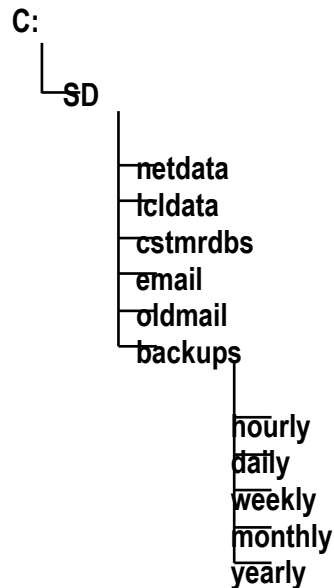
Other specific commands, peculiar to individual forms, are in general indicated contextually within the forms (i.e., the forms will have command or option buttons for specific tasks, which may be executed either by clicking with your mouse on the command or option button, or by pressing the quick key that is indicated on its face). If you are in a form and are not certain whether some specific command is functional there, and you feel the need to use it, again, try it. In all probability, we've felt the need ourselves, and have already programmed the function in. If you find we have not and feel the function would be useful, let us know and we'll probably add it—and ship you a free upgrade in thanks for your assistance.

Finally, don't forget the on-line command summary that's part of the ServiceDesk MainMenu (with portions available contextually as relevant in particular domains). Generally, it will be much easier to use that than to re-read the summary that's here provided. It's right there at the click of a mouse; it's more to the point, and the organization is more obvious than here. Please, don't forget to use it as your primary resource and reminder.

B. Directory and File Structure

In normal operation, you'll never need to know anything about the underlying directory and file structure that ServiceDesk uses. However, for tinkerers who may want to examine or manipulate the files directly, or even for regular operators who'd simply like to know how information is organized, the following is provided.

When you install ServiceDesk on any computer's hard drive, the installation program will create a main directory and sub-directories on that drive as follows:



Now, here is a description of the files contained in each of these directories.

C:\SD. This is the primary ServiceDesk directory, and aside from the sub-directories listed under it, you'll find it contains all the program files, and several ancillary files, some provided, and some which you must create. The first four are program files (that is, they are the actual programs you may run within Windows). These are:

servdesk.exe	This is the primary ServiceDesk program.
sdbackup.exe	This is the SD-Backup program.
sdtools.exe	This is the SD-Tools program.
zips.exe	This is the bonus, little zips utility program.

Also, there are various support, non-program files that ServiceDesk uses for a variety of purposes, as follows.

sdlogo.bmp	A bitmap file for the ServiceDesk logo. If, as the user, you wanted a different logo displayed as ServiceDesk starts up, you could substitute the graphic of your choice, giving it this name and recording it to the c:\sd folder. Alternatively, you could use Microsoft Paint or similar utility and modify the existing logo, as wanted.
servdesk.stl	A list of all the states and their corresponding abbreviations, as needed when inserting full addresses into Callsheets.

The next five files are ones we have custom-created specifically for your business. The first of these is very basic, as it informs ServiceDesk simply of your company's name, and of the unique FileNamePrefix under which it must look for the other custom files in your installation.

servdesk.ini Provides a company name and FileNamePrefix, as needed for initiating program operation.

The remaining four of the provided custom files contain information that's uniquely needed to perform properly in accord with your business's unique geographic territory.

*******.map** This file contains instructions the ServiceDesk program uses to paint the unique on-screen map of your service area. It is written in Ascii format and can be edited—if any tinkers are interested, from any word processing program. In the place of asterisks, the actual file name will contain a letter sequence (i.e., FileNamePrefix) derived from your business name.

*******.str** This file contains the list of street names, coordinates, etc., that's unique to your business. For rapid access, each station keeps its own copy of this file opened at all times while ServiceDesk is running. In the place of asterisks, the actual file name will contain a letter sequence derived from your business name.

*******.blk** This is an additional, supplementary list of street names, except any street (within a single zip area) that's longer than about five blocks will have a separate entry for each five-block segment, and every entry includes numbers showing the block numbers included within it.

*******.cty** This file contains the list of city names and corresponding abbreviations for your unique service area. Again, in place of asterisks, the actual file name will a letter sequence derived from your business name.

The other operating file in this directory is one you'll custom-create yourself.

*******.prg** Substituting the appropriate FileNamePrefix for the asterisks, this name must be given to your Invoice-Format-Instruction, the file that ServiceDesk will use to produce a unique invoice output that is suitable to your own invoice format, using the driver for your printer that is built-in to Windows.

The remaining two files in this directory are not actually part of ServiceDesk, but are provided for your ancillary use, if wanted.

DunningLetter.doc This is a sample document for printing dunning letters from a word processing context. It is configured for Microsoft's Word.

DunningEnvelope.doc This is a sample document, likewise configured for Microsoft's Word.

Note that all these files must be found on each station's own 'c:\sd' directory—for each station looks to its own such directory when needing to use information from these files.

C:\SD\NETDATA. This sub-directory contains the primary ServiceDesk data files (i.e., virtually all the operating records that are specific to your business). By design, it is active only on the particular station you have designated as FileServer; all other stations (and the FileServer itself, if you're using it as a station) look to this sub-directory, on the FileServer drive, for their own use of the many files it contains. This means, of course, that on satellite stations this sub-directory will be empty and unused, while on the FileServer drive, it's

used from all stations, making it rather important indeed. The files stored on the FileServer's drive—in its \SD\NETDATA\ sub-directory—are as follows:¹⁶²

<u>netsttng</u>	Contains all the network-wide setup parameters that you customize for your operation (i.e., those parameters you record from within the 'network' section of Settings form).
<u>cllshftl</u>	Contains the data pertaining to all current Callsheets.
<u>cllshftl.arc</u>	Contains the data pertaining to all archived Callsheets (this file will eventually become very large).
<u>cllshftlg.arc</u>	Contains a directory which keys the binary structure of the above file.
<u>qcefile</u>	Contains QuickEntry data.
<u>adlist</u>	Contains the list of your own yellow page ads that will display during a SourceOfJobs survey.
<u>jobsourc</u>	Contains data showing the results of each completed SourceOfJobs survey.
<u>jbscrtdq</u>	Contains results from the Mini-Survey that is completed during Survey-On periods if the main survey has been dumped.
<u>invlog</u>	Contains the next-to-be-used invoice number.
<u>schdlist</u>	Contains all items in the current schedule, including name, invoice number, date and time scheduled, etc.
<u>schdlist.arc</u>	Contains all items archived from previous schedules.
<u>wipfile</u>	Contains the data pertaining to all current JobRecords.
<u>wipfile.arc</u>	Contains the data pertaining to all archived JobRecords (eventually this file becomes very large).
<u>wipfile.log</u>	Contains a directory which keys the binary structure of the above file.
<u>stcklist</u>	Contains the listing you create of all item types that you intend to maintain as standard stock (an example of a completed such file, as used by an appliance servicer, is provided on your installation CD as 'OtherFls\StckList').
<u>StkAux</u>	Contains most of the information entered the the PartsMoreInfo window, although quantities for each type of specialized truck remain part of the main MasterPartsPlan file.
<u>Trucks</u>	Contains a description of your different specialized truck types, if any, and the listing of each of your trucks.
<u>stckindx</u>	Contains an index which includes each of the alternative part numbers, included in the above file, in sorted order.
<u>spplrlst</u>	Contains names and abbreviations for each of your suppliers.
<u>ilist</u>	Contains a unique entry for every item of your stocking inventory (supply items listed as bulk sets).
<u>ijrnl</u>	Contains data regarding every transfer of a stocked-parts inventory item.
<u>prtsrqst</u>	Contains data regarding every current request for a non-stock item.
<u>prtsprcs</u>	Contains data regarding the process of ordering and acquiring each non-stock item.
<u>prtsrqst.arc</u>	Contains archived data regarding completed non-stock item requests.
<u>prtsrqst.log</u>	Contains a directory which keys the binary structure of the above file.
<u>prtsprcs.arc</u>	Contains archived data regarding completed non-stock item acquisition processes.
<u>prtsprcs.log</u>	Contains a directory which keys the binary structure of the above file.
<u>slsjrnl</u>	Contains records regarding tech, invoice number, name, and totals in various categories for every completed sale.

¹⁶² Those filenames that are underlined here are ones we think are likely to contain still valid, useable information—even if potentially created during initial, fictitious practicing before beginning real ServiceDesk use. Thus, it's only the *not*-underlined files that we think you'll likely want to delete while in the process of purging your initially-created, pretend data (a task you'll likely perform before beginning real ServiceDesk use, see page 271).

slsjrnl.str	A secondary, substantial copy of the normal SalesJournal (see above), only this one is in Ascii, comma-delimited format, so as to be accessible for separate client use and manipulation from within any spreadsheet program.
arfile	Contains data regarding every item of Accounts Receivable.
fndsjrnl	Contains data regarding every receipt and deposit of money.
fndslog	Contains data enabling an internal quasi-archiving of older data within the above file.
appsjrnl	Contains record of each check received in payment on billed jobs, showing which invoices check was applied to.
*****.eda	With name you provide substituting for asterisks, contains a record, for a particular customer, of surpluses and deficiencies in payment, as applied by you at large to the customer's particular (i.e., this file) Errors and Discrepancies account.
timelog.**	Contains daily TimeClock entries for whichever person whose initials are found in place of the two asterisks.
<u>erngsrt</u>	Contains data you've input, from the Rate Of Earnings form, regarding the compensation method and rate for each person on staff
addstrts	Contains a list of streets that have been manually entered (because missing from main StreetList), whether via the job-creation process from Callsheets, or directly from the Streets form, and designated for addition into the StreetList.
POList	Contains a list of P.O. Numbers from jobs you've most recently created. Builds, as you create jobs, to contain a maximum of the most recent 500 such numbers, then is automatically trimmed down to the most recent 300. Used, during the invoice-printing/job-creating process to assure you're not, and to warn you against, using the same P.O. Number more than once.
<u>DeptList</u>	Contains a list showing the complete name for each of the separate departments you may have created if having enabled the Departmentalize option.
DeptAsnd	Contains a seriatim listing, showing invoice number and department-assigned, for every job created while the Departmentalize option is enabled.
<u>TimeFrms.Lst</u>	You may create this file if wanting a different set of time or time-frame options, as offered in the lists of time-frames appearing in the Callsheet, ScheduleList and DispatchMap, to assist in scheduling.
<u>Generic.Frm</u>	This file, along with <u>Custom.Frm</u> and <u>Narda.Frm</u> , define the format for each of the respective forms from which may produce completed invoice/claims documents.

As you can see, these are your really important files, mostly ones that are continually changing on a daily basis as you conduct your work. They are, sensibly therefore, the very files your SD-Backup program is designed to make regular copies of. And, if you run any other backup routine, they are the files you should, at a minimum, have it setup to regularly copy. It's simple: just setup to backup the entire "C:\SD\NETDATA" directory from the FileServer drive.

C:\SD\LCLDATA. This directory is used primarily for storing various temporary and backup files that are created during several different sorting and archiving routines. By storing these files locally on whichever station is performing the routine, the routine itself is often speeded up, and it frees the FileServer from the momentarily intense processing burdens which would otherwise slow down its service to other users. Also, and incidentally, this provides extra backups in case the FileServer happens to fail. You may, nevertheless, delete files on this sub-directory that have a 'bak' or 'tmp' extension—any time you like. Aside from such files, this sub-directory also contains the following:

lclstng	Contains the settings, as created in the 'local' section of the Settings form, for this station's local operating parameters.
----------------	---

bckuploc	Contains the directory and file path last used when using the ViewBackups utility. Makes it more convenient when you next use the utility, allowing ServiceDesk to volunteer the location for you.
dnngloc	Contains the directory and file name last used when creating a Data document for the creation of billing reminders (i.e., statements or dunning letters). Again, makes it more convenient when you next use the utility, allowing ServiceDesk to volunteer the location for you.
MailLoc	Contains the directory and file name last used when creating a MailingList for creation of promotional/thank you letters. Again, makes it more convenient when you next use the utility, allowing ServiceDesk to volunteer the location for you.
Printers	Contains a list that remembers the particular printer you last specified from each printing context. Before printing, each printing context consults this list, and offers the same printer if one was previously specified.

C:\SD\CSTMRDBS. This directory contains the CstmrDbase indexes and list, which (to review from the main text) reference entries in both the JobsCurrent and JobsArchived files. For the sake of rapid access, each station maintains its own current opened copies of these files, allowing each instant access for the frequent searches that are involved when your Auto-CstmrDbase Search feature is turned on. These files are as follows:

cmbnmndx	An index based on sorted customer names.
cmbadndx	An index based on sorted addresses.
cmdtindx	An index based on sorted telephone numbers.
cstmrdb	A listing that sequentially matches entries in: first the JobsArchived file, then the JobsCurrent file, including identifying snippets from each. It is actually items in this file that the indexes reference, while this file in turn references records from the underlying job files.

C:\SD\EMAIL. Contains a separate file for every item of current E-mail. Contains no files if there are no current items of mail.

C:\SD\OLDMAIL. Contains a copy of each item of E-Mail formerly used and discarded. Each item exists as it's own separate file, named for the date on which it was discarded, followed by a sequential extension. Items may be loaded and read from any word processing program.

C:\SD\BACKUPS. Contains nothing except within its sub-directories, and these will contain nothing except on a station on which you've elected to run SD-Backup. On such a station, the SD-Backup program will, with commencement of running, examine its own "C:\SD\BACKUPS" sub-directories. If it finds no backups in its yearly sub-directory, or that existing backups there are non-current (i.e., they are from a preceding year), it will copy all files from the FileServer's "C:\SD\NETDATA" directory into its own "C:\SD\BACKUPS\YEARLY" sub-directory. And so on for the monthly, weekly, daily, and hourly categories. With each such backup of files from the FileServer, it will also create a tiny log file (within the relevant sub-directory) indicating when such backup was performed.

C. Final Setup: Items to Check-Off

Following is a listing of tasks you should perform to assure ServiceDesk operation is physically ready and optimized for full and effective operation.

As a first area of consideration, we'd like to suggest you consider whether certain *settings within Windows* are optimized for ServiceDesk operation. In particular, there are three matters you should review:

First, since ServiceDesk is an application you may be starting with some frequency (unless you simply leave it running all the time), we suggest you move a copy of its icon directly into your Windows Desktop (from whence you may start it more quickly simply by clicking on the icon, rather than by having to go through the Windows Start Menu, then Programs, then the RossWare Computing selection, and so on). There are several ways to do this. Most easily, locate the "ServiceDesk for . . ." listing/icon in the Rossware Program group (i.e., as though you were going to double-click on it to start the program), but instead of double-clicking, right-click instead. In the small menu that's presented, one of the options will be to "Create shortcut." Click on this, and Windows will insert a second listing/icon into the menu group. Now simply, drag this out of the menu listing section and onto the desktop. It's that simple, basically, to create a desktop shortcut for any program.

Second, there's a particular configuration for the Windows TaskBar that we think works best, not merely in conjunction with ServiceDesk, but in general. Basically, this is to have it programmed to hide itself when not in use, and to always be by-mouse accessible even if you have some program in operation that would otherwise cover it. To set it up this way (again, there are methods other than what we'll describe, should you prefer to use them), simply right-click on any unused region of the TaskBar, then select *Properties*, then the *TaskBar Options* tab. Now, in the form presented, see that you have both *Always on top* and *Auto-Hide* checked to true. Also, it's best to have the TaskBar situated on the bottom of your screen (you can drag it to any of the four screen edges), because the *ServiceDesk* screen allows a small space for it to remain visible there when it's reduced to it's (mostly) hidden state. As so setup, you'll find you can simply drag your mouse to the bottom of your screen whenever you wish to use the TaskBar. Even while ServiceDesk is running, the TaskBar will immediately appear, allowing you to select from its options (including any other programs that may be running). At the same time, the TaskBar will not significantly cover any ServiceDesk space.¹⁶³

Third, if you're familiar with Windows, you know you can change and select from among many different color schemes (right-click anywhere on the desktop, select "Properties," then choose the "Appearance" tab). We want to give you one caveat in regard to selecting these schemes. Some create title bar text that is larger than standard. This makes the title bar taller, and results in a portion of your ServiceDesk Callsheets failing to fit on the screen. If you notice this has happened, it's because you've selected such a scheme. To correct, simply change back to a scheme that uses ordinary size text in the title bar.

In regard to *ServiceDesk itself*, almost nothing (aside from minimal installation, etc.) is required for the *most* basic operation, though substantial setup effort is required for many, beyond-most-basic the features. At

¹⁶³ As a connected Windows hint, many users are not aware of how easy it is to switch from one application to another, while each is running (and even if one or more cover the entire screen, as does ServiceDesk). One method is to use the TaskBar, as described in the text preceding this note (i.e., just pull your mouse pointer to the bottom of the screen, the TaskBar pops up, then you click on any other running application you want to switch to). Even more easy and convenient, however (because you don't have to use the mouse), is to use the simple **Alt-Tab** method. If you hold the **Alt** key down while repeatedly hitting **Tab**, you can scroll through all running applications and simply stop on the one you want. It's very easy.

any rate, we'll describe various of the elements involved, in something approaching a likely probable sequence.

Fourth, for any function at all, it is essential that at least some matters in the Settings Form be filled-in (see page 223). Most likely, you'll want to complete it somewhat thoroughly from the very start. Remember that the 'Net-Wide' section only needs filled-in from one station (the same settings for that section will appear at every other). The 'Local' section needs to be separately filled-in at each. Also please consider that for beginning use it's most sensible to leave optional, extra-work type features (such as the SourceOfJobs survey, for example, turned off). There is so much new stuff coming at you in the learning stage, it's simply not wise to complicate things more than necessary at such a point.

Fifth, if you've created fictional data (i.e., pretend Callsheets, JobRecords, SalesRecords, etc.) as part of your testing and learning process, you may want to get rid of that stuff before starting entry of genuine business information. The relevant files are all in the '**SD\NETDATA**' folder of whichever drive you are using as FileServer. We suggest using DOS or Windows Explorer (whichever you enjoy most) to see what files are in this folder and to delete whatever seems appropriate. It will make most sense, in this regard, to delete those files that contain nothing but practice information, while retaining those that contain *setup* information you've already worked to input and will be continuing to use (please note that we might have created an automatic utility for this purpose, except the decision to delete or not can vary depending on circumstance).¹⁶⁴

Sixth, be sure to setup the *SD-Backup* utility on some drive or station other than your FileServer. Really, it can save your life, and to set it up all you have to do, essentially, is run it—or better yet put it in the Windows Start menu on whichever station you're using it from. Thus it will start itself automatically each time that station boots up (see page 215).

Seventh, before having ServiceDesk print up your service tickets, you'll have to have a number of things ready. This includes having appropriate invoices in your possession, a machine that will appropriately type information onto them, and having created a Invoice-Format-Instruction file via which ServiceDesk will comprehend the wanted format (see page 267). The last task, incidentally, is very easy, and hopefully you'll already possess appropriate invoices and printer (please note this step is not absolutely essential; you can easily run everything else in ServiceDesk and simply not have it print up your invoices).

Eighth, you should setup QuickEntry forms for your home-warranty, OEM-warranty or other very frequent, institutional-type clients (see page 57).

Ninth, if you want a different list of time-frames offered to assist in scheduling (other than the standard, default list), you'll need to set that up too (see note 67 on page 96).

Tenth, if you want drop-down lists to appear in the Item(s) Type and Item(s) Make boxes of the Callsheet, you'll have to do some advance setting up in the UnitInfo system (see page 201; but note this is typically delayed until reaching Learning Mode 6, see page 262).

¹⁶⁴To help you distinguish between files that contain what's probably mere practice data (such as pretend Callsheets, for example), versus those that contain still useable setup information (such as data in the Settings form, for example), we've provided a mode of guidance in another portion of this manual. It's in the pages that explicitly list and describe each of the files potentially in that '**SD\NETDATA**' folder (which is, incidentally, the only folder that contains items you'll likely want to delete). You'll find this section beginning at page 267. There you'll see that certain of the listed file names have been underlined. These are the ones we think you'll likely want to save. Others we think you'll more likely want to delete, though in all cases the decision is up to you (please note you are not likely to find all the listed files actually in your folder at this point, as many will not likely have yet been created).

Eleventh, prior to making use of ServiceDesk's InventoryControl features, you'll need to create your own company's MasterPartsPlan, using the MasterPartsPlan form, and you'll need to log-in beginning inventory (see page 145).

Twelfth, prior to attempting to download messages electronically from your answering service (a substantial convenience, but hardly essential), be sure they have the technical ability and that they have appropriately adjusted their format (see page 266).

Thirteenth, prior to invoking any SourceOfJobs survey (so important as to be a travesty if you didn't use it, though you can certainly wait many months before doing so), you must be sure to create an appropriate file regarding each of your outstanding Yellow Page ads (see page 281).

Fourteenth, you should turn-on the AutoArchive feature from one station (see page 209), perhaps also the WipAlert feature (see page 118), and so on.

Of course, there are other many features that may involve some degree of setting-up before use, but the above is a pretty good synopsis of most such matters at least.

Chapter 13

A BEGINNER'S GUIDE TO EASY, STEP-BY-STEP IMPLEMENTATION

As we've stated elsewhere, ServiceDesk does so much that if you attempt to comprehend it instantly, you're likely to meet with extreme frustration. Of course, for any involved subject, proper learning does not work that way. It's a step-by-step process, with a student mastering first one element of functionality, then another, so that each builds on another in a gradual process. Plus, as new elements are added in this fashion, there's already a conceptual framework into which new elements can then be fit for full and proper understanding.

At least that's how it should work, but in ServiceDesk we've had a problem in that all its parts are so interconnected (everything ties together) that it becomes difficult to separate one element from another. Thus, in the past we've essentially forced new users into a kind of "sink or swim" predicament, throwing at least most of the system toward them in one fell swoop. We realized this made for a daunting gear-up process, but for a long time we'd simply not yet figured a better way.

Well, finally we did. What we've done is to modify the ServiceDesk program code in a manner that makes it accommodate usage of selected parts only. In other words, you're allowed to set it into any of several "learning" modes. In reaction, it will essentially sever the links between what you're learning and other elements that you've not yet conquered. Thus, it becomes practical for you to use and master limited elements without being forced, simultaneously, to use and master the whole set of features.

The heart of this new system is a simple little window in the *Settings* form. To access this window, go to the Settings form (Ctrl-F1) and look toward the bottom. There you'll see a button labeled "Learning Steps." Click on this button and the *LearningSteps* window will appear. As you'll see, at the top there's a caption reading "*Operate in mode requiring only . . .* ", then there's a numbered listing of items describing eight, increasingly advanced levels of functionality (the last being unrestricted, no limitations).

As you'll note, the system defaults (upon initial ServiceDesk installation) to mode 8 (unrestricted, no limitations). This is so that, as you're doing your initial orientation in the first four chapters of this manual, you'll have full access to all the functions and links between, to see how they work, particularly when all interconnected. The notion, however, is that after that general orientation process, you'll skip ahead to this chapter, follow its instructions for setting-up Learning Mode 1, then immediately begin using it—in actual business operation. Naturally, you'll continue at that level for some time (perhaps a few days) until you feel perfectly comfortable and proficient with it, then advance to Learning Mode 2, and so on—until you've plowed all the way into full-fledged, unrestricted operation.

A big advantage of this approach, by the way, is it allows you to start using the very basics all but immediately, and free of any entanglements with the more advanced systems. Previously, we had clients who delayed implementation for weeks or even months, until they felt they'd mastered an understanding of virtually everything. Other clients would dive right in to real-life implementation, but sometimes without adequate preparation and understanding, only to find themselves confused by various alarm systems complaining that this matter or that was not being adequately addressed. Most eventually made it through, but some foundered.

Now, if you'll but follow the steps as recommended here, we can all but guarantee it will be comparatively easy. In the following sections, we'll describe each of the steps, explain what's involved, and provide hints on navigating your way through.

Learning Mode 1 (Callsheet Use and Printing of Invoices)

The basic notion for this first mode is that, within even hours of receiving your ServiceDesk package, you'll begin using the Callsheets to manage your telephone work. Additionally (and assuming you're not converting from a previous software system), you'll use them to create the work-order/service-tickets for your technicians to take on the job.

The basic notion is that, at this initial stage, this will be the full extent of your business's involvement with ServiceDesk. You'll not be using it to actually manage jobs once they are created. You'll not yet be using it to manage your schedule and dispatch operations, or inventory control, parts ordering, funds control, sales recording, and so on. All those functions will, for the time-being, continue to be operated via your old methods.

Here is a list of tasks we suggest you go through, checking off each preparatory to beginning this first stage of limited operation:

- (a) See that you've completed the physical check-off items for ServiceDesk itself, as described in the preceding section, but only through the *ninth* item there listed;
- (b) As suggested in the introduction (and for a broad conceptual understanding), see that you've completed a careful reading through Chapter 4 of this manual;
- (c) If you've purchased the video tutorials, now is the time to watch Lesson # 2 therein;
- (d) Read all of Chapter 5 ("Call Management"), and do it especially carefully, since this chapter concerns the very processes you're about to employ (bear in mind, though, it describes Callsheet functions in conjunction with full-fledged, unrestricted operation; for Learning Mode 1 some functions are adjusted, as described in the following paragraphs);
- (e) Finally, go to the Settings form, click on the 'Learning Steps' button, and set the system into Learning Mode 1.

With this accomplished, please begin immediately to use ServiceDesk for the purpose of managing your calls (and printing up your service tickets, if applicable). Do it with the very next telephone ring. There's no need to put it off. Throw pen and pad away (at least so far as used these functions). Type the call's relevant information (whether constituting a service order or not) into a Callsheet. Then, when the next call comes in, type into a new one, and so on. For each instance where the incoming call does involve a service

order, type-in all the relevant order information, then click on the Callsheet's *Job/Sale* button to print up a service-ticket for the job.

Operationally, the act of placing ServiceDesk into *Learning Mode 1* has the consequence of limiting your choices in the Create Job/Sale form. Normally, the default choice there is to *Print the Invoice* and simultaneously *Create a JobRecord* (and, incidentally, to insert an appointment into the ScheduleList). When set to *Learning Mode 1*, however, ServiceDesk allows you simply to print the invoice, without creating either a JobRecord or ScheduleList entry. Thus, with those two particular records not being created as part of the process, there's no need for you to worry about managing them from within the ServiceDesk context.¹⁶⁵

Please note specifically that while you may, at this stage, use the DispatchMap to review a customer's location while taking their order (using the Callsheet's *ItemLocate* feature, see page 44), there will not otherwise be any use for the DispatchMap, because you'll not yet be creating appointment entries in the ScheduleList. And there will not as yet be any use for the JobsCurrent or JobsArchived forms (because you're not yet creating JobRecords), or even of the CustomerDbase system (which is based on those records). And the drop-down lists for the Callsheet's Item(s) Type and Item(s) Make boxes will not yet be operative. In short, your ServiceDesk usage will be limited to precisely what we've described, and no more.

Of course, this means you'll still be using your old systems for all tasks except such call management and printing-up of your service tickets. We realize such a mix of systems may seem awkward, but the only alternatives are to either make no transition at all, or else to do it *all at once*. Though slightly awkward, we think a transitional period of mixed methodologies will usually be most sensible.

We suggest you continue in *Learning Mode 1* for a few days, as long as it takes to feel comfortable in this environment.

There is no need, incidentally, to continue reading in this chapter—until after you've practiced for a while in Learning Mode 1, and are in fact ready to move on to the next mode. Please (in other words), delay reading the instructions in regard to each following learning mode until you are in fact ready to move onward to it.

Learning Mode 2 (Adding-in Schedule and Dispatch Management)

As stated, when you invoke the Job/Sale process from a Callsheet in the context of Learning Mode 1, the system will not (as it otherwise normally would) insert an appointment reference into the ScheduleList for you. The big change in this mode is that now it will do exactly that—meaning that now we'll expect you to begin using, and successfully managing, the various tools ServiceDesk provides for schedule and dispatch management. To prepare for this stage of operation, we recommend the following:

- (a) If you've purchased the video tutorials, now is the time to watch Lesson 3 therein;
- (b) Now please carefully read all of Chapter 6 ("Schedule and Dispatch Management"). Again, bear in mind that some of the described operations are modified by the limitations of this Learning Mode (as

¹⁶⁵ If wanted, you may choose to limit your involvement at this stage even further. If you want to delay in using the system to print-up your invoices, for example, that's perfectly fine. And if you want to delay in having ServiceDesk assign invoice numbers to your jobs (even if otherwise printing invoices), that's fine too (at later stages invoice numbers will be mandatory, since it's via assigned InvoiceNumbers that ServiceDesk uniquely identifies each job).

particularly described in following paragraphs), but please try absorb the matters described there, regardless;

- (c) Go to the Settings form and assure that the '*Next Invoice Number*' value is appropriately set (this is *necessary* now, since the system is going to be using appointment references within the ScheduleList, and each such reference needs to be identified by an invoice number); and
- (d) Also in the Settings form, click on the 'Learning Steps' button, and set the system into Learning Mode 2.

With this accomplished, continue using the Callsheets just as you did in Learning Mode 1. Only now, when you invoke the Job/Sale process from a Callsheet, you'll notice that ServiceDesk does something besides merely printing-up a service ticket for you. Assuming you have an appointment reference in the Callsheet's '*Date & Time*' box, it will also (as part of that process) insert a corresponding appointment reference into the ScheduleList (though still not creating a JobRecord).

This means that now, after you've taken in a few scheduled service orders via your Callsheets and likewise invoked the Job/Sale process from each, you'll find that your ScheduleList (accessed by pressing F6) suddenly contains a number of appointment entries. And if you press F5 to view your DispatchMap, you'll see each of the appointments are graphically displayed, in the correct positions. And now, we'll expect you to manage these (along with others that enter the list), control your dispatching, and so on, via the processes described in Chapter 6.

There are, of course, a few differences from normal, unrestricted operation in this Learning Mode 2. Most significant is that your appointment references (within the ScheduleList and as displayed on the DispatchMap) will not as yet have any within-ServiceDesk JobRecords to reference (because, obviously, you're not yet creating those). This has consequences such as: (1) If you try to invoke a ShowJob action from the DispatchMap (see page 84), the system will report it cannot find the job (for, in fact, *no* JobRecords yet exist); (2) The system will refrain from its normal practice of retaining appointment references until PostVisitReports are made in their connection (see page 101); since we're not yet managing actual JobRecords within ServiceDesk, there'd be no purpose in these; (3) When making new appointments in connection with a job that was previously created from a Callsheet (e.g., the technician was already there on the initial appointment, he ordered a part, it's come in, you've called and rescheduled the customer, and now need to get that second appointment into the ScheduleList), such appointment references will have to be added into the ScheduleList by manual means (see page 95), since as yet we have no JobRecords from which such insertion could be invoked more automatically (*ibid*); (4) Your various actions within the Schedule and Dispatch system will not have a historical record, since there are as yet no JobRecords for such a history to record to; and (5) There will still be no CustomerDbase to work with, since again that feature is based on ServiceDesk JobRecords, which do not yet exist.

Operationally, this mode is perhaps slightly more awkward than the first one, because we're not yet involving ServiceDesk JobRecords within our setup, and yet it's normally intended for *so much* interaction to occur between these and the ScheduleList. Even so, we think it will be beneficial for you to operate at this stage for at least a few days, until building at least a reasonable competence therein (the involvement of JobRecords, as occurs in the next Learning Mode, introduces a lot of additional expectations, and that's why we wanted to keep this mode separate).

Please work each day, while within this mode, to assure that *every time* an appointment is made (regardless of the context), there's an appointment reference inserted to the ScheduleList in its regard. Remember that when you're first taking in a job order via a Callsheet (and invoking its Job/Sale process),

ServiceDesk will make this insertion for you. But for subsequent appointments on that job, you'll have to invoke the insertion manually (typing in each of the fields) from within the ScheduleList form itself, by clicking on its 'New Item' button (bear in mind this will be easier in all subsequent stages).

In consequence of fulfilling this duty, you should find that your DispatchMap constantly reflects what's actually scheduled with all of your customers (make sure that it does, and correct if there are discrepancies). Thus you can use the features described in Chapter 6 to facilitate dispatch and related processes each and every day. And now you can schedule intelligently while taking in new job orders via the Callsheets (using the ItemLocate feature to compare location with other scheduled jobs, the techs they're assigned to, etc.). And so on.

As in Learning Mode 1, we suggest that at some point in this stage, probably when about midway through, you review the applicable command summaries as provided on the ServiceDesk MainMenu bar. For this Learning Mode 2, click on "Command Summary" then choose "DispatchMap Controls" for one review, and "ScheduleList Controls" for another.

At any rate, when you feel you've reasonably mastered (and are practiced in actual usage of) the basic operations involved with Scheduling and Dispatch (excepting, of course, those that cannot yet be done, lacking attached JobRecords), it's time to move onto Learning Mode 3.

Learning Mode 3 (Adding-in JobRecords, and their associated management)

Hooray! Finally we're ready to address the final anchor of ServiceDesk's foundational triad. Having mastered call-taking operations (which incidentally involve the *initiation* of jobs) and schedule management (which also obviously involves jobs, though we've managed it to date with those jobs existing completely external to ServiceDesk), we're now going to introduce the underlying context within ServiceDesk by which each and every job is represented specifically *as such*. To prepare for this stage of operation, please do the following:

- (a) If you've purchased the video tutorials, now it's time to watch Lesson 3 therein;
- (b) Carefully read all of Section A ("The WorkInProgress System") within Chapter 7. As always bear in mind that some of the described features may not yet be applicable, because we're still limiting things in connection with the particular learning stage we're in;
- (c) Go to the Settings form, click on the 'Learning Steps' button, and set the system into Learning Mode 3; and
- (d) If you've not previously done so, we also suggest that while you're in the Settings form you now turn 'on' the feature labeled "Do nightly Archive/Cleanup of files" (from one station only), along with the feature labeled "Send WipAlerts" (also from just one station please).

With this done, you'll find something rather different occurring (at least as compared to previous Learning Modes) when you invoke the Job/Sale process from any Callsheet on which you've written a new service order. Now, in addition to being ready to print-up the service ticket and insert an appointment reference to the ScheduleList for you, the system also defaults to a mode where it's ready, as a simultaneous part of that one simple Job/Sale event, to create an actual *JobRecord* (press F7 after invoking the process when now in this mode, and you'll see that now you've got your first, real-life JobRecord now in the system).

It is the JobRecord, of course, that's the primary instrument in ServiceDesk by which each job is managed, and into which the system records each of the events that occur in connection with a particular the job. Thus it's a milestone, indeed, to now be at the stage where we're using these electronic job representations. It opens up all kinds of possibilities.

Those possibilities are primarily discussed, of course, in the section we've instructed you to read for this learning stage, and to avoid redundancy we'll not elaborate too extensively here. Suffice it to say (as, for the most part, you'll see upon such reading) that now you can do things such as: (1) Inserting secondary (i.e., after the job was created) appointment references to the ScheduleList much more automatically, using procedures that can be initiated from the now-existing JobRecord, including ItemLocate and On-Map scheduling, much as can be done from a Callsheet when *initiating* a job; (2) Doing a ShowJob action from the DispatchMap; (3) Reviewing the details of a job from within ServiceDesk; (4) Manually recording details of job performance within its History section; (5) Having the system automatically record, within the JobRecord's History section, various scheduling and dispatch events as they are performed; and (6) Finally, the CustomerDbase system will begin to become operative as you develop an accumulated history/set of JobRecords to form its underlying basis.

Of course, with the system of JobRecords now in play, along with the new benefits there will also be some new operational duties. Foremost among these is that now the system will expect: (1) For every job-based appointment that's inserted to the ScheduleList, someone will dutifully make a PostVisitReport in its connection (see page 110); and (2) You must now make sure there is appropriate activity, in connection with each JobRecord, to indicate adequate work is being done to achieve its completion (otherwise, you'll be hounded by WipAlerts; see page 118). Specifically, since many of the within-ServiceDesk auxiliary and ancillary processes are not yet in play (at this learning stage), it happens that many of the activities that would otherwise record automatically to a JobRecord (such as checking-in ordered parts, for example), will not yet do so. Thus, for this learning stage you may find there's a greater burden (than there will be later) to manually document (within a job's History section) various things you do (such as ordering parts, etc.) to work the job toward completion.

Speaking of which, we'll explain here that operationally ServiceDesk makes just a few internal allowances for this Learning Mode 3. Specifically: (1) Whereas normally it will not retire a JobRecord out of the CurrentJobs file and into the JobsArchived space until after it has been recorded to the SalesJournal (and its "status" marked accordingly), for this learning mode and the next one (i.e., until Learning Mode 5, where we introduce *Sales Reporting*), the system will "archive out" any JobRecord when it's simply been set to "*Job Completed*" status.

This means, incidentally, that operating in this mode there's another special duty you may sometimes face. Typically, JobRecords will be set into "Job Completed" status in consequence of a query answered ("Is the job done?") during the PostVisitReport, but sometimes circumstances will conspire otherwise. It may happen, for example, that the technician returns with a *possible* part order, depending on what the price proves to be after research is done—and the PostVisitReport is made with the matter still hanging, so the query ("Is the job done?") is answered negatively at such time. When the price is determined, the customer declines to proceed, so now the job needs closed out. In a later stage (after we've gone on to Learning Mode 5 and introduced Sales Reporting), the procedure will be to simply make an entry in the JobRecord's History (indicating that the customer declined), then record its completion to the Sales Journal. But since we're not yet invoking any Sales-Entry procedure, it will at this stage be necessary (in like or similar circumstances) to make the History entry *plus* manually change the JobRecord's status into "Completed" mode (otherwise, again, you'll eventually be hounded by WipAlerts asking you to either show activity on the job or complete it).

Another internal allowance ServiceDesk makes for this stage is that, when you're making PostVisitReports, it will refrain from querying you in regard to functions (such as funds collected, parts needing ordered, parts used from stock, etc.) that you've not yet implemented. In fact, queries are limited in *each* Learning Mode to just those that are relevant to the functions that exist at such point. Thus new queries are added, in regard to each such function, as you implement the corresponding Learning Mode in steps yet to come.

Learning Mode 4 (all the above, plus Funds Management)

From here on out, each new stage should be comparatively simple and straightforward. The reason is because now, rather than a step-by-step adding-in of what are essentially the system's three interdependent *roots*, we'll instead be doing a step-by-step adding-in of what are instead more like *branches* on the tree, which is obviously a simpler process (though these are also highly interwoven, the addition of a branch is just plain easier).

At any rate, to prepare for this stage (and if you've purchased the video tutorials), it's time now to watch Lesson # 4. That lesson will, in fact, pertain to this and the remainder of the learning modes, but since that lesson in the tutorials is not distinctly divided into corresponding subdivisions, it will only make sense to watch the whole lesson all at once.

Regardless, it's time at this point to carefully read Section D in Chapter 7 ("Tracking and Depositing Funds"), then go to the Settings form and switch to Learning Mode 4.

Having done as applicable from the above, simply follow the procedures (at least to the extent the present stage allows) as outlined in that manual section. The idea, obviously, is that now you'll be using the built-in ServiceDesk system to document and manage each item of money that's taken in by your business (including the making of bank deposits). It's a terrific system, and you're almost guaranteed to love it very shortly.

The one limitation, for this stage, is that you'll not yet be able to use the normally-intended system for reporting on those funds that are received *after* a job has been completed and recorded to the Sales Journal (i.e., checks received on billed jobs, see page 157). The reason is because the *normal* method depends on having an AccountsReceivable record, in force, that such items of money can be applied toward—yet such records will not be created until the next learning stage, in conjunction with reporting on completed sales. For such reason, when such items of money come in, during this stage, you'll need to use the direct, manual method of insertion, as provided in the Funds form (i.e., use its 'Add Item' button). This is important, for from now on you'll want to have all such checks on-record within your FundsJournal for at least the purpose of preparing accurate deposits. There will be no recorded application of the check to a specific job, of course (or to any job's AccountsReceivable record), but that's okay, for ServiceDesk has no such records at this point regardless.

Learning Mode 5 (all the above, plus Sales Reporting and A/R Management)

This step is very self-explanatory. In preparation, please carefully read Section A in Chapter 8 ("Recording Sales, Tabulating, Etc."), then switch to Learning Mode 5 from the Settings form. Simply follow

the procedures described in that manual section, then immediately begin using ServiceDesk as the means for recording your completed sales, creating reports on sales and related items (such as sales tax liability), creating Accounts Receivable records, managing those, and so on.

A potential but small complication is that, depending on how long you remained in Learning Mode 4, there may be jobs on which you've collected money, yet there's no record of it in the FundsJournal (for the simple reason such money was collected *before* you moved into Learning Mode 4 and began recording funds received). If this is the case, as you go to record a completed sale to the Sales Journal via the SalesEnter form, you'll find that the system fails to identify those previously collected funds. In consequence, if doing a *paid* sale, you'll need to respond to the warning ServiceDesk provides (indicating that funds have not been found to satisfy a paid sale) by indicating you want to proceed with the entry regardless (an option that's provided, though it requires use of your custom password). If, on the other hand, you're recording a *billed* sale on which *partial* payment was collected (and again, that payment is not in the system because it was made before you began recording such matters), you'll need to manually enter the partial payment into the A/R-Creation form (again, an option that's provided, password-required). Though a small inconvenience, this transitional stage will pass very rapidly.

A primary thing to remember at this stage is that now the system expects that each JobRecord will eventually be recorded, as a completed sale (even if for a sale amount of -0-), via the SalesEnter form. And it expects you to manage your AccountsReceivable (which result in part from that process), keeping on top of them by sending out statement or dunning letters when past due, and so on. ¹⁶⁶

One of the great beauties, as you achieve this stage of ServiceDesk implementation, is that now you've achieved *great* security for most of your various processes. No longer can you lose a job (either between the cracks or via a wily technician claiming it as his own), or even unduly neglect it, without knowing about it. Each must be recorded, as a sale of some kind at least, or else alarms are triggered. And no longer *can you* inadvertently give a sale credit for having been paid (absent password-protected overrides, at least), unless you've got entries in your FundsJournal reflecting such payment. And no longer can you charge yourself with having collected such funds, unless you also show they properly reached deposit or other proper disbursement—or again alarms are raised. You've come a long way baby!

Learning Mode 6 (all the above, plus the Non-Stock Parts Ordering and the UIS system)

There's not much to explain here aside from what's in the relevant sections of the manual (Section B in Chapter 1, "Managing Special-Order Parts", and Section B in Chapter 10, "The Unit-Info System"): please read both carefully in preparation for this step, and follow the procedures they describe. Of course, don't forget also to switch into Learning Mode 6 from the Settings form also.

The functionality that's introduced here is, primarily, just a great convenience. The system makes the burden of managing your special-order parts process far lighter (automating and facilitating each step), and integrates it with all other processes. Thus, there will now be a query during each PostVisitReport concerning whether parts need ordered. And there will be opportunity to create and/or attach UIS sheets to the job in

¹⁶⁶When you go to the Settings form for the purpose of changing into this mode, you'll notice there's an option you can check, immediately under Mode 5, labeled "*Leave out this function for the time-being.*" The reason for this option is because we've had some clients who, for various reasons, did not want to use the sales and accounts-receivable systems that are built into ServiceDesk (wanting instead to retain the use of systems they used before). Yet, they *did* want to use the parts ordering and inventory control systems that are involved in Learning Steps further along. This option enables that strategy. See footnote 119 at page 182 for more information.

question, etc. And, when actually ordering, checking in parts, and so on, relevant entries will be made to the connected JobRecord's History (documenting these processes right on its face), and so on.

Another benefit, that's connected with setting up the UIS system, is added functionality in the Callsheets. Now, with the UIS system setup, you can have handy drop-down boxes in the 'Item(s) Type' and Item(s) Make' boxes of the Callsheet, offering for auto-insertion the types of entries you'll typically be making to these boxes. Not a huge thing, but it's rather handy.

Learning Mode 7 (all the above, plus Inventory Control [i.e., stocking parts])

In preparation for this step, please carefully read Section C in Chapter 7 "Managing Stocked-Parts Inventory", and of course switch to Learning Mode 7 in the Settings form. Then, of course, you've got to do all the setting up for your inventory system that described in that section. That in itself is not easy, and indeed involves probably more work than the prep for any other learning stage (though it's not ServiceDesk work *per se*). That's why we've left this as the last major learning mode. Other than that, there's nothing special here that needs to be said about it.

Except, we should say, when you do finally get around to it, there are enormous rewards in getting this system setup and operating. Now, in addition to all the other kinds of *security* that ServiceDesk offers, you'll also have security in knowing you are never losing items of inventory without knowing about it, in knowing you've got the intended items at intended locations, in the quantities intended, and that you know exactly what you have and where. Security in knowing what needs reordered, what needs restocked and where. Security in knowing what your usage has been for each item, costs to obtain, etc. It is, in short, all very wonderful, and of course there's also enormous convenience in terms of the processes (usage documented as part of PostVisitReports, easy restock, reordering, etc.). You'll love it!

Mode 8, Full Implementation (Unrestricted, no limitations)

Congratulations. You've now completed all the intermediate learning stages. It's time to reward yourself by switching into Mode 8 from the Settings form. Notice, this is *not* a developmental learning mode. You are now a graduate. Yahoo! Horray! Can life get any better than this?

Actually, there is still learning to do.

In particular (and if you've purchased the video tutorials) you should now budget some time to watch Lesson # 6 (dealing with the various auxiliary functions) and # 7 (which goes over the various utilities that come with the system). And regardless, you should read the corresponding Manual chapters, 9 and 10.

But at least at this point all the primary operating features have been put into play. What's left are add-ons, auxiliary and ancillary features that may enhance, improve and help your work, but are not essentially integral with fundamental processes.

Probably the most important of these is the SourceOfJobs survey (Section G Chapter 10), but it's by no means the only one. To learn what still may be added to your repertoire, it's a good idea now to peruse this manual's Table of Contents (particularly in regard to Chapters 9 and 10). Better yet, why don't you simply read

those two chapters in their entirety (and implement desired features as you encounter their description). Then, to further deepen your understanding and skills, look over Section 4 in the Appendix (i.e., the one on “Technical Information”). The title may sound scary, but the descriptions are as down-to-earth as circumstances allow, and there’s a ton of information there, much of it very useful.

Aside from this, we suggest you review sections in the manual on an as-needed or as-wanted basis. And please, let us know how this step-by-step implementation and learning process has worked for you. We’re anxious to know.

APPENDIX

1

Printing From ServiceDesk

Any business that is today still hand-writing basic order information onto its invoices is, well, not exactly modern. With ServiceDesk, in contrast, you're now ready to leap ahead. With a single, one-time entry of data, you can now move seamlessly from taking and documenting the call, to initiating a job record, entering the item into your schedule, and printing the job invoice. Obviously, the last process requires some ancillary equipment and materials, not to mention some set-up work in ServiceDesk (at least if you want anything other than the default setup that it ships with).

In regard to the last matter, please note that if you're perfectly happy with that default setup, there's positively no need to read this chapter. However, we believe most operations will do better using pre-printed forms (rather than blank paper as the default setup is designed to work with), allowing ServiceDesk to print only the textual job information (*within* spaces on the form) rather than the entire form image. If you agree with this preference, you'll need to read in this chapter to learn how to do the setup, as wanted for your particular invoice format.

Hint: If using the video tutorials, you'll find a review of this subject included in Lesson # 7.

A. The Physical Setup

If you do not presently own a printer suitable for printing order information onto your invoices, and are not sure what to look for, we offer this advice. First, for the typical setup you probably should look beyond printers of the *laser*, *inkjet* or *bubblejet* variety (all of these simply fuse toner or squirt ink onto the front surface of a page). Even in this age, most servicers still find it most practical to use self-carbonizing, multi-part invoices (if the technician's going to write information onto a paper in the field and give a copy to the customer plus return with one, obviously, he's either going to have to write twice or else have a multi-part, carbon-transfer type of paper to write onto).¹⁶⁷ The limiting factor with multi-part invoices is that, if you're going to print through to the underlying sheets from any kind of printing machine, you must have one that forms its characters by striking the page with impact. What this means, essentially, is that you need an *impact*-type printer (i.e., one that actually strikes the page with force, something like an old-fashioned typewriter). With today's technology, this almost always means a '*dot-matrix*' type of printer .

¹⁶⁷ The only other alternative would be to have him inputting data electronically in the field, and to also have him equipped with a portable printer to produce the piece of paper your customer will rightly demand. This may be the course of the future, but most small service companies are not prepared, at present at least, to invest so much in field electronics that will soon go obsolete. Multi-part invoices, actually, are a very mature and practical technology.

If you do not presently possess a printer of the dot-matrix variety (and yet are persuaded of the benefits involved in using multi-part forms), we suggest you obtain one.¹⁶⁸ On the other hand, there is nothing in ServiceDesk that compels you to use multi-part invoices (we simply think the for most purposes it's more efficient than alternatives), and if using nothing but single sheets, other printer technologies (laser or inkjet) are probably preferable.

In terms of the invoices themselves, there has been some confusion among users as to whether ServiceDesk prints the entire invoice image (i.e., you just put in a *blank* piece of paper while ServiceDesk prints everything on it), or if it's intended for ServiceDesk to just type-in specific data on an otherwise pre-printed form (i.e., a paper already having designated *spaces* for customer name, address, telephone numbers, description work done, charges, etc.). As a general matter, we intend the latter (though the former obviously can be done, just as it is in the provided default setup). Indeed, for ordinary purposes we intend that ServiceDesk will type only initial job information onto your invoice form, mostly from the Callsheet that started the job. Later entries, of time spent, estimates made, work done, moneys collected, charges to the customer, and so on, will typically be hand-written by whatever party is responsible.

In terms of the *design-format* for your invoice form, if you do not presently have a design you're happy with, as part of your ServiceDesk package you are granted license to use the same invoice design as is shown in the Exhibits (last page of this manual). It can be "stretched" to 8.5 x 11" size if wanted. Not to toot our own horn, but we believe this is a very smart-looking and effective design. If you wish to use it, we've even provided a digital file of the form, which you can take to your local printing company. Have the proprietor there do the needed adaptation work,¹⁶⁹ and he should be able to quickly produce a perfect master image, which can then be used for printing your own similar invoice forms (this could save you considerably over having someone else do the typesetting from scratch). The file is in '.cdr' format (produced in Corel Draw 3), which most printing companies should be able to use without problem. You'll find the file on your ServiceDesk installation CD, in the '**OtherFIs**' folder, under the name '**Invoice.CDR**.'

The next question in regard to invoices is whether to use continuous forms (i.e., the kind where each invoice is connected and folded over the next in kind of a 'Z' pattern, with perforated tapes on the side for pulling through your printer's tractor feed), or more ordinary invoice sets where the three parts of a particular form are attached to each other by gluing at the top, but each set is entirely separate from others. Many

¹⁶⁸ Several years ago, there were literally dozens of dot-matrix printers on the market (because they were then the cheapest technology available)—creating a vast selection at very moderate prices. Sadly, ink-jet type printers have largely supplanted that end of the market, so that today's dot-matrix offerings are much more limited, and not quite so favorably priced. Still, there are some excellent printers, at still reasonable prices. The Epson LQ870 is our favorite, available with cut-sheet feeder at around \$600 (although, please note, we've found that the Windows driver for this printer is somewhat flawed, causing problems with the spacing of lines when printing to the invoice. For this reason, we've installed and use the driver for the Epson LQ510 for such purposes, which works perfectly).

¹⁶⁹ As provided, the invoice image in this file includes the full logo section at top, in full, as illustrated on the last page of this manual, and as applicable, obviously, to a different business than yours. It also includes a surrounding rectangle that is intended to communicate the outside dimensions of the paper on which the image should be printed (the intended width is slightly greater than 5.5", meaning your printing company will need to split legal size rather than standard letter size stock for the underlying paper). The logo section will need to be replaced, obviously, by one applicable to your own company, and the outer rectangle simply removed. These are easy actions if the file is simply loaded into Corel Draw, and manipulated from there (as someone working at your printing company should be able to do for you). It may be a good idea to photocopy this page and take it to your printing company for their review.

In case your printing company is unable to work in Corel Draw 3, we've also included an image of the pertinent portion of the invoice in the more widely-used .EPS format (that file, '**Invoice.EPS**', will be found in the same folder on the CD as the other). The only disadvantage here is that this file is solely a graphic image, rather than the actual source file in which the image was created. Because of this, there's less flexibility in changing details, of the image, if you wanted to.

If you are at a complete loss in regard to coming up with the logo/name heading to place at the top of your invoice, you *may* ask us to do this for you. Though it's not in our normal line of work (essentially, we're *software producers* rather than *graphic artists*), it's something we've nevertheless developed some expertise in. If you wanted us to make a logo/name heading for you (and, indeed, incorporate it into a complete graphic file ready for your printing company to use), we'd be happy to do so for a reasonable fee (perhaps \$100 to \$150, depending on difficulty). If interested, just ask.

people are not aware of how easily the latter can be used on modern printers. All you need is an optional cut sheet feeder to go with your printer, generally available for around \$200. These hold anywhere from 50 to 150 invoices at a time, and automatically feed each into your printer as needed. We highly recommend this approach, as there's no need to pull individual invoices apart from others or from the side perforations, the parts stay together while the invoice is in use, loading them into the printer is easier, the invoices are nicer, cheaper to produce,¹⁷⁰ etc. For most operations, we think this makes the greatest sense by far.

Another factor to bear in mind (for those unfamiliar with networking, at least) is that you don't need separate printers for each station from which you intend to print invoices. A single printer can easily be made available to all the stations in your network.

B. Creating a File that Specifies the Format Needed for Your Invoices

As mentioned elsewhere (see page 73), most servicers use their own, unique design for a work-order/invoice form (or at least prefer to use one out of a myriad of stock designs out there). In ServiceDesk, our philosophy is that you should be able to print job information onto whatever format you want.

Of course, for this to be accomplished, ServiceDesk must have a means of acquiring information, from you, regarding the invoice printout format that you, in particular, want. In general, you provide this information to ServiceDesk via what we call the *Invoice-Format-Instruction* file. This is simply a file that contains the specific instructions informing ServiceDesk of the particular invoice-printout format you want. ServiceDesk ships with such a file provided for you already (the initial default setup), but the expectation is you'll likely want to replace it with your own, different file, that specifies a format as specifically designed for your personal preference.

The Invoice-Format-Instruction file goes by a specific name, consisting of the unique FileNamePrefix that was created as part of your package, followed by a .PRG extension. Thus, if your business name was "Tri-City Mechanical," and we setup your custom files with a FileNamePrefix of "TriCity," the system would expect to find your Invoice-Format-Instruction file under the name **TriCity.PRG**.

Your Invoice-Format-Instruction file may be located in either of two places, depending on strategy. As first installed, the provided, default file will be located within each station's own \sd folder, and it's to the file located that (i.e., at each such location) that ServiceDesk will look, when going to print an invoice. You might call this the "*distributed*" strategy, as copies of the file are individually distributed to each station. This was the originally the only strategy option for locating this file, and was done in such manner with the thought that it would allow for the flexibility of setting up for different printout formats from different ServiceDesk stations (if and when such flexibility was wanted).

¹⁷⁰ In our own operation, we're paying about \$450 per lot of 5,000 invoices produced (which works out, obviously, to about 9 cents per invoice)—though it took some shopping to find a rate that low, and we expect there are significant variations in going rates, depending on the region. At any rate, we've also heard from *some* servicers that they can't find a local printing company that's willing to do such work. This makes us a bit incredulous, as we've always found that virtually any local printer, in our area at least, is not only able, but anxious to do such work. Regardless, if you have trouble finding such a printing company, you can certainly contact ours. It's Douglas Printing, 949-369-5106 or dougprint@earthlink.net.

Ultimately, however, we found that almost all users wanted just one printout setup, and it became a nuisance to have to update each station with a copy of the file when it was created and each time a change was made. So now we offer what you might call the “*centralized*” strategy instead, which allows you to maintain just one copy of the file, in the `\\sd\\netdata` folder of the Server. The one caveat is that, if you elect to use the centralized strategy (generally recommended), you’ll need to be sure and delete the local copies of your Invoice-Format-Instruction file (from within the local `\\sd` folders) at each station. Otherwise, ServiceDesk will give you an annoying little note indicating that it’s found files in both locations.

With that as background, the next question becomes: How do you create this file?

i. General Strategy

Fortunately, we’ve made it very easy. It’s done with use of a little auxiliary program, shipped with ServiceDesk, called *SD-Tools* (click on ‘Start’ in the Windows Taskbar, select ‘Programs’, then find it in the ‘Rossware Computing’ program group).

As you’ll see on running SD-Tools, your first option is to specify whether you want to setup for ‘*Invoice Printout Setup*’ or for the ‘*Source of Jobs Survey*.’ Specify the first option, then, when queried about which file you wish to load, press your Esc key. This will allow you to start a new file from scratch.¹⁷¹

At this point you’re asked a series of questions regarding the dimensions of your invoice. Simply answer the questions as they’re presented. At conclusion, ServiceDesk will place an image on the screen for you. First and most obviously, this image shows a sheet of paper that represents your invoice. Secondly, it shows a series of text items that may be printed to your invoice.¹⁷² Initially, there is no particular arrangement in regard to either the position or font size and type for these items. It is your job to provide that.

Specifically, you should take each text snippet and position it (just **drag** with your mouse) to the approximate location (relative to the on-screen image) where you want such item printed to your invoice. It’s very simple to do this.

¹⁷¹In the alternative to creating your own .prg file (or continuing with the default one that’s originally installed with ServiceDesk), you might prefer to start with one that we’ve pre-configured (but for a text-only type setup, rather than the text-plus-form-image that originally installs). Especially, if you like the invoice format as is illustrated on the last page of this manual, we’ve provided two pre-configured .prg files, setup for text-only printout. *Sample1.Prg* is setup to work with that form exactly as pictured (its size is approximately half of a letter-size [8.5x11] sheet of paper). *Sample2.Prg* is made to work with the same form, only with the form stretched to fit a full letter-size sheet of paper. At any rate, you should see both files listed during the Open-File dialog, and can select either at your preference. Both are configured to do a fairly nice printout, and could be used almost immediately with little revision, if you adopt the matching invoice format (and, of course, have it produced for you by some printing company). Depending on the particular printer being used, however, you’ll probably need to make some adjustments in the file (using SD-Tools’ nudge buttons to move the entire printout as needed [up, down, left or right], in order to line everything up perfectly. And, of course, you’ll need to save the file under the name required for your installation (see footnote at page 295). But even so, please note that this ‘Sample.Prg’ is less than perfect. Basically, it’s about the best we could do using fonts that we’re confident will exist on your machine (TimesNewRoman and CourierNew). The .prg file that we actually use in our business is much better (again, see the exhibited illustration on last page of this manual), because it uses fonts that are more apt for the purpose. As suggested in the text, you may optimize for your own situation by using fonts that are available to you, and perhaps acquiring better ones if needed.

¹⁷²Actually, you’re also going to see another object on the screen: a tiny little image of the graphic invoice image that was provided as part of your beginning, default invoice printout setup. This will be visible in the upper right-hand corner area. Assuming that you’re not going to use that image as part of your ideal setup (which, again, is an avoidance we generally recommend), you may either just drag that image completely off the area that represents your invoice, or (better yet) go into Windows Explorer and delete the actual file (it’s called *InvLogo.Bmp*, and will be found in your `c:\\sd` folder).

ii. Selecting Fonts

In addition to location, you should also select an appropriate font for each item to print in. Do this for any text item by **right-clicking** on it. At this point, you'll be presented with a dialog box that allows you to make your selection.

One factor to bear in mind in regard to selecting fonts is the difference between those that are *proportionally-spaced* (meaning each character occupies a variable space, proportional to its shape) versus *fixed-pitch* (where each character occupies the same space for any given font size). With a proportional font, the length of any line segment will vary depending on the characters that make it up. This can make it hard to predict whether any and every line, of varying text, will fit properly within a given space. With a fixed-pitch font, by contrast, 25 characters will always equal the same length for a given font size, no matter what characters are in the line. Thus, you can always be sure that a given number of characters will fit properly within a given space. For this reason, we think it's better to use a fixed-pitch font when printing to any form such as an invoice (though you may certainly do otherwise).

Unfortunately, the use of fixed-pitch fonts has become a little archaic in the Windows environment, so there a couple of complications you should know about.

First, it's not obvious (simply from looking at the font name in a list) which are fixed-pitch and which are proportional. Often times, you must look at actual text rendered in the font to find out. Hint: look at a small 'i' or 'l' and note if it takes less space than say a large 'B' or 'O'; if so, you've got proportional spacing. Also, if textual segments with an equal number of characters (such as the address-line snippets compared to the city/state-line snippets on your SD-Tools graphical layout) are in the same font yet are unequal in length, it's a tip-off you've got a proportional font.

Second, as typically setup, most Windows installations offer very few fixed-pitch fonts (the list you'll see in the font dialog box is of those fonts that already exist in your Windows installation). Typically, you might see fixed-pitch offerings limited to just Courier, FixedDys and SmallFonts, for example—and none of these are particularly attractive for use in an invoice. Fortunately, it is not difficult to add new fonts to your Windows selection. We purchased a package from Expert Software called CD Fonts, for example, at a cost of just \$9.99. It contains 1500 different fonts, a few of which are fixed-pitch and fit our purposes nicely.¹⁷³

In regard to setting your invoice font (or fonts) from SD-Tools, you should note that each text snippet sets font characteristics for itself only, with one exception. When you set font characteristics from the *CustomerName* text snippet, you'll find you're given the option to make whatever you've set there apply to the other primary fields as well. This is to make it easier for you to change the font characteristics for everything at once, when such is your purpose. Generally, we think it looks best to use a uniform font type and size, with the exception perhaps of the InvoiceNumber (where it looks more sophisticated, we think, if this is set apart in a different, typically larger style; see invoice example on last page of this manual).

¹⁷³Specifically, from this particular package we found the *Corporate Mono*, *Letter Gothic*, and *Roman Mono* font families were all fixed-pitch (we liked *Corporate MonoCondensed* best). There may be others we did not find, out of the total of 1500 fonts offered, as this particular package (as is the case in most instances we've seen) makes no effort to distinguish between fixed and variable pitch fonts, except to the extent that sometimes the font *name* may give you a hint. Anyway, we've included the particular fonts that we liked best, for invoice purposes, on your installation CD in a folder logically called "**Fonts**." You can install them to your machine by going to the Windows Control Panel, there selecting the 'Fonts' icon, from the box that then displays selecting 'Add fonts,' then browse to locate the fonts as provided on your ServiceDesk installation CD (placing that CD into your computer's CD drive, of course), then further follow the dialog to install, etc.

iii. Additional Tools, Testing, Saving the File, Etc.

Another design tool allows you to change the invoice dimensions, in an existing setup, by right-clicking on any portion of the page face that's not covered by a text-snippet. This brings up the same dimension-setting dialog you began with when starting from scratch. Also, you can move all fields at once (up, down, left or right) using the special nudge buttons you'll see located up in the tool/menu section of the form.

Basically, you're going to be using "eyeball" techniques to position the text items about where you think they should fit on your invoice page, and of course your first approximation may not be precise. Conveniently, we've added a button labeled "*Do a Test Print.*" Simply click on this button when you want to see how your existing setup prints out. If some text items print too high, drag them a little lower and test again, repeating the process until you get everything just right. One hint: use blank paper during this process, and simply hold it up to the light against an actual invoice to check for positioning: blank paper is much cheaper than invoices.

When you finally have all the text items printing exactly where and how you want them, your final task is to save an appropriate file representing your work—and, essentially, describing it to ServiceDesk. Simply click on the 'Save Changes' button and you'll be presented with a dialog for this purpose. At this point, you'll need to decide whether you're setting up under the *centralized* or *distributed* strategy (see page 269). Unless you're wanting to output with different formats from different stations, the former will generally be easier and more convenient.

Assuming the former, save your file to the `\sd\netdata` folder of the server (otherwise, it would need to be saved to the `\sd` folder at each individual station). Be sure to save it under the required name, ¹⁷⁴ and your task will largely be done.

The one major qualification is, if this is your first setting up—and if you're using the generally easier *centralized* strategy—you're going to need to remove each of the distributed Invoice-Printout-Setup files from within the individual `\sd` folders at each station. Otherwise, users at each will get an annoying message (asking which of the two different files they want to use) each ServiceDesk starts up

iv. Adding Items From the Optional-Items Garage

With ServiceDesk Version 3.4 and above, we've added the ability to optionally print several new items of text information to your invoice. These items will display on the same Invoice-Setup-Page as the standard and traditional direct-from-a-Callsheet items as described above, only they are shown off to the right-side in what you might conceptually think of as the "Optional-Items-Garage." So long as such items are left in this garage, they do not become part of your invoice-printing setup, and the connected info will not print to your invoice. If you want any such item included, however, the task is no more complicated than to simply drag it from the garage (again, using your mouse) to whatever location as is wanted on the invoice. Set font characteristics just as you would for any other item.

¹⁷⁴The Open-File dialog box *should* propose the correct name for you, but for your information here we'll review: the expected name will consist of your company's own unique `FileNamePrefix` in ServiceDesk, followed by the extension `.prg` (in the installation for Aardvark Appliance Service, for example, the `FileNamePrefix` is `aardvark`, so the necessary `FileName` is `aardvark.prg`). If you are otherwise curious as to what that unique `FileNamePrefix` is for your installation (it should be obvious from this context because it's offered for you), you can look in your `c:\sd` directory. There, you'll see a series of file names that each have a particular series of characters *from your own business name*. That series of characters is it, the little phrase that's used to name all the particular ServiceDesk *program* files, as opposed to data files, that are unique to your customized package (it's merely the `FileNameExtensions` that vary with each particular type of such file).

In case the purpose of any of these new, optional text items is not obvious, here is a description pertaining to each:

EXT1, EXT2 . . . EXT4 These four little snippets are for the Tel#Notations that may optionally be connected with each of the four telephone numbers on a Callsheet or JobRecord.

00000 This is for a second printing of the job's invoice number—useful in any context where you want this number printed more than once on the invoice, particularly if you want the second printing to be in UPC, bar-code format (simply position it in the correct place and select that as its font).

DEPARTMENT Will print the name of the company department to which the job is assigned (assuming, of course, you are using that option).

ASSGND TECH Prints the name of the tech assigned to the job (but, bear in mind, at present the tech assignment is evident to ServiceDesk only in the context of initially creating the job, when such box is filled-in from the CreateJob form; if the invoice is subsequently re-printed from the JobsCurrent form, for example, that initial assignment was not tracked, and this item will be left blank).

PRCHS ORDER # If you want a client's P.O. Number printed in a location separate from the right-end of the CustomerName line (where it still must be inserted from the Callsheet), use of this option will pull it (as printed to the invoice) from that location and print it where so specified.

CNSMR CNTRCT # Same as for client's P.O. Number above, except in reference to a homeowner's Contract Number (as in a home-warranty situation, for example) which on the Callsheet would be listed at the right-end of the LocationName line.

JOB CODE Presently, there is no context in ServiceDesk for handling this item of information. We intend to add such capability, for those clients who need it, in the future.

**MODEL NUMBER
SERIAL NUMBER
OTHER NUMBER
PURCHASE DATE
SELLING DEALER** } These items all involve text that's taken from any UnitInfoSheet that's attached to the job.

SERVICER # Identification as given to your company by a specific major client, and so indicated in connection with that client's QuickEntry form.

STATE ID # Identification as given to your region by a specific major client, and likewise indicated in connection with that client's QuickEntry form.

\$S.CALL Amount of your service call for the location involved, under the variable S.Call rate system (see page 93).

EXTRA NOTES Will print text from the "Extra Notes" section of the initiating Callsheet's MoreInfo form. Notice that this item is special because you can size it to your

taste (i.e., for the capability of more text or less, and to fit a given space in your invoice, including multiple lines). To do this sizing, first drag this item to the approximate location you'll want it, then double-click on it. A dialog will ask for desired width and height.

The addition of these optional items represents a substantial enhancement in both flexibility and capability. It is also in response to numerous requests. In such regard, let us provide just a little more explanation in regard to the last item. Some users have wanted the ability to include, as typed to the printed invoice, substantial additional notes (more than could be fitted or seemed contextually appropriate in the Callsheet's 'Description of Complaint/Request' box). Some have wanted to have special directions typed in, for example, others a brief description of previous work at the same location. Whatever the need, you can now include as much space as wanted on your invoice for such extra text, setup the printout for it, and, when wanting such text to be included simply type it into the box, provided for the purpose, in the Callsheet's MoreInfo form.

v. Dealing with a Reversed-Assumption Invoice-Format as regards Customer/Billing/Location Issues

For almost all servicers, situations arise where there are essentially two parties that you are serving on a job, one of whom is paying for it, and the other who lives at the location. Whenever this happens, there is obviously a need to have both parties referenced, not only within the internal documents of your software system (as is done, for example, in the ServiceDesk Callsheet and JobRecord format), but also upon the printed work-order/invoice. This need has long been recognized, and nearly every software system has addressed it. Nothing new there so far as ServiceDesk is concerned—except for one thing.

Most of the earlier software systems made an assumption in regard to these two parties (either one or both of which might in some senses be called a “customer”) that is precisely opposite that made in ServiceDesk.

Specifically, it has typically been the practice, where two such distinct parties exist, to refer to the one *at the service location* as the “customer,” and to the remaining party (the one being billed) as a simple “bill to.” At the same time, forms were typically setup so that the primary name/address/telephone space was intended for information pertaining to the location—with a secondary name/address/telephone area being available, for optional and conditional use, only in those instances where there was a separate bill-to.

Again, that's opposite what's done in ServiceDesk—where we figure that the party who's paying is really the one that should be considered your “customer.” After all, they're the one that's ultimately going to write the check. They're the ones that control what work you can do (and how much you can charge) if you're to be paid at all. They're the ones that may give you more business if you make them happy, etc., etc., etc. So, we've structured it to give that party the prominence it deserves. The “go to” name and location, on the other hand, is a mere third-party beneficiary of your relationship with that true customer.

Because of this, ServiceDesk is structurally opposite from those older systems. While they use the primary name/address/telephone space always for the location info, we use it always for the paying party's. And while they use a secondary space for bill-to info if it happens to be different, we use the secondary space for go-to info if *it* happens to be different.

We explain this partly to “sell” you on this different strategy in ServiceDesk, and partly to explain the context for an accommodation we've made for those still using the old. In particular, if, as you're adopting

ServiceDesk, you're still left with thousands of old invoices that were configured for the old assumption (or even if you simply insist on keeping that as the format for your invoices), ServiceDesk will need to make some accommodation when printing to those invoices. We have allowed for exactly that.

Specifically, in the top gray area of the SD-Tools form, you'll see a little checkbox labeled "*On SD printout, reverse standard Cstmr/Blld/Lctn Assumptions.*" If you simply check this, and save it as part of the appropriate .prg file, it will accommodate printing to invoices that still have that old assumption.

With this done, ServiceDesk will pull a little "switcheroo" as it prints your invoice, in particularly needed circumstances. Specifically, if there appears to be a legitimate "go-to" address in the Location-Info area (from the originating Callsheet or JobRecord), it will place all of the Location-Info text into the positions that, in your .prg form, you've designated for the "Customer"—while placing its Customer-Info text into the spaces you've designated for the "Location" (in which case, obviously, those can be thought of as pertaining to the "bill-to"). If, on other hand, ServiceDesk does not detect a legitimate "go-to" address in the Location-Info section (it makes the determination by looking for an opening bracket in the address line, i.e., '[', as should precede the grid reference that should exist in any ServiceDesk go-to text), it assumes that the text in the Customer-Info section represents the one and only party connected to the job, and refrains from making any switch.

By this means, ServiceDesk can adapt perfectly in its printout to an invoice structure that uses the old assumption—while still retaining its own assumptions internally. The only remaining concern is if you're using NARDA forms for your invoices, which don't have any separate spaces for bill-to information. In this case, you can simply drag the various Location-Info fields off of your invoice-printout setup, while of course also checking that "Reverse assumptions" checkbox. This will result in loss of separately-printed billing info to your form, but since the NARDA has no place for it anyway, what can you do (aside from selecting a better form, which we highly recommend)?

vi. The Option of Adding a Graphic to Your Invoice

A final capability in this context arises if you want to print any kind of *graphic* image onto your invoice. As an example, this is what we've already done for you, to setup your initial default printout so that it prints an entire form image onto the sheet of paper, in addition to text concerning the job. That provided form image is simply a *graphic file* that was added to the provided printout setup. If you wanted to similarly setup to print a form image onto previously blank paper (rather than using pre-printed forms, as we generally recommend), but of a different form, you'd simply need to place in a graphic file having the image you prefer. Or, you might want to use a graphic for some other purpose, and set it up to print only within a limited area on your invoice form.¹⁷⁵

At any rate, if you want to add a graphic to your ServiceDesk invoice printout, there are three simple steps to the process.

First, you've got to come up with the graphic file that has the image you want. This is not ServiceDesk's task. It's the task of a graphics program, such as Microsoft's *Paint*, for example (included with every Windows installation, just click on 'Start' from the Taskbar, then 'Programs', then look for it under

¹⁷⁵ If, for example, you were using stock forms that had all the lines and spaces already printed in, but just an empty space for your company name, address and logo, etc., you could set it up to print a nice graphic of the same into that space. Or, if you're a company that does business under two different names (and want to be sure each invoice has the appropriate business name at top), you could setup one station to print a name/logo with one name (probably onto an otherwise pre-printed form), and another with the other name/logo. Or you could be creative. Maybe have one little corner in the invoice where, as a graphic, you place a photo of one of your techs, under which is a caption reading "Our technician of the month."

'Accessories'), or *PhotoShop*, *CorelDraw*, or a legion of others. Or, you might obtain a pre-made graphic from any of many sources. Regardless, the one criteria so far as ServiceDesk is concerned is that the graphic must be in what's called "*bitmap*" format—which means essentially that the file name will have a **.bmp** extension on its end.

The second step is to give your graphic file the particular name and location that will enable ServiceDesk to find it. Specifically, it must be named "**InvclLogo.bmp**" and placed side-by-side in the same folder that contains you're the operative .prg Invoice-Format-Instruction file (i.e., either in the **\sd\netdata** folder on your FileServer if you're following the *centralized* strategy, or in the **\sd** folder of each station where you want it used, if your following the *distributed* strategy).

The third step is to specify the specifics, in terms of size and position, of how this graphic will be printed to your invoice. This is done from the same place where you're otherwise specifying the invoice printout format: from within SD-Tools. When you run that program and specify the 'Invoice Printout Setup' option, the system looks to see if a file exists in the name and location as just described. If so, it places the graphic image (as found in that file) on the screen of your invoice-printout-setup. At first it's small and in the top-right corner of the work area. Your job is simply to set its size and position as wanted (just as with the text snippets). To set its size, **double-click** on it, then respond to the dialog. To set its position, simply drag as wanted.

As when setting the positions of text snippets, we recommend test printing to see exactly how the graphic ends up printing on the page and in comparison to the text snippets themselves. While the on-screen work area provides a pretty good *approximation*, there will likely be a bit of variation between precise on-screen positioning and what you actually see on the printed page. Also, you may find this variation is different depending on the printer the output is sent to, meaning that if you switch printer types, it may be necessary to adjust your setup, somewhat, to keep every item printing (in terms of the graphic's print position compared to each text snippet's print position) as wanted.

As when otherwise designing your invoice printout setup from within SD-Tools, when done simply save under the appropriate name and location. When ServiceDesk next goes to print the invoice, it should find that file (and the accompanying InvclLogo.Bmp file), and print exactly as wanted.

vii. Configuring Your Ticket for Tear-Offs

If you do a significant amount of shop work, you've likely thought it would be nice if, in addition to printing a standard ticket, ServiceDesk would also print a "tag" to go on the machine, and a "claim ticket" to give the customer. We've added provision to make the above easy to achieve.

The basic idea is, design a new form image with all the graphics you want, including tear-off regions appropriate to your purposes (whatever regions you want are just fine).

Next, use the SD-Tools utility to modify your username.PRГ file for text to print in spaces, where and in the format wanted, for the the non-tear-off regions of your form (this step is precisely the same as it would be for any normal ticket design, as specifically described in the first section of the manual's appendix).

Now, use SD-Tools again to assemble the particular text-to-print fields that you want, in the particular places that you want them for the first tear-off. But, instead of saving this work as username.PRГ (as you did for the first and normal file), save it instead (same folder location) as TearOff1.PRГ.

Finally, do precisely the same for your second tear-off area, only this time call the resulting file TearOff2.PRG.

The result is simple. ServiceDesk is programmed so that, when printing an up-front ticket, it looks for TearOff1.PRG and TearOff2.PRG files. To the extent it finds either, it adds to the printout text fields as specified therein.

2

Setting Up for Electronic Retrieval of Messages From Your Answering Service

Most answering services today are already taking and storing their messages on computer. Most also have the capacity to transmit that data electronically over a modem (their systems typically refer to this as "remote printing"). All that's needed, for ServiceDesk to feed this information into a Callsheet, is a common-language protocol by which we can distinguish from among the various items of information in the data stream. With such a protocol, ServiceDesk can easily place each such data snippet into its correct Callsheet location.

The protocol is simple. As part of their own software system (StarTel is one of the more commonly used packages), your answering service must already configure a customized inquiry page for your company (this is the page that comes up on each operator's screen when she is taking a call on your line). This page consists, basically, of a series of input lines, each preceded by a label which denotes the type of information that should be entered on it (see example in Exhibits).

Now here's the key. All you must do is have your answering service label its lines—on the inquiry page it uses for your company's calls—with the particular words ServiceDesk is prepared to recognize. These KeyWords are as follows:

- In:** Following this word, ServiceDesk will look for the date and time the message was taken, and will plug such values into the 'OriginInfo' section of the Callsheet.
- Nm:** ServiceDesk will look for the name of the caller. If the data consists of two words, ServiceDesk will assume it's the first and last name of the caller, and will invert the two words, inserting a comma between, before plugging them into the Callsheet's 'CustomerName' Line.
- Telephone Number.....:** The expectation here is self-explanatory. The ellipses are added because with their addition it leaves only enough space on a standard 38 character line (which is typical for answering service software) for entering a complete telephone number, and nothing more. Without such limits operators sometimes put other data in such a line, which gets lost when ServiceDesk tries to stuff it all into a mere 12-character Callsheet telephone number box.
- Adr:** Self-explanatory.
- Cty:** Self-explanatory.
- Prblm/Rqst:** Ask your service to allow for two lines of input data under this heading. This data is plugged into the Callsheet section similarly labeled, where there is

obviously plenty of space for two lines of data from your answering service, and it's a place where that much data is often useful.

Items(s) Type.....:	Self-explanatory, except we again add ellipses (making the entire label 21 characters) to limit the length of field remaining for operator input.
Item(s) Make.....:	Same as above.
Appointment Dt & Tm.:	Same as above.
How will this person pay?:	It's often useful to request that an answering service ask this question as a polite means of assuring the customer knows payment is expected when the service call is performed. ServiceDesk inserts any info your answering service inputs on this line into the 'LocationName' line of the Callsheet. The field is unessential if you prefer not to use it.

Once your answering service has set itself up to precede each of its message lines with these keywords, and has taken a few messages under the format, go ahead and try getting messages electronically. In all likelihood your service will need to request "*remote printing*" from their software (if needing to help them out, you can tell them that on the StarTel system the appropriate command is Ctrl-R). In response, their computer will dial yours (assuming it's been provided with the correct telephone number). To answer and receive the messages, you'll need to bring up your ServiceDesk Communications form (Alt-R for 'R'eceive messages, or use the menu). From that form you can set for "*Auto-Answer*" before the phone starts ringing, or set to "*answer Now*" after it's already ringing. Once the connection is made, you should see the entire data stream fill rapidly into the form's receive box, while each item of information correctly pops at the same time into the appropriate position of its own Callsheet.

Your service should set its modem, incidentally, to communicate at 1200 baud, E,7,2 (ServiceDesk will look for your modem, internally, at whichever CommPort you've specified via the SettingsForm). If these settings won't work for your situation, let us know and we'll send you an updated version which allows you to change the settings.

A final note. It is standard on the commonly used StarTel system (used by answering services around the country) for each message to be separated with a string of hyphens (see sample printout in Exhibits). ServiceDesk looks for this string to determine that one message has ended and the next begun. If your service uses a different means to indicate message separation, let us know and we'll adapt ServiceDesk to comply.

3

Creating Your Yellow Pages Ad List

When your dispatch operators perform the SourceOfJobs survey, it's obvious they need a list of your Yellow Page ads to match with the customer's responses. Plus, the same list is needed so that ServiceDesk can tally to each ad the responses it generates, and to display these in your SourceOfJobs Report. It follows that such a list must be provided somehow, and obviously, this is not a list we can create for you. You must do it yourself.

Again, we've provided a framework for you in SD-Tools. And your task, still again, is to run that program (click on its icon from the ServiceDesk program group in the Windows Program Manager). Upon selecting the Yellow Pages AdList option, all you have to do is to type, each on its own line, a simple description of each of your Yellow Page ads. In this case no sample file is provided, but to give you some idea of how such a list might be designed, consider this list that was created by Aardvark Appliance Service:

8 S.C.Chamber 95 (Why Call Us, appl sctn)
19 Donnelley 95 (in column, appl section)
22 Donnelley 95 ("16 Reasons," appl sctn)
41 PacBell 95 (in column ad, appl section)
42 PacBell 95 ("16 Reasons," appl section)
46 PacBell 95 ("How to Choose," appl sctn)
88 S.C.Chamber 95 (Why Call Us, DW sctn)
196 S.C.Chamber 95 (Why Call Us, rang sctn)
201 S.C.Chamber 95 (Why Call Us, refr sctn)
238 S.C.Chamber 95 (Why Call Us, wshr sctn)
430 PacBell 95 (in column ad, DW section)
967 PacBell 95 (in column ad, rng section)
986 PacBell 95 (in column, refer section)
1231 PacBell 95 (in column, washer section)

Notice that the people at Aardvark have chosen to place a page number at the beginning of each ad's description. This is an effective strategy, for it enables them to simply ask their customer: "Can you tell me which page number the ad is on?" If their customer said "page 88," obviously they could reply with something like: "Oh, that's in the San Clemente Chamber book then," to confirm that the correct ad had been identified. It works quite well, but you should feel free to use any kind of distinguishing description you wish, and also to place each item in whatever order you prefer. Just be sure it's a separate ad that's referenced on each line.

Obviously, the Yellow Pages Ad List is something you'll need to re-create (or at least revise) each year as your actual ads change and you conduct new surveys for the new year's set of books. Again, just use SD-Tools for the purpose.¹⁷⁶

¹⁷⁶ Another factor you might notice is that ServiceDesk will automatically add two selections, which do not come from your list, in the list of ads actually presented to your surveyor. One is labeled "Unknown (YP elswhr, cnt rmmbr whch ad used)." Obviously, there'll be

One factor to bear in mind, in terms of revising an existing Ad List, is that it should not, as a general rule, ever be done in connection with a survey that is already in progress. One reason is because, if you add new ads that weren't there for part of the survey period, they'll be under-represented (vis-a-vis ads that were in the list for the full survey period) in terms of results. An even more critical reason is because the JobSource form, which reports on survey results, tabulates responses to each ad based on the exact sequence of characters, for the ad, as included in the list at the time each particular questionnaire is conducted. If you change the sequence of characters for any ad's reference within the list (actually, ServiceDesk looks at just the first 20 characters of each description), the revised listing will no longer be matchable to previous questionnaire results—making survey reports, as applicable to such matters, rather useless.

Thus, whenever you revise an existing Ad List, SD-Tools will ask you to acquiesce in starting an entirely new survey (i.e., the files containing old results will either be purged for you, or re-named to remain in archive form, at your discretion). This prevents the potential problems, described in the last paragraph.

cases where a customer looked up the number at home, wrote it down, then called you from work (or something similar), and so won't have the ad in front of them, and will be unable to tell you which specific ad they used. This option is provided for such situations. The other added selection is labeled "*Other ads (type description on input line)*." Actually, this is not a selection per se, it's merely there to remind your surveyor that if someone is responding from an ad that's not in the list (from a previous year's book, for example), she should type in a description of the ad rather than selecting from among the offered options. ServiceDesk will then tally this description to the "*Other ads*" category in the survey results, and will also offer a separate display showing a listing of the entered descriptions.

4

Technical Information

As any software must to stay current, ServiceDesk has evolved over time. As features have been added, changed and improved, we strived to keep the manual updated to the current state of affairs. Over time this has resulted in a manual whose structure and language is less straightforward and simple than we (and probably you) would like—for as things change it's far easier to insert an explanatory paragraph here, a footnote there, change a description slightly over there, etc., versus a complete re-write that, could we afford the time to do it, might describe the new situation more simply than by using such cumbersome appendages.

One consequence is that early in the year 2000 we realized the main text contained much more technical detail than the average reader would probably care to read. Realizing it would be better to keep the main discussion comparatively simple, we've gradually been moving the more technical discussions into this separate appendix area, where they may be consulted as needed, or perhaps merely made the knowledge of a resident “expert” in your company.

A. Specifics Regarding the Customer Database System

If you've used other CstmrDbase systems, you may have a bit of un-learning to do before making full sense of the ServiceDesk setup. Ours is different.

In a typical system, the CstmrDbase consists of a master list. This list has one entry for each customer. Each such entry lists things such as a customer's name, address, telephone numbers, etc. The list of entries has no function except to be a repository of this information, and a source from which such information may be drawn when needed. Typically, the list involves a considerable amount of housekeeping work on part of the user—in first creating the entries, keeping them up-to-date, eliminating duplicate entries, correcting for errors, address or telephone number changes, divorces, etc. These are complications we sought to avoid in the ServiceDesk system. How have we done it?

Basically, we eliminated the master list. ServiceDesk has none. Instead, it simply uses the accumulated history of past jobs (i.e., the one that will develop as you use the system). Each of these jobs, after all, has—as part of its record—all the information that would likely be in any master list (and much more). The trick is simple. ServiceDesk maintains *indexes* to this mass of past jobs. By doing logic-based searches within these indexes, it can instantly locate any name, address or telephone number that arose in any of those past jobs, along with the full information set from each job that pertains.¹⁷⁷

¹⁷⁷ Bear in mind that you don't literally “see” these indexes. Their function is invisible, enabling behind-the-scenes work which the casual user needn't even think about—except we explain them here because, by understanding at least some basics regarding their operation, you'll better comprehend the external working of which you indeed must be a part.

So why do you need a master list? In fact, there *are* some potential benefits in more traditional systems (such as that a master list may contain information of a kind that would not normally be put directly into a job record, for example), but for a typical ServiceCall-performing operation, we think their disadvantages (especially as compared against the many benefits of a pure job-record-based system)¹⁷⁸ outweigh any advantage. Regardless, please do not look for such a dedicated master list in ServiceDesk, for there is none. In our system all customer information is found, quite simply, in the actual record of past jobs.¹⁷⁹

At least, that's the generality by way of background, but in fact there are several specifics it may be helpful to know.

First, in the foregoing paragraphs we've used the term "*past jobs*" somewhat loosely. In fact, the CstmrDbase indexes reference jobs both from the JobsArchived file and from the JobsCurrent file (at least those that have "in the past" *been* indexed, see following). In understanding this, please try to avoid a mistake common among new users who sometimes think it's *Callsheet* text that should be indexed to the CstmrDbase. Though first written to a Callsheet, job information is not *that* until it's part of an actual JobRecord, which of course is initially *created from* a Callsheet, but is certainly different from it.

Second, you should know how the CstmrDbase indexes are created and maintained.

Utilities for doing so are found in two different places. One is in the JobsCurrent form, another in the JobsArchived form. The difference is that, while the former is designed merely to *update* the indexes from time to time, the second creates a *new set* of indexes entirely from scratch. Why are both functions provided? In fact, when you're first using ServiceDesk either method would work fine for all purposes. However, after you've accumulated many thousands of job records, the process of making new indexes from scratch may take many minutes of intense, no-other-use-possible computer time. To minimize that inconvenience, you'll more typically use the mere update method. Yet sometimes you may still need the MakeNew method because it's possible, from time to time, that some corruption will creep into these files, and it's only by building virgin indexes from scratch that it can be eliminated.

To be more specific in regard to the *mere-update* method, it's provided in the JobsCurrent form as part of its *Archive* utility. That utility, obviously, invokes the process of moving completed JobRecords out of the JobsCurrent file and into the JobsArchived file. This event *necessitates* updating of the indexes, because they reference the *location* of these jobs (i.e., in which file and wherein), after all. Thus it's logical to incorporate updating of the indexes into the same process that involves moving the underlying records. In fact, during this routine the process updates the indexes to reflect all records in the JobsArchived file and all that *currently* exist within the JobsCurrent file. We make the distinction as to those records that "currently" exist for this reason: Any new jobs that are added, after the CstmrDbase indexes are either updated or newly created, will obviously not be in the indexes. Thus, these particular jobs will not show up in any CstmrDbase search (although, remember they will still be accessible from within the JobsCurrent form itself, based on any of its built-in searches).

¹⁷⁸ Suppose, for example, one of your customers has moved to a new location. They call for service and, as you begin typing-in their name, you see a reference to past jobs at the previous location. Looking at the most recent such reference, you ask if the address is "still 123 Somerset Lane?" "No," the customer replies, "we've moved," and so she proceeds to give you the new address. At this point you simply type the new address into the Callsheet, and there's no need to change any master-list entry. Nor does your system lose track of the fact that earlier jobs for this customer were performed at a different address than the one you're now typing (as would most master-list systems since they match jobs to whatever is the *current* entry within that master list). Of course, the next time this same customer calls and you bring up the most recent job record (i.e., the one you presently just created), it will have their new address. So, just as if you'd used a master-list that with extra effort had been updated for that new address, you can instantly insert the correct data.

¹⁷⁹ Actually, to kibitz just a little, there is a ServiceDesk context that allows extra, non-job-record-based information to be attached to specific customers, and used in several contextually appropriate circumstances. That is in regard to the QuickEntry forms (see page 64).

This last fact, incidentally (i.e., non-availability of post-index-created jobs via a CstmrDbase search) argues strongly for running the JobsCurrent form's Archive utility with some frequency—or better yet, letting the Auto-Archive utility do it for you automatically on a nightly basis (see page 209).

Regardless, there is a related detail that bears explanation. To absolutely maximize the speed at which CstmrDbase searches are performed, we've designed it so that each ServiceDesk station maintains its own local copy of the indexes (this way they are not each simultaneously clamoring to access common files from the Server). Plus, each station keeps the index files open during normal operation (rather than opening and closing as needed with each use). This complicates the matter of making new or updated indexes from any particular station, for the others must receive updated copies. Utilities are built-in to both procedures to assist in this (the Auto-Archive sequence takes care of everything for you), but with this word of explanation it may make more sense. Also bear in mind that if you've failed to use the built-in utilities for updating other stations, it's easy to copy the relevant files over (from one station to another) using Windows Explorer or other Windows utility (it's simply all four files that you'll find in the 'sd\cstmrdb' folder that are relevant).

Third, you should understand the basis ServiceDesk uses in deciding which text from a given job record should be included in the indexes. Obviously, not every item of text in every job is going to be properly applicable, and ServiceDesk must have some means (without *true* intelligence) of deciding what's merely extraneous text within a field and what's real information of the kind that should be indexed.

It's criteria is fairly simple. In regard to the *CustomerInfo* block from the job record, the system figures that any leading text in the name field must be a true, properly-index-able name, and any in the address field similarly index-able for that purpose, and similar for the two telephone number fields. In regard to the *LocationInfo* block, however, these assumption seem considerably less valid. On many jobs, after all, the customer and location will be one and the same, and presumably therefore you'll have true customer information only in the first block. This leaves the second block free for extraneous notes, and if your office is like ours you'll occasionally use it for such purpose. So how can ServiceDesk tell? Quite simply, it looks for an opening bracket (i.e., "[") in the LocationAddress line (as would appear in conjunction with a grid reference, for example, and should exist if you've used that block for true customer information).

In other words, if the system finds text such as "**123 SOMERSET LANE [923E5]**" in the LocationAddress line, it will include that and other LocationInfo-block text in the indexes. If, on other hand, it finds something like "**THEN TURN LEFT ON HALIFAX**" (or even "**123 SOMERSET LANE 923E5]**"), it will not index any of that block's text.

One more caveat in regard to what's indexed: by design, we intentionally do not index name, address or telephone numbers that belong to those customers that, within the QuickEntry system, are designated as HighVolume-type clients (see page 57). The reason is because, presumably, you'll be generating many, many jobs under these clients names. If each such job created its own index references based on the HighVolume client's name (or address or telephone numbers), it would expand the indexes needlessly, plus be of no real use. Instead, these jobs will all be indexed (subject to the proviso explained in the preceding paragraph) based on information in the LocationInfo block of the job record, making any needed lookup very practical indeed.

Fourth, it may help to be reminded that, as a means of displaying the job records it locates, the CstmrDbase system uses an abbreviated instance of the JobsArchived form. Why, you may ask, did we do it this way and not use a unique form, designed solely for the purpose? Basically, it's a matter of economy. The JobsArchived form was already well-designed for displaying job records, so why not use it? It seemed sensible to us.

But the JobsCurrent form is likewise designed to display job records, so you may wonder why we did not use it instead? Well, it's an already heavily-used form, doing many tasks, while the JobsArchived form was more lightly used. To impose additional duties on it (and build-in the needed adaptations) was simply more logical.

Of course, each form (JobsCurrent and JobsArchived) in their standard modes are designed to access and display job records only from their own respective contexts (current jobs or archived, as the case may be). In the CstmrDbase context, we've had to adapt the JobsArchived form to allow it to display either archived (which is more normal for that form) or current jobs. This has an important consequence that may seem more logical if explained here. Specifically, you cannot edit a job record when viewing it from directly within the CstmrDbase context. Instead, if it's a *current* job record (i.e., it's still in the JobsCurrent file) you must go to the JobsCurrent form and load exactly the same job record into it, for editing or other operational tasks. If it's an *archived* job record, you must reload it into the JobsArchived form with that form in its standard, rather than CstmrDbase-display mode.

To simplify either task, we've built-in a very nice shortcut. Suppose you've looked up any job (current or archived) via the CstmrDbase system. The job is displayed for you in the abbreviated, CstmrDbase-displayed mode of the JobsArchived form. You want to do some real, operational work (such as editing, for example) within the record. What do you do? Simply press F7 if it's a current record, or Ctrl-F7 if it's an archived one. *Ordinarily*, of course, such action would load the forms with *most recent* records loaded into them (then, if wanting a specific different record, you'd have to use some means to locate it). In this context, however, the system will load the respective form—with the record that you were already viewing in the CstmrDbase context loaded! (Actually, in the case of an archived record it will convert the already displayed JobsArchived form into its full-use mode, and reload the same record as was already displayed). This is a very handy tool, so don't forget about its availability.

Fifth, be sure to consider all the different modes by which you may conduct CstmrDbase searches (as summarized beginning at page 66), each of which involves a matter of considerable convenience within its own context.

Sixth, if there is an item (name, address or telephone number) that you believe should be coming up in a CstmrDbase search, but is not, consider the following queries by way of troubleshooting:

- (a) Has either a MakeNewIndex or JobsArchived-form-imbedded Index-Update routine been run since the job was created (created, that is, so that it had an actual JobRecord viewable from either the JobsCurrent or JobsArchived form)?
- (b) If it's a last name you're searching for, was it listed on the job in proper last-name-comma-space-then-first sequence ("BOYLES, JOHN", in other words, when the name you're searching on is "BOYLES")?
- (c) If it's an item from the job's LocationInfo block, does the address line of that block have an opening bracket (i.e., "[") anywhere within it?

Seventh and finally, if the above checks fail to provide a solution to any failure-to-show-up problem, or if you have the item showing up but the system otherwise fails, please don't forget to consider that your indexes may have somehow been corrupted, and you may need to run the JobsArchived form's MakeNewIndex routine to make new ones from scratch.

B. Details on the StreetList System

In our main discussion on use of the StreetList (in Chapter 5), we described how to insert a street name to a Callsheet (see page 64). A detail we did not mention is that, as you do this, the *label*-text for that section of the Callsheet (i.e., either the Customer- or Location-Info block) will, in most instances, momentarily flash an informative message. Most frequently, it will change from its normal reading (such as “Customer Name, Address and Phone Numbers,” for example) to show this statement: “*Confirmed Address Number in Allowed Range.*” You might wonder, what does this mean, and what exactly is involved in the underlying mechanics?

Basically, as part of your custom package we’ve created two StreetLists. One is the normal list, and it’s what you see (within the drop-down list) any time you’re typing a street name in a Callsheet. The other is hidden, and contains somewhat different data. Specifically, each listing has a high and low number indicating the range of addresses that exist within a given segment of the street.

What happens is that, as you select any street for insertion from the normal list, ServiceDesk consults this second list (we call it the “BlockList”). Assuming it finds a matching entry, it checks to be sure the address number you’ve indicated fits within the indicated range. If so, the system flashes that nice, confirmatory message.

Of course, the system may find you’ve typed a number that’s outside the indicated range. If so, there will be a warning message suggesting you reconsider. In fact, the system assesses how far from the allowed range your number is, and it adjusts the stridency of its warning accordingly. Obviously, there is a very nice error-catching function implicit in this system. In many cases it will help you catch an error in terms of the number you typed, and save your technician from costly frustration in seeking the house in question. In other cases it will help to catch the fact that, perhaps, you entered the number correctly but selected an inappropriate street from the list.

At any rate, there is still another function provided by the BlockList, which may best be understood by mentioning a detail about the StreetList (again, that’s the one you see). The StreetList is constructed to have a single entry for each instance of a given street name that occurs under a given zip code. Just one, no more. Thus, you may have several entries within this list of what is in fact a single street, for sections of that street that pass through different zip areas. On the other hand, for all of that street’s length that exists within a zip area, there will be but one entry, even if that length falls across considerable spaces on your DispatchMap. The grid reference that’s given for this section is targeted for the middle thereof, yet the job in question may be at either end. Thus, absent some solution, you could end up with some jobs being displayed less perfectly (in terms of position) than desired. Essentially, the standard, StreetList accounts for position of a long street on the basis of zip only, and with no more positional-specificity than that.

This is where that BlockList really comes in. We’ve described how, as you’re inserting a street from the standard list, the system silently checks there for a match. Well, it so happens that when we build the BlockList we fictionally break up any long sections of a street into segments. We then include separate entries for each such segment, and of course each includes a specific grid reference for the block range it includes. Thus, when you pick a street and the system checks in this list, it may find *multiple* matches. If so, it searches from among those to find *particular one* that includes the address number you’ve indicated. Upon so finding, the system inserts *its* grid coordinates—rather than the presumably less perfect ones that were displayed in the main list. When this occurs, you’ll see a different message as the corresponding Callsheet label flashes (in

fact, in this instance it will literally flash off and on for a few seconds). It will now read “*Grid Coordinates Honed for Greater Accuracy.*”

By this means the system fully accounts for the address number as it creates a grid reference. That is one of the topics we wanted to discuss in this section. The other involves difficulties you may have (of one kind or another), along with the concern of how to handle grid references if your system (or any portion of it) was not keyed to an underlying map book.

Our first concern is if you can't find the street you're seeking.

So, you've got this great new system; you're showing it off to some visitor, and as you're typing in their street name to show them how cool that drop-down list is, the list disappears, because there are no matches to what you've typed. Dang! How do you deal with this?

Well, the first likelihood is that the wanted street *does exist* in the list, and it's simply there in a somewhat different format (or spelling) than what you've typed. The standard, as-you-type search only works (i.e., it only shows matches) if there's perfect correspondence between your text and entries from the StreetList as *matched when comparing from the beginning of each such line*. For such reason, if you're typing “LILAC ST,” but the actual entry is in the list as “N LILAC ST,” you're not going to get a match. Similarly, if the customer tells you they're on “LA BOMBA” (so that's what you type) but the real entry is in there as “CALLE LA BOMBA,” you are (again) not going to see it—at least not via the standard, as-you-type search method.

But there is a solution—based on another kind of search. If you're not seeing the street you want, try hitting F1 (you do need to be in that same context). At this point ServiceDesk does a different kind of search. It takes *whatever text* you have as a street name, then looks for and displays any line from your StreetList that has that text existing *anywhere within* its line. Thus, if you'd typed “LILAC ST” but it was in the list as “N LILAC ST,” this search will find it for you.

Bear in mind, you'll get more matches if your search target is less demanding (“LILAC” would likely produce more matches than “LILAC ST” for example). And please realize this kind of search may take (depending on factors) up to a few seconds. Probably about half the time that you don't initially see the street you want, you'll be able to find it via this method.¹⁸⁰

So what about when the street is simply not there at all?

As has been indicated elsewhere, the raw data for your StreetList comes from a database developed and maintained by the U.S. Census Bureau. This is the same data that's relied on by virtually all commercial street programs that map the entire country (regional map producers typically have their own data sources). The data is quite good, having excellent (if perhaps somewhat dated) coverage of almost all incorporated areas, urban and suburban. For rural and unincorporated areas, the coverage may be less thorough.

A missing street presents a challenge, since at the least you need a grid reference so that a job on that street can properly display in your DispatchMap. It's obviously no great challenge to manually type in the

¹⁸⁰ Another small issue concerns the fact that, occasionally, our grid references may be slightly off as compared to those in your map book. Particularly, you'll find this happens when streets are very close to the edge of a grid, or overlapping between two. We use a complex set of algorithms in the effort to produce references that mirror, as closely as possible, those in your book, but since our process is different than what's used by your book publisher (theirs is probably manual), it's impossible for the result to be in all cases identical. Still, our references should be pretty darned close to precise. If you see any evidence they're significantly off, please let us know, as we may need to adjust the setup we created for you.

whole address (just as you would in any non-ServiceDesk system), but how do you come up with that grid reference?

Well, as a first option (and assuming your system is keyed to an underlying map book), we suggest you keep a map book handy by your desk, and when this happens go to its index, manually lookup the street, and type-in its reference on the Callsheet (between brackets after the street name, just as if ServiceDesk had inserted it). Sure, this is a bit of work, but you end up with a good reference, and if you didn't do the lookup now, your tech would probably need to do it later.

But what if your system is not keyed to an underlying map book, or (even if it is) the street of interest is outside the map-covered area?

If you go to your DispatchMap (F5) and hold down your left mouse button (on any space that doesn't have a job reference), you'll see an underlying grid. That's the address system ServiceDesk uses, internally, for positioning things (anything and everything, really) on your Map. Everything you see has some *position*, essentially, on that grid. Indeed, if you're using a map book and ServiceDesk needs to display a job based on a page and grid reference that references it, ServiceDesk has to perform a little trick first. Essentially, it looks in a *translation table*, which is one of the elements we custom created for you. That table tells it that, for a given page and grid combination from your book, the corresponding on-screen position must be X (actually X and Y, where X is the horizontal and Y is the vertical component), in terms of that underlying grid setup that you see when holding down your mouse.

Because that underlying grid is what ServiceDesk *always uses* internally (and any outside reference is merely *translated* to it), we call it the "**NativeGrid**."

You need to know about this, in particular, if your system was not keyed to an outside paper book, because that means in all instances your references will be *directly to* this NativeGrid, instead of having your book's reference stand in as a kind of go-between. And again, even if you have a book, you may have areas it does not cover, and this concern becomes equally important. ¹⁸¹

At any rate, suppose that the street you want has been proven (even via extended search) as absent from your StreetList, and that finding it in an applicable map book's index is also not an option (either because it's not in the book or you simply don't use one). Now, even though, most obviously, it's little problem to type the address manually, how do you come up with an appropriate (and accurate) reference to the NativeGrid?

As a first solution, if upon examining your DispatchMap you can formulate a really excellent idea of where the job should display, just click there. ServiceDesk will note the position you've clicked. Now, go back to your Callsheet, position your cursor at the end of the street name, and hit **Ctrl-V** on your keyboard. ServiceDesk will now add the reference for you. It's that easy (as you might guess, ServiceDesk is simply using the Windows clipboard; having inserted the grid-reference thereto when you clicked in the DispatchMap).

This is particularly the procedure to use when you have that rare, distant job that's not even in the territory we drew for you. In that instance, it's best to represent the job's reference on the periphery, outside the drawn area, but in the direction toward which the tech's going to have to drive in order to reach the job.

¹⁸¹If you have a mixed system (i.e., some of your territory was covered by a book and some was not), you'll see the evidence in your StreetList—for some of references will be to your book, while some will be to the NativeGrid. It's easy to distinguish between the two. Rather than the format used by your book (which might be something like, say, "922F5"), NativeGrid references will always consist of two numbers separated by an asterisk (e.g., "19*26"). Quite simply, the first number defined number of grid spaces moving horizontally from the left edge of your map. The second defines number moving vertically down from its top edge. Very simple!

Use this *click-at-wanted-to-display-position* method, and you can easily create the reference that will make the job reference display right there.

There is a major shortcoming to this method, however, in that if you or your operators may have little idea, as they look in the DispatchMap, where a given address *ought* to display. And if you or they are inaccurate (especially if you're grossly inaccurate), it may result in very poor routing of the techs.

This shortcoming was addressed in late 2006 via creation of a new custom data file, along with programming to harness it. The new file is called a *ZipsLocationList* (it has a .zpc extension). Basically, it's little more than a list of your zip codes, along with a references to the NativeGrid that indicate the weighted center of population within each.¹⁸²

To explain use of this new list, let's again paint the scenario. The street you want is not in the drop-down list, and use of a map book reference is not (for whatever reason) an option either. Here's what you do:

Just finish typing the street name. Don't worry about the grid reference. Just go the next line, where normally (at least if doing full-on manual entry) you'd begin typing a city name. But don't do that. Instead, just type the zip code. As you type the fifth digit, you'll see some instant magic. ServiceDesk will find the zip code in your ZipsLocationList, appropriately insert its NativeGrid reference in the street address line, and change the city-name line to its appropriate full text.

This is, by far, the *easiest* method of dealing with a missing street. It's so easy, it's fun. But bear in mind, the display location it creates (being based on zip only) is but approximate. Depending on how big an area any particular zip covers in your DispatchMap (along with the vagaries of where a particular job actually happens to fit within the zip), the relation between true and displayed location may range from exact (when lucky) to as much as several grids off.

So now I've fully entered info on a street that wasn't in my StreetList. Shouldn't that now be added, so it's available next time?

The answer is yes, of course, and ServiceDesk has the means to make it happen.

The process, basically, is that as you do the job/sale process from a Callsheet in which a new street name assuming manually entered (and assuming you included a corresponding grid reference, properly enclosed in brackets), ServiceDesk will notice the fact, and ask for permission to add the street to your main list. If needed, it will also query you for further information. Then it will transfer the information to a special file, where it holds added streets until you request a process that sorts them into the main list (this is something you should do periodically from the *Streets* form, accessed either from within the File section of the MainMenu or by pressing Ctrl-F5).

By means of this as-you-work process, we expect your StreetList to become increasingly complete—so the frequency of finding a street name missing should be ever more rare. Additionally, as the Census Bureau updates their data every couple of years, it's part of our ongoing-service package to re-do your data with the latest from them. Their data, in itself, is becoming better and better.¹⁸³

¹⁸²There's a fun little thing you can do with this new data. In your DispatchMap, hold down Z on the keyboard (think Z for zips). You'll see each zip display in its population-weighted center. We're not sure of any practical use for this, but it's cool.

¹⁸³If you are served by a local map book and are positively determined to have the best possible StreetList, an alternative would be to contact your book publisher and ask if you can obtain their index in electronic form. If you can provide such a list to us, it's not a big problem to convert it to the format ServiceDesk uses (see page 350). The major drawback is that most publishers want a hefty sum for such a list.

Our final concern is that you may find cities indicated that are either outright wrong or, at least correspond poorly with local custom.

In compiling your StreetList we connect city names to each street based on its zip code. We determine the city name that's applicable to a zip based on a master list that's provided by the Post Office. Too many times, we've found this Post Office-provided data leaves something to be desired (for a given zip it connects a city name that is very inappropriate). Of course, since we're not personally familiar with your area, we don't know in which instances the connection is faulty—although it will likely be very apparent to you. Accordingly, we've created an easy means for you to correct any city-to-zip connections, as reflected in your StreetList, that may be faulty (for a list of city names that were pulled for the zip codes in your area and attached to each StreetList entry per Post Office data, see page 323 in the Appendix; if interested in seeing the city-to-zip connection per this data for any zip code at all, use the Zips utility as discussed at page 218).

Suppose, for example, you notice that all entries from your StreetList with zip code "92675" erroneously show a city name of "MV" (which we'll assume is the abbreviation within your system for "MISSION VIEJO"). In reality, you know this zip (and all its connected streets) actually fall within the city of "SAN JUAN CAPISTRANO." How can this be corrected?

Begin by loading your StreetList form (either select it from the MainMenu or press Ctrl-F5). There click on the button labeled '*Change City Names*'. Then simply follow the prompts. In a matter of seconds, you can have ServiceDesk correct all the entries pertaining to any zip that previously showed an incorrect city name. Then in that respect at list, your StreetList will be perfect forever thereafter.

Our last concern is for listings, as found within your provided StreetList that have no corresponding zip code or city reference at all.

As you're typing in a street name, you may occasionally see a street listing that has *no* corresponding city initials or zip code. It simply happens that the raw data (which we obtain from the Census Bureau) has a number of street listings that lack this information. Prior to August 2000, we did not include these listings in any of the StreetLists we made. Typically, there are very few such listings, so little is lost by their omission. While creating a package for a client in a particularly rural area, however, we discovered that a large percentage of street listings for their region lacked zip and city data, so that if such listings were omitted from *their* list it would have been very much poorer for it.

For this reason we modified our production machinery to include these listings in the client's StreetList. Having done so, we figured there might be some benefit in having such listings included for any client (i.e., better to have a listing with no zip or city than to have no listing at all), and so have maintained the machinery for all packages since.

If the street you are seeking happens to fall into this category, it simply means you'll have to manually type-in the city name, rather than depending on ServiceDesk to do it for you. Of course this assumes that the listing you've selected is genuinely for the street you want, which is less certain than normal because there is initially no city referenced to it. You'll just have to verify by seeing that it displays the job to an appropriate location on your DispatchMap. If not, obviously, you may need to select a different listing or else enter the street to your Callsheet manually.

You might think, as another option, it should be a simple matter to electronically scan the pages of your book, then simply convert the images to computer text using OCR software. In fact, it's not simple at all. Not only is there significant operator time and effort involved in scanning so many pages, more significantly, since these indexes are inevitably done in very small type, the OCR conversion produces a plethora of errors. Plus, there's a potential issue about whether, by merely purchasing a book, you've gained a license to convert and use its index in electronic form. Regardless, we've tried it, and can assure it's *not* a practical solution.

C. Underlying Machinery in the DispatchMap

In the main chapter discussion regarding the DispatchMap, we explained that every scheduled appointment displays to the job in its correct geographical position (see page 83). Actually, the display should always be at least *near* the correct geographical position, but for several practical reasons, it may in fact be skewed slightly, from such an ideal position, in one direction of the other. For any that are interested, we want to here provide an explanation as to the particular reasons why.

The first and most important reason is because, you may have two jobs whose locations are so near one another that, if displayed with the reference precisely over each location's center, the references would overlap, making at least one of the two illegible. For this reason, the map-display intentionally avoids a scheme that involves precise centering of each reference over the job's putative, real-world location. Instead, the system references an invisible, on-screen grid (approximately 32 columns by 23 rows).¹⁸⁴ For any job whose center falls *anywhere within* a given grid square, the reference is displayed squarely within that square. Thus if a job's center was, say, toward the right side within such a grid square, it's reference will end up displaying slightly to the left of a geographically perfect position. While this produces a slight compromise in displayed positional accuracy, it helps to avoid having references write one over the other.

Of course that in itself doesn't guarantee that there won't be write-overs, because you may have more than one job that properly falls within a given grid (which, if each was displayed with only the above-described allowances, would result in a complete writing over of one to the other). Our solution in this instance is positional stacking. Basically, each reference is sized such that it consumes the space of an entire grid horizontally, but only half a grid vertically. Thus, the first job that falls within a given grid is displayed in that grid's top half. The second such job is displayed in it's bottom half. The third (this is not typically common) will be displayed in the top half of the next grid down, and so on in a growing stack as needed to accommodate as many jobs as are targeted for a common grid section. Again, you can see there's a potential reduction in displayed accuracy here, but only as needed to make each display reference legible.

Aside from the demands of legibility affecting positionally-displayed accuracy, there's also the fact that we don't even have perfect positional information in the first place. We have a grid reference, and in fact the location may be anywhere within that grid. When the Dispatch goes to position its display reference for the job, it only has that grid reference, and thus does not know (with any greater precision than that) where the job is located.

Then there's a question of resolution. Ideally, we will have preferred to design your system so there is a one-to-one relationship between grid sections in your paper source map (and the corresponding references connected to each job) and the invisible grid built-in to our DispatchMap. However, if your territory is somewhat large, this may have been impractical (because if we did so scale it, you'd only be able to see a tiny fraction of your territory in one DispatchMap screen, and so would have to pan ridiculously to survey its entirety). Accordingly, there might be a two-to-one or even three-to-one relationship (meaning, in the last case for example, that one on-screen grid represents the same territory as a three-by-three grid section in your paper map). We mention it because, if this is the case, you'll find that even when two streets have different grid references, they may still display in precisely the same location on the DispatchMap. As a matter of scaling, it simply is not practical to do it in any other manner.

¹⁸⁴ If you want to actually see this grid, incidentally, it's easy. Just left-click and hold (with your mouse button) anywhere on the map that does not have operative text on it. The grid will display with a an obvious cross-hatch of bright red lines.

While all of this concerns the positioning of scheduled appointments (or rather their references) in the DispatchMap, there is another, unrelated matter of potential inaccuracy we want to address. It concerns the DispatchMap feature that allows you to view past day's schedules.

Basically, though this function should be quite reliable, it may in some extreme circumstances fail accurately display all the appointments from particular days in the past. If you're interested, we'll explain why.

First of all, accomplishing the purpose of viewing past appointments is not so easy as you might assume. Until this feature was added, the Map derived *all* its data only from the current ScheduleList. If this list is properly maintained, it contains only items scheduled for the present and future. Typically, this will not be more than several score of records at a time. Thus, to decide what to display for a given day (current or prospective), the machinery has only to review this small number of entries, and see what is applicable for that day.

To show past days, by contrast, the Map must refer instead to the ScheduleList-Archive, and this is a file that eventually becomes very large, containing tens of thousands of records. Because of this large size, it's not practical (in terms of promoting a prompt display) to have ServiceDesk search through every record (as it does in the current list) to find items pertaining the particular day that's requested for display. So another means is required. Basically, it's this. As you page up or down among past day's within the Map, ServiceDesk searches forward or backward in the file, as the case may be, only so far as seems necessary to probably account for all entries pertaining to the day in question. The caveat is that the algorithms by which we accomplish this are limited in scope, which means that in some extraordinary circumstances the system may fail to accurately display all jobs as scheduled for days in the past..

You'll also notice that the DispatchMap can display jobs from past days only up to the date when the ScheduleList form's Archive routine was last run. This is because they have not yet been moved out of the current ScheduleList and into the Archive—and it is only from within the latter that the DispatchMap looks for jobs in the past (whether a day is past or not being reckoned according to the computer's internal calendar).

D. How ServiceDesk Interprets Appointment Notations

There are many different ways in which a human operator might designate an appointment for a certain date and time. Being intelligent as we are, we probably could decipher the intent behind any format that makes reasonable sense in our own language. Unfortunately, computers and software are still not so smart as we when it comes to matters seemingly so simple. It is fortunate in this regard that, in respect to much of what you type into a Callsheet or other form, there is little need for ServiceDesk to *understand* what's there: for the most part, it simply records what you've entered and transmits or displays it, as appropriate.¹⁸⁵ In

¹⁸⁵ Actually, there *is* some interpretation in a few other boxes (see also, page 87). In regard to the two name boxes, for example (pertaining to both Customer and ServiceLocation), ServiceDesk looks for a comma. If it sees one, it assumes that everything preceding must be the customer's last name, and that any word or phrase immediately following (with only single spaces between each word) must be the first name. Otherwise (i.e., if there is no comma), it logically figures that it must be looking at a business name (where no comma-in-between inversion of words would normally be used), at least in regard to any succession of words beginning at the front of the line, with each such word separated by no more than one space at a time. If you've used a larger group of spaces, ServiceDesk figures everything to the right of such a group must be some non-name or other supplemental information you've included in the line. Of course, ServiceDesk also looks to see if there's a 'HighVolume' client's name in the CustomerName box (as defined in the QuickEntries form), and finally, it also looks for an open and closing bracket with characters between (i.e., "[\${@%}]") in each of a Callsheet's address lines, assuming if such are found that the intervening characters must be the job's map grid reference, which likewise must be interpreted according to a particularized format that fits with your own in-use map. ServiceDesk makes all these interpretations, according to such conventions, to free you from the burden of having to specify such details explicitly.

regard to what's entered in the 'Appointment Date and Time' box, however (and later transferred to a job's appointment reference in the ScheduleList), ServiceDesk must be able to make a sensible correlation between what's entered and the actual calendar context of date and time. For this reason, it's necessary to use a format that ServiceDesk, being relatively dumb in such matters, is prepared to interpret.

Specifically (but with some exceptions, to be discussed), ServiceDesk is programmed to expect three particular "words" within any appointment reference. It defines words as adjoining groups of characters, each separated by a single space (i.e., it would think "**o8kl3 v9f drt**" is three words).

For the first word, ServiceDesk expects to find either a single number referring to the desired day-of-the month (i.e., a number in Arabic text between 1 and 31) or an 'X/X' -type of number/number combination that refers to the month *and* day-of-month (as in "**12/31**" or "**2/19**").

For the second word, it expects either a weekday name, or as many leading characters from such a name as you may choose to use (e.g., "**MON**").

Finally, for the third word, it expects any of the following:

- a time (e.g., "**9:00**" or "**3:45**");
- a pair of times separated by a hyphen (e.g., "**9-12**" or "**1-3:30**");
- a simple "**AM**" or "**PM**"; or
- an exclamation symbol ("!") to indicate no specific time has been committed to (see page 99).

No other forms of text (such as, for example, "**1stCALL**") are acceptable for this third-word position.

Thus, a complete appointment entry could look like any of the following:

20 THURSDAY 9:45
1 TU 3-5:30 C/F
7/15 MON 9-12
5 SUN 7:00 EMRGNCY
13 FRIDAY !
2 WED AM 1stCall

Notice that in some of these examples there are additional words after the first and magic three (we even have an example where "1stCall" is acceptable as a *fourth* word). ServiceDesk doesn't mind such additional words (which you can use internally as notes). It simply *must* see the magic first three words — each in one of the described-acceptable formats.¹⁸⁶

To make things easier for you, ServiceDesk does manage some minimal intelligence in regard to interpreting these entries. Notice, for example, that you are not expected to specify 'AM' or 'PM' in regard to the time (nor to use military time). ServiceDesk will assume that a 9-12 appointment must be for the morning,

¹⁸⁶ There are three potential exceptions, in terms of ServiceDesk not paying attention to what comes after those first three words. One is that if you place a plus or minus symbol as the *fourth* word (i.e., the full appointment entry might be "**13 FRI 9-12 +**" or "**13 FRI 10-1 -**"), ServiceDesk will either add or subtract one hour to the stated time-value when auto-sorting your jobs (for more information, see page 112). Or you can add an exclamation as the *fourth* word (e.g., "**13 FRI 9-12 !**") if you want to prevent its sequence from being altered at all by the auto-sort process (*ibid.*). Finally, you may add an AM or PM designation as a *fourth* word (e.g., "**13 FRI 5-6 AM**") if wanting to *override* ServiceDesk's normal AM/PM assumptions (see *note* following).

and that a 3-6 must be for the afternoon. It becomes more difficult for this distinction, obviously, as your appointments reach the more extreme edges of the day (is an appointment designated for "7:00" more likely to be am or pm, for example?), so ServiceDesk establishes a simple dividing line. If the number designating a single time, or the average between a pair of times (i.e., 10:30 when "9-12" is stated) is 6:59 or less, ServiceDesk assumes it must reference a 'PM' time. If it is 7:00 or greater, ServiceDesk assumes you're intending 'AM' time.¹⁸⁷

Additionally, when allowing you to state merely a *single number* for the day of the month, ServiceDesk must somehow deduce which month and year you're referring to. Essentially, it seeks and finds the last time when such a number arose in the calendar, as compared to the present date. If the found date-number was less than five days previous, it assumes that as the date referenced. Otherwise it assumes that your number refers to the next time the number arises in the calendar.¹⁸⁸

While it's convenient to be able to state a single number for the date, you can readily see (based on the above) that if you're wanting to specify a date that's more than about 25 days hence (depending on number of days in the current month), it would be impossible, on the basis of a single number alone, for ServiceDesk to accurately interpret your intent (the described assumptions would fail). For that reason, and for those specific instances where you're specifying an appointment further into the future than 25 days, you'll find that both auto-insertion methods will create a more complete reference (e.g., "12/31 THU" rather than merely "31 THU"). In this instance, obviously, there is still the need for ServiceDesk to infer the year. Roughly, it uses the same convention as when inferring both month and year, but in this case will correctly decipher your intent so long as you're not attempting to schedule more than about 51 weeks into the future (or again, more than five days in the past).

It may seem odd when first considered to find we've set it up so that you need put in only a single, day-of-the-month number for the date, rather than a more standard format such as "12/31/01 9-12". The reason is because we also want a reference to the day-of-the-week (for verification of intent as described in the following paragraph, and related reasons), and yet we want to keep the appointment reference relatively short (a reference such as "12/31/01 MON 9-12" would obviously be unwieldy). Thus we find something like "31 MON 9-12" is generally most effective, particularly because the applicable month and year should in virtually all cases be obvious from context. Regardless, you'll note that you can select an option in the Settings form that will cause the month indication to be included in all instances, if that is your preference.

As for as the second-word, day-of-the-week portion of your entry, ServiceDesk uses it solely for the purpose of verifying you have not, in some manner, goofed in regard to the intended day of the appointment (it checks on this during the Job Creation process). Thus, if you've specified "16 FRI 1-4", but the following Friday is, in fact, the 17th, ServiceDesk will alert you to the error and require correction—thus helping you

¹⁸⁷ In the event you've got a job for which these assumptions fail (e.g., you've scheduled an 8:00 PM job, yet "8:00" is a time for which ServiceDesk would *normally* assume AM), the solution is for you to go ahead, in *this particular kind of situation*, and type an appropriate AM or PM designation as a *fourth* word in your appointment description. In this situation, ServiceDesk will recognize the added spec, and override its normal assumptions (remember that a space between the time and AM/PM is required, as in "13 FRI 8:00 PM", for example—not "13 FRI 8:00PM"). In fact, you may notice use of the "AM" or "PM" is *permissive* in all cases—but is *needed* only in those rare circumstances where you're scheduling a job outside the typical time-range, thus causing the normal AM/PM assumptions to fail. Specifically, in any case where you're scheduling a time (or time-range with an averaged value) that falls within the 7:00 AM to 6:59 PM range, you can rely on ServiceDesk's assumptions. It's only when wanting to specify something *outside* that range that you must add an AM or PM to communicate your intent.

¹⁸⁸ One consequence of this assumptive process is that you have a maximum of five days to clear past items out of your ScheduleList (just run the Archive routine from the ScheduleList form on a daily basis). If you wait longer than five days, ServiceDesk will end up misinterpreting all older-than-five-day items, still there, as being prospective. It has facilities for dealing with this (and will inform you when the situation arises), but we note it here so you can understand the cause.

avoid a very common mistake. Indeed, ServiceDesk checks the entire format during the Job/Sale process, and will coach you in making necessary corrections.

E. How ServiceDesk Collects Information for a Finished-Form Document

If you become involved in making the formalized Finished-Form documents that are part of the system described in Chapter 8C (see page 217), you may find it profitable to know some of the more technical details regarding how we've setup the system. Describing those matters is the purpose of this section.

In general, it is ServiceDesk's job as part of this system to peruse through the various and sundry files it has compiled, in reference to a particular job, and attempt from there to glean what's relevant for inclusion into each of the various text boxes of the form you are preparing. It is further its job, obviously, to attempt to place each item of information into the particular textual format that you're likely to want for the box into which the item is being inserted. If you know some details about this process, it will help you understand the results you see, and perhaps help you plan better toward getting the results you want.

First, in regard to the name, address and telephone numbers, you'll notice these are very straightforward, with information being pulled simply from text within the item's JobRecord. The one complication is that, for the Narda form, there's only spaces for one info-set (i.e., not separate sections for both the paying party and location info). As a convention for the Narda form, ServiceDesk places the name and address set here that's found in the 'Location Info' section of the JobRecord. However, in the event it does not find you've used that section for an apparent name and address (a matter it deduces by looking for a comma in the LocationName line and an opening bracket in the LocationAddress line), it inserts text from the JobRecord's Customer-Info section instead. (Since the Generic form has boxes corresponding to both the Customer and LocationInfo section of the JobRecord, there are obviously none of these considerations in reference to text inserting to it.)

In regard to the P.O. Number, if there's one present in the item's JobRecord, ServiceDesk will insert it to the "Special Authorization #" space in the Narda form, and place an "X" in the corresponding checkbox for you. The "Customer's Request" information, as inserted to the form, is obviously transferred directly from the JobRecord's corresponding "Problem to be solved and/or Description of Customer's Request" box, the "Date Call Received" text is based on the 'Origin Date' for the JobRecord, and so on. Most of that is pretty straightforward.

For the inserted "Completion Date," ServiceDesk examines the History for the job and looks for an entry with the phrase "job completed." Assuming such an entry is found (ServiceDesk will have created precisely this kind of entry on completion of any PostVisitReport in which the query "Is the job done?" was answered in the affirmative), the date of such entry will be inserted for you.

For the blocks describing times when the tech started and ended on each visit, ServiceDesk again examines the History for the job. Specifically, it looks for the kind of entry that's created when, in conjunction with a standard, PostVisitReport, the technician indicates what the start and end times, during his visit, happened to be. Finding such appropriate descriptive text, the system gleans the actual times stated, and inserts them into the form for you. What it looks for, specifically, is an entry beginning with standard date and time, then a tech's initials followed by the key word " there ", then an appointment reference, a comma, then

two times separated by the key word “ to “, then another comma and more text (as in “3/4/00 16:42: DS there 14 TUE, 11:40 to 12:30, diagnosed bad valve . . .”). If you’ve made any manual changes in the job’s History and corrupted this precise kind of format, you may find that ServiceDesk fails to winnow out the start and end times as wanted.

It’s also by reading these particular entries in the History, you should know, that ServiceDesk deduces which technician’s name should be inserted into the Narda form space that calls for one. At least, this is conditionally the case. But in fact, you may have a job on which different technicians made separate visits. If that’s the case, ServiceDesk chooses the name of the technician on the last indicated visit (at least in respect to the last entry it is able to successfully interpret as such). But it’s still conditional. If (as we figure will usually be the case) there’s a SalesJournal entry for the job, the system will choose the technician indicated in conjunction with it (always bear in mind that, regardless of what’s inserted for you, you can change any of the form’s text boxes to what is specifically wanted at the moment).

Still another matter concerns the description of work performed (i.e., the space specifically labeled “Service Performed” on the Narda sheet). In this case, there may be much text in the job’s History that is not a specific description of work the technician actually did, and obviously ServiceDesk lacks the genuine intelligence that would be needed to separate and winnow out such completely descriptive words from other text. Instead, it does the next best thing. Basically, it identifies entries made in a PostVisitReport by looking for the same key phrases as described above (i.e., it looks for references to the times when a tech started and ended his visit). It then assumes that at least most of the language following the preliminaries in such an entry is probably a description of what the technician did, and so culls this specific text and places it into the relevant box on your form. If there were multiple visits, it includes the text from each, while placing a pipe symbol (i.e., “[|]”) between each one to help you see the separation. We expect this is one box (on the form that is) that you’ll edit in virtually every instance, but it should be very easy. In most cases it will probably be a matter of using your mouse to highlight extraneous text, then hit delete, for maybe a two or three second operation overall.

The last *major* matter in regard to which the system reads the job’s History is in regard to parts used from stock. It depends entirely on descriptive language in the History for this purpose, so be sure all stock-used is properly reported through the PostVisitReport system, so that such proper descriptions will be there for this purpose (and again, bear in mind that if you’ve previously changed the ServiceDesk-inserted History text so that’s it’s now different from the expected format, ServiceDesk may fail to glean the needed meaning). Specifically, to deduce that particular language in a History refers to parts-stock used, ServiceDesk looks for the word “used”, followed by any two words, then the words “from stock” (as in “*used 1 3363394 from stock*”). On finding this kind of a phrase, ServiceDesk deduces the quantity and part number of item used, and makes corresponding entries in the form for you (actually, there’s more work to do in figuring the prices that should be inserted for such parts, as described later in this section).

Several of the Narda boxes refer to information specifically about the machine that was serviced (i.e., make, type, model, serial, purchase date and selling dealer). As you might guess, this information is taken directly from any UnitInfoSheet that’s attached to the job, with each such item of information inserted to the appropriate box on the Narda form (or Generic form, for those boxes that are applicable).

The Narda form also includes, you might notice, boxes for a “Servicer Number” and “Servicer State Number.” For this data, ServiceDesk looks to the QuickEntry template for whichever client is in the CustomerInfo block of the item’s JobRecord. In other words, if you have a QuickEntry template for Whirlpool, say, and the job has Whirlpool listed as the client (i.e., it’s their name and address in the CustomerInfo block of the JobRecord), ServiceDesk will look to the QuickEntry template that you’ve created for Whirlpool. If it finds

that you've listed a Servicer ID and Servicer State # in the spaces there provided, it will insert those same numbers into appropriate spaces on the Narda form.

Also on the Narda form there's a space for a "Service Agreement Number." Basically, if you've indicated a "Contract Number" in the item's JobRecord (like many other items in the JobRecord, this will typically have first been entered from the job's initiating Callsheet), that's the number ServiceDesk will place into this box for you (see page 59). Also, you may note that besides its own pre-printed Number in the upper-right corner, the Narda form has an open space directly below. This is for your own, internal job-identification number—which of course in ServiceDesk we refer to simply as the InvoiceNumber. Thus, ServiceDesk will place its own InvoiceNumber, for the job, in this location, and print it to the form. This will be useful, obviously, to help you refer back to the job when a Narda claim is returned, paid, or there's any similar action.

There are some other, miscellaneous boxes on the Narda form which ServiceDesk makes no effort to fill. If they need filled, you'll have to do it yourself, manually—which should be little effort, considering how few are left.

Aside from those many miscellaneous boxes, there are of course two large, multi-box sections remaining, in both the Generic and Narda forms. These are: first, the set of boxes that describe parts used; and second, the set where charges are totaled.

In regard to parts used, we've already mentioned that, for parts used from stock, the system deduces such usage by reading through the job's History (see several paragraphs back). That, of course, explains how it deduces how many of which kind of parts were used (which it sensibly places into the appropriate boxes on your form). It does not explain how it deduces what *price*, for the parts, should be inserted in boxes indicating such. For this deduction, ServiceDesk looks into either: (a) your MasterPartsPlan; or (b) your Journal of Stock Transfers (see page 144). Specifically, if loading info to the Generic form, the system figures you must intend *retail* price for parts used, and so gets this price from the item's entry (which should include expected retail price) in the MasterPartsPlan. If, on the other hand, it's loading info to the Narda form, we expect that you probably need to list *your cost* on the part (since, as a rule, that's what Narda-type clients demand to have listed). So in this case, the system finds the most recent Journal entry showing use of the part, and inserts the price paid (as indicated in that entry) for the part.

While that describes the situation in regard to stocking parts used, it's obviously a different matter when dealing with parts special-ordered for a job. Here, it's actually a bit more simple. ServiceDesk simply checks the PartsProcess files (both Current and Archived) for entries that correspond with the invoice number of the job in question. For any such entries that show a date in the 'Date Received' box, the system figures you presumably must have used the part, and so inserts its description, part number and so on into appropriate boxes of the form. For price to insert, it looks for a value in the same entry's '\$ Sold' box, and inserts this into the appropriate space on your form. For Distributor Invoice Number, it looks in the 'Notes' section of the entry. If, as the first "word" in this entry, it finds text that's all numeric (i.e., "1568556" as opposed to "C1568556") it concludes that must be the invoice number on which you purchased the part, and inserts it to the NARDA as such. Regardless and in any case, some editing may sometimes be required. If for example a wrong part was ordered and received, the system may erroneously include its entry, as though used, within your form. Alternatively, a part may have been ordered, used and received without the expected processing through the PartsProcess system. If this occurred, it's entry will not be found and inserted for you. You'll have to be watchful for such possibilities, and edit from within the form accordingly.

Finally, we are concerned with how ServiceDesk fills the various boxes in either of the forms' totaled items columns.

In the Narda form, you may notice, there's a box (under the column of prices for parts used) labeled "Sub Total". Here, and with obvious logic, the system simply places the total of the various parts prices that are listed above.

Aside from that simple matter, and in its last reading of the job's History, the system looks to see if there's an entry there, as is made by the system when the job is finally recorded to the SalesJournal (see page 167). Specifically, it looks for this precise text "*ttl \$*" (as appears as part of an entry such as "*3/15/00 13:04 BN: rcrdd blld cmplt in SlsJrnl, ttl \$117.54*"). Assuming that it finds that little snippet, the system figures you've indeed made a SalesJournal entry on the job, and that it should be able to find such an entry within your SalesJournal. Thus, it makes a search therein. Assuming that it finds such an entry, it pulls the amounts indicated and places them into appropriate spaces on the form.

A question, in this regard, is what should the system do if it finds a parts total, as indicated in the SalesJournal entry, that differs from the total as deduced by adding the prices for parts as listed? To which figure should it give deference as values are inserted to the form? In the case of the Narda form, there's a box for you to fill-in your "Handling" fee on parts used. Thus, for this form only we figured it was sensible to deduce that any difference, between parts total as indicated in the SalesJournal entry and parts total as derived by adding actual parts, must be a handling fee, and that value, if any (i.e., the difference between) is inserted to this box for you. In the case of the Generic form, no such assumption seems sensible (nor is there a separate box for a so-called handling fee). So, if there is a discrepancy in this case, the system alerts you with a descriptive warning, and for the time-being places a row or asterisks in the form's PartsTotal box.

As you edit in either the parts description boxes or the totaled item boxes, you'll notice that the system automatically does the appropriate math to adjust the carryovers for you. As we've indicated elsewhere, however, the system is new, so it will not be too surprising if you discover imperfections. If so, please let us know.

A final caveat concerns the searches that are done within the PartsProcess Archive (to find parts special ordered) and the SalesJournal (to find totals charged). In respect to the first (and to conserve time searching), the system looks back no further than 300 records deep into the Archive (of course it searches the current PartsProcess file in full). In respect to the SalesJournal, it searches no more than 1000 records back. In consequence, if you're trying to load data into a form from a job that was completed some significant time ago, you may find the particular items of data which correspond with these searches do not automatically fill-in for you. Of course, you can still type-in the relevant information manually. It's just not as easy or convenient.

F. Dealing With Divergent Sales Tax Rates

Most of our clients face a uniform set of sales tax rates (i.e., one rate for parts and another for labor, regardless of locale) across their entire territory. This makes for nice simplicity, as typically the worst complication is having to deal with an occasional tax-exempt sale (see note 121 at page 167). However, we've had some clients who face the ugly dilemma of different tax rates in each of the various communities they service, and sometimes different rates for different *kinds* of customers too. This technical section contains information on how that kind of dilemma should be addressed from within ServiceDesk (if yours is not such a situation, obviously, this section may be ignored).

To start with, let us review the standard situation, where rates are uniform throughout a user's territory. In that instance, it would be your job to first specify the applicable rates (for parts and labor) from within the Settings form. In consequence of this specification, the system would perform a verifying calculus each time a completed sale was entered from the SalesEnter form. Specifically, each time you entered the string of numbers that indicates constituents in the sale and total, it would do its own calculation to verify that, with appropriate tax amounts added (as inferred on the basis of rates indicated in the Settings form), the entered total reflected the proper summed amount. If the numbers did not balance, of course, the system would notify of the discrepancy and request correction.

So what can be done for the user that faces many different rates, depending on locale where the service was done? If, as is allowed by the Settings form, just one rate set were specified by this kind of client (i.e., one for parts and another for labor), the SalesEnter system would obviously not allow any sales to be reported—much less recorded—at any different rate. A real problem. Obviously, in solution we *could* create a more expansive mode for specifying tax rates (more expansive, in other words, than is currently offered in the Settings form)—perhaps a table, for example, that would allow you an unlimited number of rows where zip codes could be listed in the first column, and applicable parts and labor tax rates (for each zip area) in the second and third columns. However, and as mentioned, some of these same clients have faced rates that vary depending on type of customer, too. Setting up, additionally, with specifications for that would require a three-dimensional table, and who knows what kind of flexibility for other clients facing still different divergent tax needs. After all was considered, we thought it best to keep the specification system simple for the ordinary ServiceDesk user that faces just a single tax structure, and make separate allowance for the user with special needs. That is what we have done.

The solution is implemented via an option in the Settings form labeled '*Allow for divergent sales tax rates depending on locality, type, etc.*' Once this option is enabled (click on it then save), you'll find that the boxes for specifying a uniform tax rate are, logically, *disabled*. And now, when you go to enter completed sales from within the SalesEnter form, you'll discover the system is much more permissive. Yes, it will still sum the entered constituents and compare that result to the entered total, but now it will allow an assumed tax rate (applied to all constituent parts) ranging from 0 to 15 percent. Thus, so long as the entered total is at least the sum of the entered constituents, and is no more than 15 percent above, the entry will be accepted and recording allowed. Whatever cushion it finds, if any (ranging from 0 to 15 percent), will be assumed as the tax amount that was appropriately charged on the sale.

All that is lost vis-à-vis the standard system, at this point, is that if the entering operator makes an error (or the person who added the invoice did) that nevertheless keeps the constituents and entered total within the allowable comparison/tax-rate range, that particular kind of somewhat smallish error (i.e., maximum of 15 percent either improperly credited, or not credited, to tax) will not be caught by ServiceDesk.

So far as operations that are integral to ServiceDesk, this describes the solution completely. With that and nothing more, ServiceDesk will accurately collect sales information, and stand ready to report on *system-wide* tax liabilities (among other figures) for any period wanted.

Of course, as the divergent-tax-situation user you'll face an important further need. You still must *separate* your tax liabilities between and among the various jurisdiction in which you've done service—so you can report on those liabilities and send payments as required. While the ServiceDesk SalesJournal will show the tax amounts you've calculated on each job (regardless of locality), it does not separately indicate the location where each job was done, nor is there presently provision within ServiceDesk (even if the SalesJournal did include such information) to break out sales and tax liability on the basis of locality. Again, we do not want complicate the general system unduly for the normal user.

Instead, as another consequence of specifying that *'Allow for divergent sales tax rates depending on locality, type, etc.'* option in the Settings form, the system will silently create a mirrored SalesJournal file for your own special use. Specifically, each time you enter a completed sale via the SalesEnter form, the system will now make not only the standard entry into the ordinary SalesJournal, but also another entry into this separate file. And this entry, unlike those in the standard SalesJournal, will include the zip code for the job in question (along with department if you're using that option). And, unlike the standard SalesJournal, this file is in a format you can easily access from any spreadsheet or database program (i.e., it's in Ascii, comma-delimited format). You'll find the file at `c:\sld\netdata\sjsjrn1.str` (assuming c: is your FileServer; otherwise specify the appropriate drive in place of c:).

Thus, ServiceDesk collects all the raw data that's needed for you to break out your sales tax liabilities between and among different jurisdictions, and compiles it in this one file. It's left to you¹⁸⁹ to import this file/data into an appropriate context (most likely a spreadsheet program such as Excel) and there create a system that, using the data, makes reports concerning each of your separate liabilities.

Some caveats. We do not use this system in our office, and have had little feedback to date from users. As presently written, entries within the secondary SalesJournal are created only via the SalesEnter system. If you make *revisions* to the SalesJournal from the SalesView form, they will not automatically be reflected in the secondary journal (if important, you should mirror them manually from whatever spreadsheet or similar context you've setup for manipulating the file). And entries made while reporting on payments to or changes in accounts receivable are not at this time programmed for addition to the secondary journal. If you find these operations are important to your use, please let us know and we'll be happy to accommodate.

G. Resolving Possible Right-Click Problems

You'll probably never have the slightest need for information in this section, and if you do it will have become apparent from other contexts that will have cross-referenced you to here. So, if you're just reading through, please don't bother reading further here.

The reason for this discussion is because we've occasionally had trouble getting various of the ServiceDesk-intended right-click functions to operate when they involve a textbox. As an example: right-clicking in a Callsheet address box to Item-Locate to the DispatchMap. The reason: beginning with Windows 95, Windows itself uses the textbox-based right-click to enable a set of editing features. This can potentially interfere with ServiceDesk using the operation for its own purposes. However, we believe we've solved the problem in all contexts that it formerly existed. We provide the explanation here, and discussion of possible alternative methods, in case we are wrong and, in fact, you encounter any such difficulties.

In specific regard to right-clicking on a Callsheet address line to item-locate, you may instead double-left-click thereon, or right-click on the label that identifies the textual area which includes the address line (i.e.,

¹⁸⁹ Thanks to the clever thinking of Tara at Alpha Omega Service in San Antonio, here's an alternative idea (and one that takes advantage of structures that already exist within ServiceDesk). If you don't otherwise have a need to departmentalize your sales (see page 189), you can instead use the departmentalization feature to create, essentially, different "departments" for each of the different jurisdictions, within your territory, for which you must collect and report on tax. Basically, you just designate the appropriate jurisdiction/"department" at the time of creating each job (after turning that option 'on' from within the Settings form, of course). ServiceDesk keeps track of this designation, and is already equipped to separate the various figures in your sales report (including sales tax liabilities) into each of the various "departments" (in this case jurisdictions). Thus, if you use this strategy, there's no need for you to create any kind of separate reports on your own—so it should be much easier. Thanks, Tara, for the clever thinking.

either on the label that reads "Customer Name, Address and Phone Numbers" or on the one reading "Location Name, Address and Phone Numbers"). Any such action will produce the same result. Similarly, in the alternative to right-clicking on any of a Callsheet's telephone number fields to auto-dial, you may instead double-left-click thereon, or you may right-click in the open space immediately below. The same holds true for equivalent actions within any JobRecord as viewed from the JobsCurrent form.

A trio of alternative methods for the same functions may seem odd, but there's a historical reason for it. When first encountering the problem with introduction of Windows 95, we quickly put in the double-click option. But we dislike the double-click for actions that are very frequent (it's just more effort), so before long we added the alternative right-click methods that involve clicking on non-textbox areas. Then, over an even longer time, we developed solutions to make the original right-click-on-the-textbox operations work. But still, we've kept the alternatives in place just in case—and because there's a number of clients who learned ServiceDesk when double-clicking was the only method. We don't want to force them into re-learning old methods.

H. Interfacing with Your System of Financial Accounting

As mentioned elsewhere, it is not intended that ServiceDesk will replace whatever system you have used (or, if you're a new company, *will* use) for writing checks, documenting miscellaneous operating expenses, figuring depreciation, and compiling financial reports such as Income Statements and Balance Sheets. It is not designed, in other words, to fulfill the functions involved in *financial* accounting—at least not mostly. However, you should note that it does, nevertheless, collect and process and compile several of the items of information that will need to be inputted into whatever method of financial accounting you otherwise use (primarily it collects information that's associated with the *revenue* side of accounting). In particular, it collects and compiles information on your sales, accounts receivable, funds collected, sales tax liabilities and inventory (this last being an exception from its "revenue-side" focus). This information will need to be inputted into your system of financial accounting as often as you compile financial statements (most likely once every month, but perhaps only once per year, depending on your company's preference and policy).

i. Basic Concepts

With but one exception, your ServiceDesk *Sales Report* (see page 172) collects and compiles for you all the information that needs to be moved from ServiceDesk into your separate system of financial-accounting. It even blue-highlights, for you, the particular figures that need to be so moved:

Business Reports

TABULATION OF SALES TOTALS

	Total of jobs completed in period	Total of jobs paid for in period	Total of jobs completed but billed	Total prevsly done and now paid
Merchandise Sold	0.00	0.00	0.00	0.00
Parts Sold	39,185.23	11,518.67	27,666.56	0.00
Total of goods sold	39,185.23	11,518.67	27,666.56	0.00
ServiceCalls Sold	1,299.59	1,299.59	0.00	0.00
Additional Labor Sold	72,133.52	26,218.52	46,915.00	0.00
Total of labor sold	73,433.11	26,518.11	46,915.00	0.00
Total Sales	112,618.34	38,036.78	74,581.56	0.00
Tax on sales	727.86	719.97	7.89	0.00
Gross Invoice Totals	113,346.20	38,756.75	74,589.45	0.00
Quantity of S. Calls:	537	9	528	0
Quantity of Invoices:	749	195	554	0
Qty of zero-sale tickets:	72	72	0	0
Average charges per S.Call:	209.72	4226.31	141.25	N/A
Average charges per Invoice:	150.36	195.06	134.62	N/A
Ratio of total Labor to Sales:	65.2	69.7	62.9	N/A
Work-Out to determine True Change in Total Dollars tied up in Outstanding A/Rs:				
Begin with SalesJournal-based figure (total of PayCode 2s minus 3s, as per above):				\$74,589.45
Subtract for change in balance of A/R Partial Payments (i.e., Paid-To-Dates):				\$48.00
Subtract for A/R Bad Debt Writeoffs:				\$0.00
True, Net Change in Balance of Outstanding A/Rs:				\$74,540.85
Work-Out to determine True Net-of-Money-Received:				
Begin with SalesJournal-based figure (total of PayCode 1s plus 3s, as per above):				\$38,756.75
Add for Changes in Pre-Pays (i.e., any amount rec'd prior to SlsJrnl entry):				\$1,475.32
Add for Changes in balance of A/R Partial Payments:				\$48.00
Add for EDA Movements (i.e., fudge amounts as used to match payments to A/Rs):				\$0.00
Subtract for Funds Lost, Stolen, or Otherwise Unredeemable (i.e., PayCode 6s):				\$0.00
Subtract for Discounts Granted (i.e., for accelerated payment):				\$0.00
Subtract for Parts Credits (since they don't actually represent money rec'd):				\$0.00
True, Net-of-Money-Received:				\$40,281.27
Based on your provision of a TaxRatesFile, exempt sales should be determined via data export. (Blue indicates the value is needed by your system of external accounting)				

Type of Report

☒ Sales Summary
 ☐ Accounts Receivable
 ☐ Payroll
 ☐ Profitability
 ☐ Performance - Clients
 ☐ Performance - Techs

Output Target

☒ Monitor
 ☐ Printer

1/1

to

1/31

Export to QuickBooks

Export with Extended Data

Produce Report

On-Line Handbook

As you can see, there are only a few such figures. These figures concern, in particular, items related to the *revenue* side of your accounting, and exclude the one *cost*-side element that must be handled separately (specifically, cost-of-goods sold). The figures are such that, if you use an outside accountant and simply provide this report to him or her, he or she should understand precisely how to handle it.

Regardless of whether you do such a simple hand-off to another party or not, the general concept is you will run this report at the end of each accounting period — so as to provide a summary of the activity that occurred during that period, for feeding into your full-accounting system.

If you do not in fact employ an outside accountant, you will need to do that feeding-into-full-accounting-system process yourself. If your full-accounting system is QuickBooks, you may notice there is an “Export to QuickBooks” button (see above illustration, and look for the hot-pink colored button toward its lower-right). It’s designed to do that “feeding into” process for you. If you use some system other than QuickBooks, you’ll need to do the “feeding into” manually. Since there are only a few figures, there is little labor involved. However, you are likely to need conceptual assistance, which we’ll here provide.

Of course, there is that other, expense-side element (cost-of-goods-sold), and it must be dealt with as well. In the following sections, we’ll discuss each of these matters in greater detail.

i. Integration with QuickBooks

We're discussing this first, because the vast majority of ServiceDesk users (and of North American small businesses in general) use QuickBooks.

Aside from that single, expense-side, cost-of-goods element, integration with QuickBooks is eminently simple.

Entering the Standard "Revenue-Side" Data, into QuickBooks

Just run that *Sales Report* (as above described) at the conclusion of each accounting period, then click on the button to *Export to QuickBooks*. Then, from within QuickBooks, click on *File*, then *Import*, then pick the file you created by exporting from ServiceDesk.

In result of the above, QuickBooks will create a set of internal accounts, each designed to appropriately receive the same values as are blue-highlighted on the face of ServiceDesk's Sales Report. And it will make internal Journal entries, inserting values as applicable to each such account.

As an example, QuickBooks will create an internal account called "*SD-Tallied Sales*." When you then run an Income-Statement report from within QuickBooks (and for a period that includes when you did such an export/import data movement), the report will show (as a direct-identified Sales-Income item) the appropriate amount from SD-Tallied Sales.

For the main part of your joining ServiceDesk-to-QuickBooks work, all you must do is the above. It is truly that simple.

Entering Bank Deposits, into QuickBooks

One small detail involves entering into QuickBooks when you've made a bank deposit — specifically, of monies that ServiceDesk managed the collection on (hopefully, you will have indeed used ServiceDesk's deposit-preparation machinery to manage building the deposit tape, and to internally document each item's disposition, etc., but this discussion remains relevant to how the entry needs to be made into QuickBooks, regardless). When you enter a bank deposit into QuickBooks, QuickBooks will demand to know the source of the money being deposited (i.e., the "account" that should be credited as the source). Where the money was indeed managed via ServiceDesk mechanisms, you should indicate the source as "*ServiceDesk-Tallied Funds Received*" (this is another of the accounts that the ServiceDesk-to-QuickBooks export-import process will have created for you).

As you do the above, you may note a small oddity. It stems from the fact that you will typically be making these deposits on a periodic basis throughout any accounting period, yet the transfer of reckoning from ServiceDesk to QuickBooks (i.e., ServiceDesk telling QuickBooks what funds it tallies itself as having received during a period) will not occur until the end of that period. For example, suppose that in a given month you collect \$40,000. In each of four Fridays of that month, you deposit \$10,000 of that \$40,000. Each time you make such a deposit, you claim the funds came from "*ServiceDesk-Tallied Funds Received*," yet it's not until afterward that the entry of \$40,000 is made into that account. For this reason, it is normal that, during the course of any accounting period, your deposits will have the effect of drawing that account into the negative. It will naturally be brought back to zero by virtue of the export-import transfer, at the end of the period. This is normal.

Entering Cost-of-Goods-Sold, into QuickBooks

The cost of your parts used (and of merchandise sold, if you are also a dealer) is the one element of financial accounting information (and as managed by ServiceDesk) that is not on the revenue side, and which requires handling outside the Sales-Summary and/or its export-import movement of data to QuickBooks.

In fact, ServiceDesk does not manage cost-of-goods-sold in any direct sense. Actual outflow of funds, to “pay” for these costs, is indeed managed directly by QuickBooks (or other separate system if you use something else), as you write checks to your vendors, typically as dictated by month-end statements, and usually paying for purchases as made in each preceding month. However, just because you are paying for a part via QuickBooks does not mean you are simultaneously, via such action, registering a cost-of-goods sold. In a typical scenario, it is indeed quite otherwise. Most often, such a payment converts cash in your bank account into an added inventory valuation in your inventory account. Both are equal-value assets so far as accounting is concerned, and the conversion has no effect at all on cost of goods sold.

So, though the money is being spent via QuickBooks, it does not mean QuickBooks directly possesses what it needs, for the sake of calculating cost-of-goods-sold. For that (and at least if it's to do so with precision), QuickBooks needs information from ServiceDesk. Any of several methods may be used.

1. The “*traditional accounting*” method takes the value of inventory at the beginning of a period, adds purchases during the period, and subtracts the inventory value at the end of the period (the resulting difference is your cost-of-goods sold). To accommodate this method, ServiceDesk will easily provide you with a snapshot of inventory valuation at any moment in time (use the shortcut sequence **F10→I→T**). If you prefer this method, it's up to you to figure how to implement it in QuickBooks. Regardless, be aware this method has a downside in that, typically, you are paying for items in a period different from when you actually received them. Especially if you are seeking to compile income figures on a monthly basis (and if you do not go to some pains to correct for this downside, such as, for example, registering each purchase/order in QuickBooks at the time it occurs, rather than waiting and registering it instead via writing a check in the following month), it can result in significant misstatement of income in any given month, though it will balance out over time.
2. Likely the best method is to use the report in ServiceDesk that's obtained via the shortcut path **Ctrl-F8→W**. A good title for this export would be “Report on Goods Sold.” You can run this report for the accounting period in question (whether it's for a particular month, a year, or whatever is the case), and get a tally of the exact items you actually used, and therefore truly bore expense upon (open the export in Excel and do a Sum function on column H to get this figure). You can then manually enter this figure into QB. This method somewhat reverses from the traditional method. So far as work in QB is concerned, you'll take your purchases, and add all to inventory. You'll then obtain your cost-of-goods-sold direct from SD, and subtract that from inventory.
3. The simplest method would be to simply treat each of your payments to parts vendors as direct cost-of-goods sold (and any credits back as reductions from cost-of-goods sold). It's not quite a correct method (at least so far as generally-accepted accounting principles are concerned), and it has the debility that any particular month's stated profit figure can be significantly distorted vis-à-vis another month's (particularly if you have a big month following a small one, or vice versa). However, those differences again wash out over time, and this method has a virtue in that it's extremely easy to implement. It's actually not bad, if you don't care too much about accurately discriminating one month's profit from the next.

4. For clarity of your own understanding, the default method that QB would naturally be inclined to push you toward (in particular, if you were allowing it to manage each of your sales transactions at the per-invoice level) would be to register cost-of-goods sold in conjunction with each and every particular involved sale (i.e., I am entering X sale, and my cost of the materials involved in it was Y). So that you know, there is no accommodation for this method in any reasonable ServiceDesk-QuickBooks partnership.

ii. Integration with Any Other Accounting System

If you are using any accounting system other than QuickBooks, general principles remain the same as described in the prior section. The one and major difference is there will be no export-import facilitated transfer of information for you, from ServiceDesk into your other system. So, you'll need to make these entries manually, as opposed to automatically. Since there are only a handful (or so) of figures, this is not laborious, and it only needs to be done at the conclusion of each accounting period. In principal, it's particularly easy, since all but one of the relevant summary figures are nicely flagged for you in the ServiceDesk *Sales Report* (please review the prior section for details on such flagging).

The one challenge you are likely to encounter, if you are not an accountant, will be in understanding precisely how this handful of figures should be entered into your accounting system. The main purpose in this section is to assist you in such understanding.

Particularly since we do not know what other system you will be using, we cannot give you precise details. At best, we can assist you in understanding such general principals as are involved. Thus, what follows here is designed to give you at least minimal acquaintance with some general accounting basics.

Every accounting system has a "Journal" or "General Ledger" (terminology varies). This is where entries are made that describe each event that changes the general accounting picture (e.g., earning money by completing a sale). Your basic purpose will be to make entries to this Journal (i.e., as exists in your own accounting system) to describe the summary information that needs transferred from ServiceDesk.

Whether it's apparent from the surface or not, all modern accounting systems ("modern" here means since Columbus) have as their foundation what is referred to as "double-entry bookkeeping." In this context, double-entry does not mean entering the same info twice. It refers, rather, to the fact that every accounting-significant event has an effect in at least two different areas of accounting interest. When you earn money by completing a sale, for example, you've added to your sales (one area of accounting interest), and simultaneously have increased either your cash (if the sale was paid for at the time) or have increased your accounts receivable. It's possible you did even a little of both. All accounting events have this dual-effect attribute (though, depending on circumstances, on different areas of accounting interest, and in different ways). The double-entry system was developed to assure every entry properly balances across both sides of this inevitable dual effect. It was a great invention.

Anyhow, we explain that as a foundation because, depending on the nature of your accounting system, you may need to make an entry in the general ledger where this double-entry mode is directly evident. Alternatively, your system may offer you an interface that somewhat hides or disguises the underlying double-entry foundation. Regardless, you'll be better enabled to understand the ultimate information that needs to go in, by reading onward.

Suppose that in a given accounting period you produce a Sales Report with the following figures (the ones underlined are the ones that need moved to your system of financial accounting).

TABULATION OF SALES TOTALS				
	Total of jobs completed in period	Total of jobs paid for in period	Total of jobs completed but billed	Total prevsly done and now paid
Merchandise Sold	0.00	0.00	0.00	0.00
Parts Sold	13,326.98	13,689.00	3,905.15	4,267.17
Total of goods sold	13,326.98	13,689.00	3,905.15	4,267.17
ServiceCalls Sold	16,798.00	17,519.00	4,023.00	4,744.00
Additional Labor Sold	14,729.34	14,953.34	5,054.00	5,278.00
Total of labor sold	31,527.34	32,472.34	9,077.00	10,022.00
Total Sales	44,854.30	46,161.32	12,982.15	14,289.17
Tax on sales	999.32	1,026.50	292.87	320.05
Gross Invoice Totals	45,853.63	47,187.83	13,275.02	14,609.22
Quantity of S. Calls:	355	368	87	100
Quantity of Invoices:	424	443	92	111
Average charges per S.Call:	126.35	125.44	149.22	142.89
Average charges per Invoice:	106.79	104.20	141.11	128.73
Ratio of total Labor to Sales:	70.3	70.3	69.9	70.1
Net change in Accounts Receivable via sales/payments:			(\$1,334.20)	
due to write-off as Bad Debt:			(\$75.00)	
---Overall change in Accounts Receivable this period:			(\$1,409.20)	
Funds written off as unredeemable, or missing and not recoverable:			\$45.00	

Again, though your system may have an interface that presents you with a different path for the purpose, the following are entries that will ultimately be going into its general ledger:

	Debits	Credits
<u>To record SD-reckoned sales for the month of November</u>		
Cash	47,187.83	
Accounts Receivable	-1,334.20	
General Sales		44,854.30
Sales Tax Payable.....		999.32
<u>To record SD-reckoned loss from bad debt</u>		
Bad debt expense.....	75.00	
Accounts Receivable		75.00
<u>To record SD-reckoned loss from bad funds</u>		
Missing or Unredeemable Funds	45.00	
Cash		45.00

If you're not an accountant, please try not to be troubled if some of the above seems mysterious. It may seem odd, for example, that an *increase* in cash is recorded as a "debit" rather than as a "credit." Regardless, it's

how it works (in beginning accounting classes they repeatedly drum into your head that the words “debit” and “credit” have no inherent meaning; they are simply *titles* given to the two different columns).

Now, for some explanation:

In the first rectangle/entry above, we are simply documenting the fact that \$47,187.83 has been fed into our *Cash* on hand (whether in the bank or otherwise it does not matter so far as *financial* accounting is concerned, please look at the example Sales Report to see where these figures were pulled from), and that our *Accounts Receivable* has decreased (based on the net of amounts going in and out from it) by the amount of \$1,134.20.¹⁹⁰ Plus, we are indicating the amount of \$44,854.30 should be credited to our *Sales* account, and \$999.32 to our *Sales Tax Payable* account. Thus, with these four entries (all of which in sum balance) we’ve input all the information that is directly related to our sales for the month.

Regardless, there are a few other matters the ServiceDesk Sales Report compiles, besides those so directly-related to sales. This is, in particular, because it also manages your accounts receivable and your funds collected. In either area, other events may occur. For example, some of your Accounts Receivable will prove to be uncollectible, and thus will need to be written off as bad debt. There’s provision within ServiceDesk for doing this (see page 177), but naturally, the fact of having done so needs to be input to your system of financial accounting. Also, it will occasionally happen that you’ve collected funds, yet they turn up missing, or otherwise cannot ultimately be collected upon (rejected bankcard numbers, bounced checks that can’t be remedied, etc.). In such cases, ServiceDesk again provides an internal method for documenting such losses (see page 158), and a summary of such activity is likewise included in its Sales Report. And again, the amounts in summary must be inputted to your system of financial accounting.

The second and third example entries, above, illustrate such needed entries. Again, you may compare the numbers shown to their source from within the example Sales Report. In the first such entry, basically, we are reporting having incurred \$75.00 as a *Bad Debt Expense*, along with a corresponding reduction in *Accounts Receivable*. In the second such entry (last in the above examples), we are reporting having incurred the loss of \$45.00 in *Missing or Unredeemable Funds*, and a corresponding reduction from our *Cash* account.

Again, regardless of the system you use, our point is these are kinds of entries that must ultimately go into its general journal. Your system may present you with an interface, by which to provide the information, which essentially hides that journal. But this is the fundamental information it needs, regardless.

¹⁹⁰For those uninitiated in the art of making such bookkeeping entries, here’s a basic primer. First of all, you should understand there are three basic kinds of “accounts” involved in any bookkeeping/accounting system: (1) asset accounts; (2) liability/equity accounts; and (3) income/expense accounts. As an example, cash and receivables are both asset accounts, as are inventory, office supplies, trucks and equipment, etc. Assets are, in other words, the things of value that exist within a business, and the corresponding “accounts” are simply the places, within an accounting system, where we keep track of such value. Liability/Equity accounts, on the other hand, represent loans the business has taken out, accounts payable, judgments owed, etc. (i.e., money owed to others), along with the balance left over (after taking the value of assets and subtracting liabilities), which is the equity of the owners. In other words, these are the “claims” against the business. For the third category of accounts, we should draw a distinction with the first two categories (i.e., assets and liabilities/equity) which represent values that, at any given instant, are static (though, of course, their values will change over time). Items in the third category of account, by contrast, describe the very process of change—or the fact of movement between assets and liabilities. Thus you may have various income accounts, such as general sales, interest earned, rent received, and so on. And you may have various expense accounts, such as wages paid, cost of parts used, interest expense, and so on.

With that as background, here’s the basic rule in regard to describing entries in connection with each of these kinds of accounts: (1) For asset accounts, an increase in the asset’s value (e.g., added inventory) is entered as a debit, a decrease as a credit. (2) For liability/equity accounts, an increase in value (e.g., greater debt) is entered as a credit, a decrease as a debit. (3) For income/expense accounts, an expense-event is a debit, an income-event (such as a sale) is a credit.

If the above does not seem logical, again, try not to worry. Trust the accountants, there is indeed a logic behind it all, but unless you’re an accountant, you needn’t understand it.

Other entries may be needed as well, depending on circumstances. For example, if you grant discounts to some of your clients for more rapid payment (see page 158), the ServiceDesk Sales Report will similarly highlight the amount thus granted (its designed to show and highlight everything as relevant, on the revenue side). Suppose, for example, you're looking at an inclusion which shows that, during the relevant period, you granted \$158.32 in discounts. A suitable resulting entry, to your system of financial accounting, would look something like this:

<u>To record SD-reckoned discounts granted on sales</u>		
Discounts Granted	158.32	
Accounts Receivable		158.32

You should notice the same principles apply, as with the other entries. We are indicating the simple fact we've incurred an expense of \$158.32 as a discount granted, and that it reduces our cash in the same amount.

Again, there is nothing complex. The general idea is to get information regarding these financial activities—as recorded and summarized by ServiceDesk (plus highlighted there for you)—into the particular system of financial accounting that you use for everything else. Though particular principles of financial accounting might seem a little weird, it's just a few figures, and the basic concepts of simple information flow are very simple.

There does, of course, remain the matter of accounting for cost-of-goods sold. Again, cost-of-goods-sold matters are *not* tabulated for you in the ServiceDesk Sales Report, and must be handled separately. In general, the options for such handling are precisely the same as if you were using QuickBooks (see prior section). The one difference, as relevant to the present discussion, is you may want to specifically know how an entry reflecting cost-of-goods-sold would look, when entered to the general ledger.

Suppose (using whichever basis is preferred), you have calculated that within the accounting period cost-of-goods sold summed to \$17,000. In such a case, your general ledger entry would need to look something like this

<u>To record use of inventory during the period</u>		
Cost of Goods Sold	17,000	
Parts Inventory		17,000

Again, it's rather simple, at least if you avoid allowing it to intimidate you.

If it nevertheless feels even slightly intimidating, try to remember that accountants require four years of college. Also remember that, regardless, the items of information that need transferred from ServiceDesk are very few and simple.

If the accounting end in itself seems too difficult regardless, perhaps you should hire an accountant, to at least help you get setup on the accounting end. He or she can read this section to learn what ServiceDesk has to input on its end of things, and teach you the process that's involved within your own accounting system. Once so taught, it should be a very easy (on a once-per-accounting-period basis) to make the entries needed.

I. Configuring for Remote Use and/or Secondary Offices

If you are at home or on vacation and want to do work in ServiceDesk (or within anything else that's installed-on or is otherwise available from within your office desktop), it's very easy. Just setup an account (these are either inexpensive or free) with LogMeIn.com, GoToMyPC.com or any similar service. Then, within that account, setup your desktop to act as a host. Once that's done, you can be at any computer anywhere in the world that has internet access, and within about 60 seconds (via a rather simple login process) find yourself with power to remote-puppeteer your office desktop — with power and flair that's virtually the same as if you were physically there.

It's a truly terrific capability, and it's hard for us to imagine why any owner or manager would not want to avail themselves of it.

Beyond that, you may have particular employees that wish to work from home. If any such employee is also equipped with an office computer, that computer can be setup (under your same LogMeIn or similar account) to also act as a host, credentials setup for it, those credentials provided to the employee, then the employee can access that computer from anywhere, and do remote work. If it's a person that doesn't otherwise have any physical presence at the office (and therefore there is no computer there assigned to him or her), it's possible you might have a spare box that's sitting around, and can be used for the purpose.

All the above is great, if your need for remote use does not extend beyond the kinds of circumstances described. However, many businesses wish to accommodate remote use on a larger scale. As a particular example, some businesses wish to setup with multiple offices (e.g., the main office, plus a satellite or two). Some may just want to have multiple remote users, and without each requiring a dedicated "slave" box at the office (i.e., which they can remote-puppeteer via something like LogMeIn). The remainder of this section will discuss the optimum strategy for achieving such a further purpose.

To start, let us state as a general principle: Just because multiple other persons are at one or more different locations (in particular, as compared to your main office), it does not follow they should be unable to work within the same program and data set — as if they were in the main office.

Regardless, the situation is at least more challenging than where everyone is working within the same premises. It requires a more extended solution, which is our remaining topic here.

We will begin by contrasting the more typical, single-premises situation.

Where there is physical proximity of all direct workstations (i.e., same building or building complex), they will in almost every instance be configured to "talk" within one and the same Local-Area-Network (aka LAN). This works great. Communication across a LAN is easily configurable, and is super-fast. Where all stations are in the same LAN, ServiceDesk setup is simple. You can go thick-client or thin (generally we recommend thin). To enhance your understanding, let's briefly discuss each.

In the thick-client scenario, you have ServiceDesk installed at each station. When you click on the icon to run it, your local processor (CPU) finds the program file on your own local drive, pulls its contents from there, loads the same into its operating memory, then proceeds to execute the program according to its prescriptions. Much of this execution involves reading from and writing to data files that are contained on the particular computer you've setup to act as a "data server." This might be an ordinary PC. It could be a Linux

box. It may or may not be additionally running ServiceDesk itself. For the purpose described here, it is simply a “box” that holds the desired data in a set of files on its own hard drive.

The thin-client scenario is not very much different. Here, you simply don’t have a local install. Instead, when you click on the icon to run ServiceDesk, your local processor reaches to grab an instance of the program file (again, to load into itself) that, instead of being stored on your local drive, is instead stored on the very same machine that’s acting as your data server. Just as in the thick-client scenario, that data server can be almost any kind of box (e.g., a plain PC, a Linux box or even a dedicated Windows server).

Hopefully, the above helps you form a good picture of a standard, within-LAN setup. We need that picture to distinguish from what’s practical when you have multiple offices (or even just one or two remote users) that are not in close physical proximity.

In this latter case, you don’t have the same advantages as when within a LAN. Quite obviously, your communication between distant offices cannot be through a LAN, because LANs only extend across more limited spaces. It must instead be via the *Wide-Area-Network* (aka WAN) — which (and in case you did not know) is simply another title for the *Internet*.

In comparison to a LAN connection, WAN connections suffer many comparative debilities:

1. Even super-fast broadband connections (i.e., via WAN) are many times slower than communication via LAN;
2. WAN connections are more vulnerable to security compromise (i.e., someone tapping into the data flow and capturing information that might harm you or others); and
3. WAN connections require more complex protocols to setup and maintain.

Items 2 and 3 can be overcome by setting up what’s called a Virtual-Private-Network (aka VPN). Essentially, it’s a way of configuring security and connection elements at both ends of a desired connection (i.e., from within Office1 and Office2) so that communications between them are as secure as in a LAN, and are individually as easy to setup and manage (this is aside from the very large complexity as involved in first setting up the VPN itself).

Unfortunately, a VPN does nothing for Item 1 above. In other words, even with a VPN, the speed of a WAN-dependent connection remains many times slower, as compared to a LAN connection.

For many applications, this speed issue is not a problem.

Suppose, for example, you are locally running MS-Word, and you’re in an office that’s remote from the applicable data store. When you open a document, the data that describes the document will be pulled from the data store, transported across the WAN, and placed into RAM on your local machine, where it will then be manipulated as you do your work. There will be no need for ongoing communication, with the data store, as you work on the document. The next need will arise only when you save the document, at which time your saved work will pass through the WAN, back to the data store, to there be saved. Unless it’s a rather large document, the time as involved in these two movements will not be onerous.

Unlike MS-Word, ServiceDesk operation involves very frequent, repeated (indeed almost constant) back and forth access between the application and its data store (including a multitude of files that over time become rather large). Because of this, if the processor that’s running ServiceDesk is in fact across-the-WAN

from its data store, you will see ServiceDesk crawl. It will be ugly. It will be virtually unusable. It is not practical.

So what's the solution?

To be very specific, how can we make it so the particular processor that's running ServiceDesk has no less than a LAN-intimate connection to its data-store, while still allowing some of its users to work in a remote office?

At first blush, it might seem it's impossible to achieve LAN-intimacy, between a remote office where instances of ServiceDesk are *evidently* running, and a data-store that's in fact at a distant location.

Did you notice our emphasis on the word "evidently?"

There's a trick, and it involves a special thing "Server" versions of Windows are configured to do, that ordinary Windows versions are not.

You're likely familiar with the fact ordinary Windows can be setup with multiple user logins, each with its own desktop that plays its own set of applications. Of course, an ordinary Windows setup will only "play" one such user desktop at a time.

Though "Server" versions of Windows have some other differences, the most significant difference is they are not subject to this one-desktop-at-a-time limitation. Indeed, a "Server" version of Windows can "play" many, many user desktops simultaneously.

Typically, of course, even a server version of Windows is *direct*-connected to but a single user interface (i.e., mouse, keyboard and monitor). You might call this desktop the "standard" or "native" one. The others are generally called "Virtual Desktops."

So, where are the user interfaces for those multiple "Virtual Desktops?"

Essentially, any other computer interface can plug into any of a server's virtual desktops by using a technology called "Remote Desktop" (aka RDP). Remote Desktop is automatically a part of every modern Windows system. If you're on any such box and have the credentials to login to any virtual desktop as being played by any Windows server anywhere, all you need is to run your own computer's Remote Desktop utility, put in the credentials, and inside of seconds you'll be running that virtual desktop that's "playing" on a server elsewhere — in spite of the fact that server may be located many thousands of miles distant.

And here's what's particularly great.

So long as that server is LAN-intimate with any data-store that an application within your own virtual desktop is running (e.g., ServiceDesk), the application will perform superbly. Typically in these setups, indeed, the data-store is on the server itself — i.e., the same machine, which makes it even better than LAN-intimate.

So, that's the trick. Essentially, ServiceDesk runs on the server (in multiple instances upon multiple virtual desktops), with workers in your remote office remote-pupeteering it (indeed, not just it, but their entire desktop environments as well). Indeed, they can do the same from their homes, or any location desired. Plus, there is no need for an underlying (and complication-causing) VPN. The RDP technology takes care of security.

How do you go about setting up this kind of configuration?

Surprisingly, it's not difficult. We have a number of users who've done it themselves. We have others who've hired professionals. There are a plethora of companies that will lease you such server capacity, online. Even here at Rossware, we offer to provide such online server services for you. Our service is called Rossware Server Solutions (RSS).

All in all, our purpose in this document has been to help you understand the general concepts. We hope it has succeeded.

J. Details re Compatible Networking Strategies Within Windows, How to do Mapping, Etc.

As a general matter, ServiceDesk should work with *any* network system that allows one computer to read and write data from/to another's hard drive, and through a Windows-based interface. There are but three complicating factors, each addressed in this section.

i. Mapping Network Drives

If you've newly setup a network, you may go to Windows Explorer, look under 'Network Neighborhood,' and happily see that, sure enough, you can read what's on the drives of every other computer in the network. Very neat. You might *think* this is all that should be needed in order for ServiceDesk (or any other Windows-based application) to access those other drives. In fact, you'd be wrong. While Windows Explorer may be able to "see" and show you the other drives without further work, those drives will remain invisible to most *applications* absent a little thing called "Mapping."

What is mapping? Basically (and in a nutshell), it's the process of designating from one computer that all or part of its drive is available for "sharing" with others, giving that shared offering a "Share Name," then, from another computer, linking to that shared offering with a Drive Letter Designation.

How do you do this?

As a short answer, we might say this is not a ServiceDesk topic; it's a Windows topic, so don't bother us; look in your Windows documentation instead. However, we're a little nicer than that, and will here give you a brief "how to" (suitable at least for Windows 95/98; later Windows versions may slightly vary) as follows.

As a beginning matter, remember there are two major steps: first, making the wanted drive available for sharing from the computer you'll be using as FileServer (we'll assume here that it's that computer's Drive C:); and second, setting up any and every secondary computer to read from that FileServer's Drive C:.

To accomplish the first step (making the FileServer's Drive **C:** available to other computers), click on the *Network* icon in that computer's Windows *Control Panel* (accessed by clicking on *Start*, then *Settings*). Under the *Configuration* tab, click on *File and Print Sharing*, and from there assure you're configured to allow others full access to your files. Next, using either *Windows Explorer* or the *My Computer* utility of that computer, select Drive C:, then click on Properties in the *File* section of the Menu. Now, under the *Sharing*

tab, provide a Share Name (something simple like "DRIVEC" works fine). Finally, select *Full* access, then click on OK. Assuming your network is otherwise functional, your FileServer should now be ready to share the entire contents of its Drive C: with other computers.

For the second step, your other computers need not just to see the FileServer drive from across the network (a basic assumption in network functionality) but, more critically, will need a Drive Letter Designation with which to reference it. This is the same kind of designation as in '**A:**' for your floppy drive, for example, or as in '**D:**' for your CD drive—only here you'll be using a different drive letter than any already used. To assign this Drive Letter Designation at any workstation, first run *Windows Explorer* from that station. In the left-hand column, locate the Entry labeled "Network Neighborhood." Click on it to expand and look within its contents. Once you have there located the other drive that you wish to map locally (presumably the one being used as FileServer), right-click on it. In the little drop-down menu which then appears, select "Map Network Drive."¹⁹¹ Windows will then produce a little box in which you specify the particular drive letter you wish to assign. Make your selection (we use **S:** as a reminder for Server, but you can select any letter not already used by an existing drive). Also check to true the box labeled "Reconnect at Log on," then click on OK.

Under *Path* type in the actual path for the drive—as revealed when you showed it expanded under Network Neighborhood over in Explorer's left column. When typing in this path, use two back slashes before the computer's name, then one before the share name, with no spaces between (e.g., as setup in our office, we use \\SERVER\DRIVEC).

With this accomplished, you should now see the newly-mapped drive listed in Explorer's left column, in a direct line below (i.e., in the same hierarchy of listing) as *My Computer*. But here, as opposed to the computer's mere name listing under *Network Neighborhood*, the description will include that all-important Drive Letter Designation. It's in parenthesis following the description, and it's proof you've accomplished your task.

As final proof, you may run *ServiceDesk*, bring up the Settings form (Ctrl-F1), and click the down-arrow in the box labeled '*Drive Designation for ServiceDesk Network FileServer.*' The mapped drive should appear in the list. Indeed, assuming that it was properly mapped, it *must* appear within this list.

Bear in mind this second part must be done on *each and every* computer that you're using a workstation (i.e., each that will be reading the common *ServiceDesk* data files from elsewhere). Simply because you're mapped the shared drive to one station, it does not mean that the next station is similarly ready. This work must be done on it too.

ii. Sharing Only the SD Folder, Rather than an Entire Drive

For security reasons, some users have not wanted to make the entire contents of one computer's Drive C: available to other stations in the network. They've wanted, in other words, to limit the shared access to just the particular folder that contains the common *ServiceDesk* data files, while preventing shared access to anything else. With the release of Version 3.8 (74), this is now possible.

¹⁹¹ In the alternative to this semi-automatic method of mapping, you could do it somewhat more manually by first clicking on "Tools" in the Explorer menu bar, then on "Map Network Drive." In this case you'll have to type-in the path for that drive yourself (more difficult, so not particularly recommended). When typing in this path, use two back slashes before the computer's name, then one before the share name, with no spaces between (e.g., as setup in our office, we use \\SERVER\DRIVEC).

To setup for this, go through the mapping process just as already described, except instead of designating all of Drive C: for sharing, designate only the \SD folder (which, obviously, will need to have been setup already on the applicable machine).

ServiceDesk copes with this different setup as follows. Normally, when a *full* drive is shared, ServiceDesk looks for a root folder within that drive called \sd, and then for the various sub-folders (under \sd) that should be created when ServiceDesk is installed and first run. The difference is, when you share the \sd folder itself, *it* does not show when ServiceDesk does its standard look for appropriate folder structure. Instead, each of the sub-folders show as though they exist on a root level. The solution, if ServiceDesk does not “see” an \sd folder on the root level of whatever drive is designated as FileServer, it looks for what’s normally the sub-folders, but on an *apparent* root level. If it finds these, it assumes that it’s dealing with a folder-only shared situation, and reacts accordingly.

iii. An Option: Using the Server for Operating as Well as for Data Files

Until release of the same revision as mentioned above (3.8 (74)), it was always our expectation in ServiceDesk that each station would keep and use its own, local copy of various operating files, including the program file itself (servdesk.exe), the file that instructs the system in the wanted format for printing invoices (*****.prg), and each of the five CstmrDbase indexes. The reason for this strategy is to maximize performance. If a station has to go to the server for each of these files (and if multiple stations are going to the server simultaneously), there may be little delays and pauses as the server works to keep up with demand.

In general, the standard setup works very, very well. However, the old-fashioned setup of having a comparatively powerful central computer with relative “dumb” terminals connected is making something of a comeback, and at least one client has been found flirting with the idea of adopting such a system. If this is the system, it becomes almost *necessary* to keep the operating files (as opposed to merely the data files) at one, central location.

Plus, there’s a practical advantage, for there’s some amount of overhead that goes into keeping up-to-date *operating* files on each of various machines. Every time you update the main program file, for example, updated copies need to be transferred to each station. When you create or revise that *****.prg file, each station needs its own copy. When the CstmrDbase indexes are updated, each station needs its own set of copies. And so on.

For reasons of these impetii, we’ve now made the system workable with a single set of Server-based operating files. The trick to make it work is very simple. From any station, simply run the particular *servdesk.exe* file that’s found on the server (rather than one that may or may not have been installed locally). When started, ServiceDesk looks to see what particular drive it’s being run from. It looks to *that* drive for other operating files—even as it continues to look locally for strictly *local* information, in particular the file that keeps track of each station’s *local settings*.

We do not yet know of anyone using this option. If you choose to do so, please let us know how it works.

K. Focused Dispatching, A Solution for the Large Enterprise

The majority of ServiceDesk clients have ranged in size anywhere from one to ten trucks. If you begin to get larger than this, you'll probably have more than one dispatcher controlling all those many techs. As these dispatchers contend with their respective tasks, they'll likely notice that the DispatchMap becomes so cluttered (having such a large quantity of jobs displayed) that its usefulness is compromised. To deal with this situation, we've developed what we call *Focused Dispatching*.

The general notion is very simple. You'll likely want to have one dispatcher concentrate on one portion of your territory and set of techs, while another concentrates on a different area and set of techs.

The method we've designed for ServiceDesk to accommodate this is, at this point in time, very basic. Indeed, the one accommodation it directly provides is that it allows you to define a *focus-link* between a particular office person (probably a dispatcher) and a set of techs. With this link established, when that office person goes to her DispatchMap, all the non-focus group jobs will be, essentially, grayed out. This makes it easy for that dispatcher to focus only on the set of jobs and techs assigned to her.

To define this focus-link is very simple. It's all done within the Settings form. The idea is to place a "third word" after each applicable person's name—that word being a number which indicates the *focus-group-number* to which that person is assigned.

More specifically, you can use any number you want for each focus group, but if, for example, you're

List of Technicians (again first and last, no commas, and note that you must enter after typing to insert in list)

Glade Ross 1
Arthur Manoogian 2
Richard Weimer 1
Dave Sommer 1
Curt Butschke 2
Leslie Nielson 2
Bob McNamara 2
Roger Mahoney 1

planning to divide your staff into two such groups, it would probably make most sense to call them groups '1' and '2' (though, so far as ServiceDesk is concerned, it does not matter so long as you use any whole number).

When ready to place each of your techs into a focus group, you should go to the Settings form, and within the ListOfTechNames there (purple section, net-wide settings) append each tech's name with the focus-group-number that you wish to assign him to. Remember, this number must be arranged as the *third* word in the name line (his first and last

name constituting the first and second words).

With that task having been accomplished, the next is to designate each of your dispatchers according to the particular group they will focus on. For this purpose, you'd probably assume you could just go to each of their names within the ListOfStationNames, and similarly designate there, much as we just did within the ListOfTechNames. While you *could* do that, it would not have any operational effect within ServiceDesk. Instead, to designate that a particular operator is assigned to a particular focus group, ServiceDesk looks at a *local* value: specifically, the local StationName as designated in the upper right-hand corner of the green section. There (note that, since this is a *local* setting, it must be done at the particular machine to which the dispatcher is assigned), you can append a number, after the dispatching person's name, matching the focus group that you wish to assign her to.

As you can see, the strategy for setting this up is indeed simple—and the result is simple too. The only thing, that occurs differently on an operational level is that, on a station where the operator is assigned to a focus group (as in the above illustration), jobs assigned to techs from any other focus groups will, as stated, be grayed out when viewed from within the DispatchMap.

In addition to the above, a user might choose to do other things. For example, you could setup each dispatcher's station so that the "home" position on the DispatchMap is centered toward the geographic area that is their focus (for instruction on how to do this, if wanted, consult RossWare).

Also, if the need should arise, we might in the future do other things to the programming itself. We might write code, for example, that would make a job by itself default to one group or another, without requiring that it first be assigned to a particular tech within that group (note that at present that's the only basis of designation for the job). We'll welcome feedback on whether you have any such needs.

L. Automating Customer Communication

The average service office logs a lot of time in telephone conversation with customers. It's a significant expense, and not very efficient — from either the customer's or company's perspective.

A few years ago we got to thinking, what if a lot of this could be automated? What if, instead of having to *talk* to someone in your office to handle their needs, your customers could manage them independently, via interfaces on your website? The concept was so great, we had to pursue it. Our *CyberOffice* set of functions is the result.

In a nutshell, CyberOffice is a set of "plug-ins" for your website where customers can attend to much of their business with you, including such matters as:

- Initially requesting service and booking their job;
- Re-booking after parts arrive;
- Confirming they are ready for and expecting the next day's appointment (or re-booking, if required);
- Tracking their tech's progress on the day of the appointment; and
- Checking on the status of in-progress work.

All of these "plug-ins" automatically integrate with ServiceDesk within the office, so — whatever your customer does online — the result shows in ServiceDesk. Plus, whatever you do within ServiceDesk that is relevant to your customer needing to see online, ditto, it's available there for your customer.

The system also involves *emails*, auto-generated via mechanisms within the office to link the customer to various of the website interfaces, as appropriate. After parts arrive, for example, and email will generate (only with operator consent), informing the customer they are in, with a link that takes them to the "plug-in" interface on your website where they are permitted to re-schedule. When you've worked out your appointments for the next day, email reminders can then be automatically sent (again, with user consent), and

it's these reminders that ask each customer to "link-to" another "plug-in" interface on your website, where they are able to confirm (or re-schedule, if need be).

The functions go on and on, and the whole system is available at the minimum cost of just \$10/month (total cost may be greater, depending on usage rate). We don't think it makes any sense not to use it. Just call when ready, and we'll send you a setup email.

M. Using our *DispatchLink* Utilities

The modern world is a great place, and we're glad to live in it.

Up until pretty much the end of the last century, if you were a consumer and called a manufacturer for service, the rep would typically ask for your zipcode, type it into a computer, and get the contact info for a handful of independent companies who could provide the service within your area. She'd read off some names and phone numbers, you'd write them down, and it was then up to you to call one or more, seeking the needed service.

The first big change occurred in result of cooperation between ServiceBench and Whirlpool. Together, those companies realized the consumer was not left feeling all that well served when, after first calling the manufacturer, they then have to make even more calls before finally receiving service. They got to thinking: "What if a manufacturer could schedule its consumers right off, during that first call, even if for an independent servicer?" By and by, they put together a system to make it happen. Basically, ServiceBench setup a web-interface for each of Whirlpool's servicers. On this interface, the servicers were supposed to keep ServiceBench informed of which zipcodes they still had vacancies for on which days. This information passes to Whirlpool's computer system, so its operators can actually see (when talking to a consumer) which servicers are available for scheduling — then go ahead and book the consumer for that servicer. The remaining element is, when that "booking" is created, it shows for the servicer on their ServiceBench web-interface.

The system was great from Whirlpool's perspective, but from that of the servicer, was a little less so. How in the heck does a servicer find the resources, every time they fill-up (or free-up) space as available for a given zipcode, to update their ServiceBench settings to so indicate? And this is only to keep them informed. It's quite aside from the fact it's something of a burden to regularly check your ServiceBench web-interface to see if new dispatches have been created for you — and, if so, to read the information, then *manually* enter it into your own local management system.

This was the state of affairs when we (Rossware) came along, and we figured it should be fixed. Sure, Whirlpool and ServiceBench had achieved automation between themselves. We figured automation was likewise needed with servicer.

We pitched the concept to ServiceBench, they bought it, and in result built their "Real Time Integration System" (RTIS), to facilitate precisely the kind of automation we'd requested. From our end, we built a little utility that (though functions have since expanded), initially had two tasks: (1) automatically keep ServiceBench informed of your real-time availability status, and (2) automatically grab dispatches, when created, and plug them perfectly into ServiceDesk.

Initially called the “*WebBasedDispatchEnabler*” (WBDE), the system was a hit from the very start – so much so that other entities asked us to work with them to achieve similar results. Soon, we had functional integration working with ServicePower, and, similarly, with LG too. Since each entity achieves its integration differently, it required separate utilities, from our end, to achieve the integration with each. We called the new utility for ServicePower the *SP-DispatchLink* (SPDL), and the one for LG the *LG-DispatchLink* (LGDL). To maintain consistency with this naming convention, we changed the name of our ServiceBench utility form *WebBasedDispatchEnabler* to *SB-DispatchLink* (SBDL).

So, that’s our present family of DispatchLink utilities (we plan to add a *Samsung-DispatchLink* soon). Each is available for the simple usage fee of \$17/month. Just let us know when and if you want one, and we’ll email you a setup.

N. Splitting Commissions on a Job

At this point in time, ServiceDesk has no elegant method of splitting commissions. However, there are means by which you can “make do.” Since commissions reports are generated on the basis of information in the SalesJournal, and since the SalesJournal only has a single field to indicate a single responsible technician as connected to each entry there, it follows that to credit different techs on the same job, you have to split up the entries on that job—i.e., make more than one entry on the one job, each for the portion of the total sale on that job that should be credited to each tech.

Suppose, for example, you had a sale for \$250 (all labor). \$150 of that belongs to TechA; \$100 to TechB. If it was a COD job, you can simply make two entries from the F9 SalesEnter form, one for each such amount, each under the appropriate tech’s initials. The system will whine and complain just a bit, warning the entered amounts don’t equal the total collected, and as you make the second entry will warn that you already made an entry on that job, and so on. In this instance, just ignore the warnings, and proceed ahead.

The real complication arises where it’s a billed job. The reason is that, ultimately, you want to have a single Account Receivable record (“A/R”), but as connected with two (or possibly more) separate SalesJournal entries. Still, it’s not difficult.

In this case you have to handle it differently, depending on whether you pay your techs when the work is completed, or only when you’re ultimately paid (i.e., whether they’re paid on Paycode 2s or Paycode 3s). Here we’ll describe how to do it if you pay them on Paycode 2s. If you pay on Paycode 3s instead (and also need to split commissions), let us know and we’ll separately provide you with instructions on that.

While there are other ways you could do it, the easiest will be as follows (again we’ll assume a total sale of \$250 labor with a \$150/\$100 split between TechA and TechB):

In the F9 SalesEnter form make your Paycode 2 entry for \$150 under TechA. In result of this entry, the system will, of course, create an A/R record showing the \$150 due. Immediately after making and recording this sales entry, hit F3 to bring up the A/R form. The record just created (being the most recent) will be the one to automatically display (so you don’t have to look for it). Change the labor amount in the form from the formerly split amount to the total (i.e., to \$250). Also change the initials in the ‘Tech’ box from TechA’s to TechB’s. Now exit the form. As you do so, ServiceDesk will save your changes and simultaneously note that, since there’s a been a change in amount, there ought to be a new entry in the SalesJournal to reflect that

change. It will offer to make that entry for you. Simply consent to having it do so. In consequence, it will make a second entry under that same invoice number (and under TechB's initials) for the added \$100. You can now hit SHIFT-F3 and should see those two entries as the most recent, exactly as they need to be, and both connected to the one A/R record properly residing in the F3 form.

Again, it's not perfectly elegant, but it's not hard either, and works pretty well.

O. Setting up each Tech as his own Parts Center

It's well known that, typically to avoid payroll taxes and workmen's comp expense, some servicers setup their techs as "independent contractors." Often, such techs are "independent" in name only. However, there are some servicers who truly make their techs into genuine, independent business centers—to the extent the techs purchase their own parts and manage their own inventory. This *had* been a problem for ServiceDesk to manage, since our design assumption had been that all parts were purchased and owned by the company itself. As of April 2005, however, we created a solution.

In principle, the solution is very simple. Rather than having the various parts management files kept in the \sd\netdata folder on the server—as is the normal practice, and which results in there being only a single parts system that's common to all users—we provide an option whereby each ServiceDesk operator can instead maintain and manage their own unique set of files.

To invoke this option, you simply need to create a new sub-folder, within the \sd\netdata folder, on the server. Give this folder the name of the tech to whom you wish it to apply, with exactly the same spelling as his listing in the Settings form's 'List of Station Names' (i.e., you'd be making a folder such as **c:\sd\netdata\John Smith**).¹⁹² When ServiceDesk sees this folder (and matches it with the name of the person who's logged in as operator), it will deduce that all parts-related files, as associated with that person's specific work, should be located in that folder. Thus (and as desired), that person will end up working with his own unique set of such files.

A remaining problem relates to warranty claims. Since each tech is purchasing parts from various distributors under his own account, it follows that for each warranty claim, it's that tech's account number that needs to be included. To accommodate this need (i.e., to make that inclusion automatic, so an operator doesn't have to manually insert the appropriate account number with each claim), you need to create a file for each such tech that provides such account numbers for him.

It's likely easiest to make this file in Excel, though you could do it in any text editor. The general idea is you make one line of text for each manufacturer for whom you do warranty service. Each line of text has two fields (or columns if you're working in Excel). In the first field you place the HighVolumeClient abbreviation for the manufacturer of interest, as setup in the QuickEntry template for them. In the second column, you place the account number of the tech for the distributor from whom parts for that manufacturer are normally purchased. If working in a text editor, the two items of text should be separated, simply, by a single Tab. If working in Excel, you'll need to save as a tab-delimited text document. In either case, the file should be saved in that same special folder we mentioned above, under the name **TechsOwnAccountsFile.txt**.

¹⁹² When companies operate in this mode, techs end up getting listed both there and in the Tech Roster, since, as true business centers, they end up working both from the normal office end, and as techs.

When ServiceDesk sees this file, and it's inserting data to a FinishedForm as applicable to the tech for whom the file was created, it will look in the file to find the applicable distributor account number as placed therein, rather than the one that's in the QuickEntry template as applicable to the company as a whole. Assuming it finds such, this is what it will use instead of the number as provided in the QuickEntry template itself. If it does not find the special number as applicable to the tech and manufacturer, it will revert to the number as found in the QuickEntry template.

A final matter concerns the question of which files are looked in, for the sake of finding parts used from stock or special ordered, for insertion to the FinishedForm. Our largest concern is to assure that any part as actually used on a job get included in a claim. Since there is some chance that parts as used might be reflected in a tech's personal parts files, or might be reflected in the parts files for the company itself, we've structured the system to look in both places.

5

Your Own Custom ZIPS, City Abbreviations, and other Special Info

After compiling a list of street names and their accompanying zip codes for your territory (as compiled from Census Bureau data), we further compiled a list of each particular zip code with its accompanying city name, as specified by U.S. Post Office data. As mentioned elsewhere, the city name connections may, in some cases, be spurious (or perhaps missing completely). We list them here to allow you to review the zip-to-city name connections, noting those that are in error (or missing), so that you may enter appropriate corrections into your StreetList, using the easy procedure described at page 291.

ZIP CONNECTED CITY CORRECTED CITY, IF ANY

LIST N/A IN PDF VERSION

For those city names presently included in the above list, we have created ServiceDesk abbreviations (i.e., these are the abbreviations you'll presently see in your StreetList) to reference them, as follows:

LIST N/A IN PDF VERSION

Obviously, you may want to reference this list if, while using ServiceDesk, it is not obvious while looking at any particular street name which city the accompanying abbreviation refers to (you may even want to make a photocopy of this page and keep it handy at your desk).

Also, if you find that you've added new city names while correcting the zip-to-city name connections (list preceding the above), it would be prudent to add those names (by writing them in), and the abbreviations you've created, to the above.

Also in regard to customization, please bear in mind that your covered area needn't be static. For a reasonable fee, we can change your map to add new areas, or take away existing ones. Just let us know if you have such changing needs.

You may find it interesting, incidentally, to know that the process of producing your customized CityList, on-screen Map, and StreetList is quite involved (the originating data we purchase from the Census Bureau is not even remotely in the form needed for a finished product). On average, it requires approximately 10 hours of human programmer time along with nearly 90 minutes cumulative processing time by a high speed office computer, all in a sequence of events that encompasses at least 35 separate operations. Obviously, such a process would not have been economically feasible with earlier generations of computer hardware from just a few years ago. We're glad they are now.

In your case, this formidable process yielded a StreetList consisting of N/A entries (i.e., one entry for any occurrence of a street name under a particular zip), along with a BlockList including N/A listings (i.e., one entry for each approximately five-block segment of a street within a given zip). If you want to get some idea of the quantity of data this involves, you can view your StreetList directly from almost any word processing program. Just load the document entitled N/A.STR from your 'sd' folder. You could even edit it from a word processor if wanted, but it's critical to keep each line exactly 48 characters long, and to save the result in 'text-only' format.

In regard to your on-screen DispatchMap, we've found that some folks thought we just took some existing, commercial map, and excerpted out the portion that was needed for their territory. This is not the case! Every single element in your DispatchMap was manually drawn specifically for you. It was created, as they say, "from scratch." We do use commercial sources, of course, to determine where the various geographic features fit within your territory, but having made that determination, every element (indeed, every point in every road, boundary or shoreline that's drawn) is manually input so as to create the final file that is your DispatchMap. This is tedious and exhausting work. We mention it so you'll have some further appreciation for what was involved in creating your unique package.

If you like to tinker, by the way, it's also possible to view (and edit, if you wish) the particular file that describes (so that ServiceDesk can display) all the features that make up your DispatchMap. In this case (and again, using almost any word-processing program), just load the file called N/A.MAP. Again, you should find it in your 'sd' folder. In terms of the layout there, line length is not critical, and the functions of each line are, for the most part, obvious. If you want to try changing or adding something (some people, for example, add in location references to show where each tech resides), go ahead. If you end up messing things up, just copy back over the original from your installation CD.

As mentioned elsewhere (see page 287), it is our intent, in creating your custom StreetList, to make one that mirrors the index section in your own map book as closely as possible—particularly in regard to the street names included, and in regard to the grid references given for each (of course, we include city abbreviation and zip codes for each street, which your own paper index may lack). In spite of intense efforts to make the reproduction perfect, it is not. Our system (given its independent source) will have produced a few street names not in your paper index, and certainly, the latter will contain entries not in the list we've produced. There will also be variations, here and there, in the grid references listed for particular streets (especially in respect to streets that lie near the division between two grid sections, or which span across multiple sections).

Still, and in spite of such differences, the list we've produced should be of extremely high quality.

If this degree of quality is not sufficient to your needs; if, in other words, you want a perfect duplicate of the data in your map book's index, you certainly may attempt to obtain the underlying data from your map book publisher. If you provide such data to us on a mere diskette (make sure it's in ASCII format, containing street name, city, zip and grid reference for each entry), we'll be happy to convert it to the ServiceDesk-needed format. Ask us for the current fee for this service.

6

INDEX

- Abbreviations
 - for HighVolume-type client, 60
 - for your city names, a listing, 323
- Accounting, Financial
 - how to input information generated by ServiceDesk, 302
- Accounts Receivable
 - Applications Journal, 179, 192
 - archiving/purging, 179
 - bad debts, writing off, 179
 - Debt Advisory, system checks on inserting Customer Info set to Callsheet, 67
 - dunning letters, statements, billing reminders, 174
 - home-warranty notices, 177
 - major discussion, 173
 - payments received on, 178
 - standard dunning letters, 175
- Acrobat Reader, 219
- [Across the counter sales](#)
 - [using the Finished Forms system to accomplish](#), 189
- Adapting to ServiceDesk, who should bend, 46
- Alpha-numeric dispatching, 90
- Alpha-numeric paging. *See* Dispatch and Scheduling
- Answering service
 - illustration of properly formatted setup, 335
 - retrieving messages, 41, 63
 - setting-up for, 278
- Applications Journal. *See* Accounts Receivable, *See* Accounts Receivable
- Appointment Density Graph, 85
- Appointment notations
 - AM versus PM contextually assumed, explicit indication optional, 294
 - day-of-month only is typically required, month is generally inferred, 295
 - technical discussion regarding, 293
- Archiving
 - Callsheets, 34, 79
 - JobRecords, 36, 108
 - or actually purging, old A/R records, 179
 - schedule of nighttime events, 221
 - ScheduleList, 101
 - somewhat different method used in FundsJournal, 163
- Arrival-On-Time Sentry, 89, 114
- AutoArchive feature
 - first mentioned, 39
 - general discussion, 220
 - option to invoke manually, 222
- Auto-CstmrDbase Search Utility. *See* CustomerDbase system, use from a Callsheet
- AutoDialing, 65
- Automated Emailing of error messages, 45
- Automated updating of your ServiceDesk program, 51
- Backing up your data, 215
- Bad debts, writing off, 162, 179
- Bad Sales Tax Situation
 - major discussion on how to handle, 299
 - specification within Settings form, 227
- Bankcard transactions. *See* Funds Control System
- Blank Paper
 - the option to print your invoice onto, 275
- BlockList. *See* StreetList, one of your custom files
- Bugs. *See* Error messages, system crashes
- Call management. *See also* Callsheets
 - main chapter discussion, 53
 - overview, 33
- Callbacks, reporting on, 204
- Callsheet Archive
 - used as a kind of telephone book, 209
- Callsheet Archive, storehouse for completed Callsheets described, 80
- Callsheets
 - archiving. *See* separate heading under this term
 - business versus a person's name, 78
 - desk assigned, changing, 55
 - drop-down lists in item(s) type and make boxes, 201
 - first examination, 24
 - first experimentation, 26
 - hibernating, 30, 68
 - interpretation of text within its boxes, *see* text and note, 293
 - navigation within and among, 53
 - numbering, 76
 - saves, when performed, etc., 77
 - special editing tools, 75
 - status, changing, 54
 - street entry. *See* separate heading under this term
 - updating to changes at other stations, 77
 - upper case characters preferred, 78
- CapsLock on your keyboard
 - ServiceDesk sets automatically, 78
- Cash. *See* Funds Control System
- Census Bureau
 - raw StreetList data obtained from, 288
- Centralized operating files
 - the option therefor, 315

Chat room
 use for support, 50

Cheat sheet. *See* Contextual Command Summeries

Checks. *See* Funds Control System

CityList, one of your custom files
 correcting inaccurate city-to-zip connections, 291
 your own shown, with connected abbreviations, 323

[Claims, formal](#)
[creating and filing](#), 182

Command Summary
 item-by-item listing, 231
 part of MainMenu, 219

Commissions
 reports, 202
 splitting, 319, 320

CommPort, specifying, 225

Completion analysis, 203

Contextual Command Summeries (aka "Cheat Sheets")
 for Callsheets, 76
 from the DispatchMap, 94
 from the ScheduleList form, 101

[Counter sales](#)
[using the Finished Form system to accomplish](#), 189

Create Job/Sale form. *See* Job creation

Customer mailing lists
 making, 207

Customer records, methods for finding, 209

CustomerDbase system
 as used via the TechInterface form, 209
 displays jobs via abbreviated JobsArchived form, 285
 first described, 38
 if expected item does not appear, troubleshooting guide, 286
 indexes, each station has own copy, 285
 indexes, making new, 121, 284
 indexes, updated with each ArchiveJobs event, 284
 indexes, using the auto-archive feature to keep current, 221
 limitations, 209
 major discussion, structure, technical background, etc., 283
 moving from job-displayed in this context to operative-form context, 286
 not all job records get indexed, 285
 searches from JobsCurrent form, 106
 use from a Callsheet, 66, 209

Customizing
 the list of GracePeriods under the WipAlerts system (see footnote), 119
 the TimeFrames for scheduling list (see footnote), 96

Cut-sheet feeders, 268

Debt Advisory
 system checks on inserting Customer Info set to Callsheet, 67

Definite versus tentative. *See* Tentative versus definite

Departmentalizing sales, 172

Deposits, preparing, 159

Dial-Up Networking, 310

Discounts. *See* Funds Control system

Dispatch and Scheduling. *See also* Job scheduling
 alpha-numeric paging, 90
 archiving past days, 101
 arranging technicians' routes, 41, 86
 checking-off that a job's been dispatched, tech has arrived, tech has finished, etc., 87
 check-off symbols and what they mean, 88
 EmailBasedDispatchEnabler, 103
 focused dispatch setup, 316, 317
 major chapter discussion, 81
 on-demand dispatch, 91
 overview, 36
 remote dispatch, 89
 re-sequencing of jobs, 87
 ServiceBench, working in conjunction with, 102
 WebBasedDispatchEnabler, 103
 whole-day scheduling, 99
 ZoneScheduler system, 102

Dispatch Enabler Utilities
 Web and Email Based, 103

DispatchMap
 description re underlying machinery, 292
 design, major elements, description, 82
 first perusal, 24
 function distinguished from ScheduleList, 37
 graphic representation of jobs may be positionally imperfect, reasons why, 292
 inset feature described, 82
 jobs positioned on basis of provided grid reference, 289
 jobs shown in two manners, list and graphic, 83
 jumping to SchdList entry, 101
 name of your custom file, may amend if wanted, 324
 overview mode, 82
 panning, 82
 printing a schedule list, 94
 printing the schedule to file, 94
 QuickLinks to other forms. *See* QuickLinks as root heading
 using the End and Home keys as panning shortcuts, 83
 using the ShowJob method, 84
 using the ShowText method, 85
 viewing past-day's schedules, possible imperfections, 293
 which day shown, 84

Display
 if it's not right, correcting, 251

Documents connected with a job
 the general scheme in ServiceDesk, 47

Drop-down lists in Item(s) Type and Item(s) Make boxes of Callsheet, 201

Dunning letters
 and statements, both as types of billing reminders, 174

[Electronic claims filing. *See* Finished-Form system](#)

Electronic Funds Transfer
 how to apply. *See* footnote at

Email
 first described, 39
 general dicussion, 197

- old, 198
- TechWindow mode, 115
- Email Dispatches
 - inserting text to Callsheet, 62
- EmailBasedDispatchEnabler, 103
- Error Messages
 - automated emailing thereof, 45
- Error messages, system crashes, 45
- ExtraNotes. *See* MoreInfo notes
- Fictional data created while practicing,
 - getting rid of, 252
- File, printing to
 - from schedule, 94
- FileNamePrefix
 - evident in name of custom files, 324
- Files
 - itemized listing and description of those used by ServiceDesk, 245
 - structure of ServiceDesk directories (or folders), 245
- FileServer. *See also* Networking
 - defined, 224
- Final setup, checking off items, 251
- Financial Accounting
 - how to interface with ServiceDesk, 302
- [Finished-Form system](#)
 - [custom form](#), 182
 - [Electronic transmission of claims](#), 182
 - how ServiceDesk collects information for insertion to forms, a technical description, 296
 - [major discussion](#), 181
 - [NARDA forms](#), 181
 - Servicer Number and Servicer State Number, 297
- Flat-rates, listing, 147
- Focus
 - shifts back to ServiceDesk while working in another application, 78
- Focused Dispatching
 - solution for the large enterprise, 316, 317
- Fonts
 - proportionally spaced vs. fixed pitch, 271
- [Formal claims or invoices](#). *See* [Finished-Form system](#)
- Function keys
 - command buttons on MainMenu serve as labels regarding function, 24
 - using, 220
- Funds Control system
 - Applications Journal, 179, 192
 - deposits, preparing, 159
 - discounts, how to handle, 161
 - entering funds collected, two contexts, 157
 - first mentioned, 36
 - housekeeping, 163
 - how security is achieved, 158
 - incomplete payments, how to handle, 161
 - major discussion, 156
 - one payment, multiple jobs, 161
 - payments, checking in those received by mail, 43
 - reviewing past records, 210
 - uncollectibles, writing off, 162
- Grace periods
 - defaults and customizing (see footnote), 119
- Graph, appointment density, 85
- Graphic Image
 - adding one to your standard invoice setup, 275
- Grid references
 - fine-tuned on basis of address number via info in BlockList, 287
 - NativeGrid. *See* separate heading under this term perhaps not in all cases identical to those in your map book, 288
 - possibly not a one-to-one relationship between grids in paper map and those on-screen, 292
- Hibernating Callsheets. *See* Callsheets
- HighVolume-type designation
 - abbreviations explained, 60
 - such items not indexed to CustomerDbase, 285
- HVC, acronym for HighVolumeClient. *See* HighVolume-type designation
- Immediate call-in method
 - specifying form within Settings form, 227
- Implementation
 - thumbnail of the transition, 43
- Indexes, CustomerDbase. *See* CustomerDbase system
- Installation
 - first time, 20
- Integrated Processes
 - from perspective of SalesEnter process, 169
 - [from perspective of Finished-Form context](#), 188, 193
 - general discussion, setting up from Post-Action Report form, 116
- Inventory control. *See* Stocked-Parts Management system
- Invoice numbers
 - changing from default when creating a job, 72
 - description of basic application, 35
 - not to be advance-printed to your invoices, 73
- Invoice-Format-Instruction file, same as .PRG, 270
- Invoices
 - finding old ones in storage, 171
 - graphic image, adding to your standard setup, 275
 - illustration of nicely-designed setup, 337
 - need to setup for your particular type, 73
 - one of many terms used to describe the basic job document, 47
 - [option to create a formalized, finished version](#), 181
 - optional items to print onto, 272
 - particularities described, 47
 - reprinting after initial creation of the job, 74
 - reversing the assumption as to customer/billing/location in the printout, 274
 - specifying format via a .PRG file, 270
 - storing when completed, 169
 - the physical setup for printing, 267
 - type recommended, 268
 - use of three different parts, 48
- Item-locating to the DispatchMap
 - first described, 28

- major discussion, 65
- Job creation
 - department, specifying, 172
 - described in broad outline, 35, 105
 - distinguished from merely entering job information into Callsheet, 34
 - major discussion, 71
 - tasks performed by ServiceDesk at same time, 72
- Job documents, the general scheme in regard to, 47
- Job Management
 - JobsCurrent form. *See* separate heading under this term
 - major chapter discussion, 105
 - overview, 34
 - rescheduling on existing jobs. *See* Job Scheduling
 - WipAlert - Supervisor, 120
 - WipAlert system, 118
- Job scheduling. *See* also Dispatch and Scheduling
 - appointment notations. *See* separate heading under this term
 - first described, 27
 - from a Callsheet, major discussion, 70
 - from a JobRecord, major discussion, 108
 - on-map versus on-the-Callsheet, 71, 87
 - two different contexts, 37
- Job ticket. *See* Invoices
- JobRecords
 - methods for reviewing, 209
- JobReports form
 - used for PostActionReports, 36
- JobsArchived form. *See* also Job Management
 - explained in broad outline, 35
 - functions within, listed, 121
 - major discussion, 120
 - making a new CstmrDbase index, 121
- JobsCurrent form. *See* also Job Management
 - actions within, 107
 - adding notes to WipHistory, 107
 - entering funds collected, 162
 - explained in broad outline, 35
 - search methods, 106
 - selecting items for display, 105
 - status options, 107
- JobsPerusal form
 - Purpose and operation explained, 106
- Labels for parts. *See* Stocked-Parts Management system
- Learning Modes, usage, 255
- LMOR and similar abbreviations, 56
- Lookup. *See* Searches and Lookup
- Mailing lists. *See* Customer mailing lists
- Main Menu
 - general discussion, 219
 - its command buttons serve also as labels for function keys, 24, 220
- Manual, on-screen version, 218
- Mapping drives within Windows, 224
- Mapping network drives, 313
- Mileage estimates, 225
 - variable service rates based thereon, 93

- Miles-Determined S.Call Rates
 - setting up for, 93
- Model Number lookup, 202
- MoreInfo notes
 - ExtraNotes, general, 56
 - ExtraNotes, specifying location and size from .PRG setup, 274
 - major discussion, particularly in context of Callsheets, 55
 - MoreNotes, 56
- MoreInfo, InventoryPlanner form, 155
- MoreNotes. *See* MoreInfo notes
- Mouse
 - right-click problems, resolving, 301
- [NARDA forms. *See* Finished-Form system](#)
- NativeGrid. *See* also StreetList, one of your custom files
 - displaying it from your DispatchMap, 289, 292
 - references to and why, 289
- Networking
 - dial-up, connecting from a remote machine, 310
 - first described, 16
 - general scheme, 223
 - mapping network drives, 313
 - option for centralized operating files, 315
 - options and methods, a technical discussion, 313, 317
 - separate work that must be completed within Windows, 224
 - sharing a folder-only, rather than an entire drive, 314
 - specifying a FileServer, 223
- New features, we're open to suggestions, 47
- Nighttime lineup of activities, 221
- Non-stock parts, ordering. *See* PartsProcess system
- Old Callsheets
 - where to find them. *See* Callsheet Archive, storehouse for completed Callsheets
- Old Mail, 198
- On-demand dispatch, 91
- Ongoing Services, 50
- On-screen Manual, 218
- Organization of ServiceDesk, basic description, 33
- [Over the counter sales](#)
 - [using the Finished-Forms system to accomplish](#), 189
- P.O. Numbers
 - must be in particular place and format for ServiceDesk to recognize as such, 58
 - searching for job under, archived, 121
 - searching for job under, current, 106
 - setting up to print to separate location on invoice, if wanted, 273
- Paging via alpha-numeric dispatch, 90
- Parts Credits
 - how to apply in warranty claim situations, 159
- Parts labels. *See* Stocked-Parts Management system
- Parts, normal stocking ones. *See* Stocked-Parts Management system
- PartsProcess system
 - checking in parts, 42
 - first mentioned, 36
 - major discussion, 122

- making inquiry/requests, 42
- reviewing past records, 210
- Password, setting or changing, 228
- Paycodes. *See* Post-completion management
- Payments, checking in those received by mail, 43
- PCA-Form, PCA-Entries, etc. *See* Problem-Customer Advisory
- PCA-Form, PCA-Entries, etc, 212
- Point of Sale functions**
 - using the Finished-Forms system to accomplish, 189
- PostActionReports
 - assuring jobs were reported on, 102
 - first described, 35
 - major discussion, 110
 - must be completed for each scheduled appointment, 38
 - the form therefor, 36
- Post-completion management
 - Accounts Receivable. *See* separate heading under this term
 - entering completed sales, 42, 167
 - finding old invoices in storage, 171
 - major chapter discussion, 165
 - overview, 37
 - paycodes, 167, 169
 - reviewing past sales, 210
 - sales, reporting on, 172
 - SalesEnter form, 38, 167
 - SalesView form, 38, 170
 - storing completed invoices, 169
- PRG files, creating, 269
- Printers (i.e., printing machines)
 - type recommended, 267
- Printing from ServiceDesk
 - cut-sheet feeders, 268
 - invoices, creating a .PRG file, 269
 - invoices, the physical setup, 267
 - job invoice at time of job creation, 73
 - making a new invoice on an already existing job, 107
 - printing the entire screen, 207
 - using the miscellaneous print feature, 206
- Problem-Customer Advisory
 - system for tracking bad customers, 212
- Quality of Service (QOS), analyzing, etc., 204
- QuickBooks
 - automated export of information thereto, 304
- QuickEntries
 - first described, 29
 - primary discussion, 58
 - Servicer Number and Servicer State Number, 297
- QuickKeys
 - using MainMenu instead, 219
- QuickLinks from DispatchMap
 - PostActionReport**, 113
 - the ShowJob method, 84
 - the ShowText method, 85
- Range-lines in DispatchMap, 225
- Rate of Earnings form, 202
- Recalls, reporting on, 204
- Red-Flagging Problem Customers, 212, *See* Problem-Customer Advisory
- Remote Access
 - using ServiceDesk from another location, 310
- Reports form
 - first mentioned, 38
 - major discussion, 202
- Restoring backed up data, 217
- Reversed-Assumption Invoice-Format, 274
- Reviewing your backed up data for accuracy, etc., 217
- Right-click problems, 301
- Salary and wage reports, 202
- Sales and A/R functions
 - electing to bypass, 166
- Sales Tax reporting. *See also* Bad Sales Tax situation
 - getting basic information from a SalesReport, 172
- Sales, reporting on, 172
- Sales, transferring info to your system of Financial Accounting, 302
- SalesEnter form. *See* Post-completion management
- SalesView form. *See* Post-completion management
- Schedule of automatic night-time events, 221
- ScheduleList
 - appointments must be listed there, 109
 - defined, 37
- ScheduleList form
 - adding appointments, usually initiated from another context, 95
 - first discussed, 37
 - major discussion, 95
 - making entries re one tech helping another, 99
 - making notation re big jobs, 99
 - re-sequencing jobs, 100
- Scheduling. *See* Dispatch and Scheduling
- SD-Backup
 - first mentioned, 39
 - general discussion, 215
 - setting up in Windows for automatic starting with bootup (see footnote), 216
- SD-Tools
 - first mentioned, 39
 - optional items garage, 272
 - the program's basic function, 217
 - using to create a .PRG file (describing invoice format), 269
 - using to create a Yellow Pages AdList, 281
- Searches and Lookup. *See also* CustomerDbase system
 - Invoice Number based (from the JobsArchived form), 121
 - Invoice Number based (from the JobsCurrent form), 106
 - Methods in JobsCurrent distinguished from those in JobsArchived and from CstmrDbase methods, 106
 - Model Number based (i.e., finding all jobs where work was done on any given model), 202
 - P.O. Number based (from the JobsArchived form), 121
 - P.O. Number based (from the JobsCurrent form), 106
 - Serial Number based (i.e., finding all jobs where work was done on a particular machine), 202

Street Name based (from within the JobsArchived form), 121

Security for your ServiceDesk data, providing, 215

Serial Number lookup, 202

Service call rates
making variable, depending on distance, 93

ServiceBench
auto-insertion of dispatches into ServiceDesk callsheets, 63
[one-click, direct transmission of claims](#), 183
using the ZoneScheduler system in conjunction with, 85
WebBasedDispatchEnabler, extended discussion, how to use, 318
WebBasedDispatchEnabler, use and applicability explained, 103

Servicer Number and Servicer State Number. *See* Finished-Form system

ServiceTicket. *See* Invoices

Settings form
first, basic setup, 20
general discussion, 223

Simplicity versus function, 44

SlideShow demo, 52

Software systems, how ServiceDesk fits in, 9

SourceOfJobs Survey
creating your Yellow Pages AdList, 281
major discussion, 210
turning on from within Settings form, 227

Splitting Commissions on a job, 319, 320

Statements
and dunning letters, both as types of billing reminders, 174

Stocked-Parts Management system
advanced features, 154
beginning inventory, tallying and inputting to, 148
creating your MasterPartsPlan, 145
different parts for different trucks, 155
first mentioned, 36
InventoryControl form, 144
InventoryPlanner form, 144
labels, making, 151
major discussion, 143
ordering restock, 151
restocking trucks, 41, 152
reviewing past records, 210

Street entry from a Callsheet
first described, 27
major discussion, 64
messages flashed during, their meaning, 287
methods of indicating selection type, 64
missing streets, what do to if the one you need is not listed, 288
technical details. *See* StreetList, one of your custom files

Street name, searching for job thereby, 121

StreetList, one of your custom files
adding new entries, 290
areas not covered by map book, how dealt with, 289
BlockList, how it works in the background, 287
city abbreviations, a listing, 323
grid references. *See* separate heading under this term
imperfections in, dealing with streets that are missing, 288
imperfections, entries having no zip or city name, 291
imperfections, inaccurately connected city names, correcting, 291
major discussion, 287
quantity of entries for yours, stated, 324
zip codes used as basis for assigning city names, 291
zip-to-city concordance, your own shown, 323

Support, ongoing, 50

System Requirements, 13

TechInterface form, 209

Technical information
major discussion, 283

Tech's time of arrival, compiling reports, 204

TechWindow mode. *See* PostActionReports

Telephone
area codes, dealing with special situations, 227
autodialing, 65
numbers, notations regarding within Callsheets, 57

Tentative versus definite. *See also* Dispatch and Scheduling
how indicated in DispatchMap and how to change, 86
what it means, specifying during job creation, 72

TimeFrames
customizing the list that's offered for scheduling (see footnote), 96
[Transmitting claims electronically](#), 182

Trucks, indicating specialized types, 155

Uncollectible or Missing Funds
recording fact of, 162

Unit-Info System, 199

Updating your program, 21, 51

Utilities, 215

Variable ServiceCall Rates, 93

ViewBackups, how the feature works, 217

Visiting our office, suggested, 49

Wage reports, 202

WebBasedDispatchEnabler
extended discussion, how to use, 318
first described, 103

Web-Page Updater Utility, 317

Website, where at, 51

Whole-day scheduling. *See* Dispatch and scheduling

Windows
general editing tools, 74
mapping other drives within your network, 224
optimizing interface for use of ServiceDesk, 251

WIP (WorkInProgress). *See also* Job Management
defined, 35

WipAlert system. *See also* Job Management
described generally, 118
grace periods, defaults and customizing (see footnote), 119

WipHistory
section in job record describing, 107

Withholdings, calculating, 203

Work order, same as invoice, etc., 47

Zips, a small bonus program, 218
Zip-to-City concordance, your own shown, 323

ZoneScheduler system, 85, 102

EXHIBITS


UNEDITED PRINTOUT
FROM AN ANSWERING SERVICE USING THE "STARTEL" SYSTEM
SHOWING A FORMAT
TAILORED FOR PROPER INTERFACE WITH SERVICEDESK

In: 12:17p MON FEB- 5 CMG
Out: 1.04p MON FEB- 5 PM r/p
Nm: LARRY GILL
Telephone Number.....: 496-5495
Adr:
Cty:
Prblm/Rqst: PLEASE CALL
Item(s) Type.....:
Item(s) Make.....:
Appointment Dt & Tm.:
How will this person pay?:

In: 12:19p MON FEB- 5 JPB
Out: 1.04p MON FEB- 5 PM r/p
Nm: OUELLET, PAUL
Telephone Number.....: 714 863 0560
Adr: 29045 MODJESKA PEAK X268
Cty: TRABUCO CANYON
Prblm/Rqst: SPIN CYCLE DOESN'T WORK
Item(s) Type.....: WASHER
Item(s) Make.....: KENMORE
Appointment Dt & Tm.: TUESDAY 4-7
How will this person pay?: CHECK

In: 12:50p MON FEB- 5 JPB
Out: 1.04p MON FEB- 5 PM r/p
Nm: CINDY, AMERICAN HOME SHIELD
Telephone Number.....: 800-776-4663
Adr: RICHARD AND NANCY SWENSON
Cty: P.O. # 10916820
Prblm/Rqst: WHEN YOU GET THE FAX HAVE
TECH COLLECT \$70 INSTEAD OF \$35, SHE
Item(s) Type.....: OWES FOR PREVIOUS
Item(s) Make.....: S SERVICE CALL
Appointment Dt & Tm.:
How will this person pay?:

ILLUSTRATION OF A NICELY-DESIGNED
INVOICE FORM AND PRINTOUT SETUP



Aardvark Appliance Service

31878 Camino Capistrano, S.J.C., CA 92675 (949) 493-6069 fax, (949) 493-5169 voice

Customer Name AMERICAN HOME SHIELD		ID # 29185022	64521
Address P.O. BOX 866		Home Phone 800-326-4357	Date Written 02/23/00
City CARROLL, IOWA 51401		Business Phone 800-776-4663	Date Scheduled 24 THU 1-4
Location Name SMITH, JOHN AND JILL		\$35 DEDUCTIBLE	Date Completed
Address 123 SOMERSET LANE [921J5]		Home Phone 493-5528	Item(s) Type REFER
City ANYWHERE, SOUTH DAKOTA		Business Phone 714-588-5225	Item(s) Make KENMORE
Complaint/Request MAKES A BAD NOISE AND IS NOT COLD ENOUGH, PLEASE KNOCK LOUDLY. DOORBELL IS NOT WORKING			Model Number
			Serial Number

SERVICE NEEDED/COMPLETED	LABOR PRICE

PARTS NEEDED/USED	PARTS PRICE

Techy/Date	Start Time	End Time	Elapsed Time	Parts Total
Techy/Date	Start Time	End Time	Elapsed Time	Tax
				Service Call
				Labor Total
<p><small>Our limited labor warranty in regard to correcting defects in our workmanship is lifetime -- there is no time limit. If a part fails owing to a manufacturer's defect, however, our labor is covered for just 90 days. There is no warranty against additional or more extensive repairs, against other problems that may occur on the same machine (even if involving identical symptoms), or against incidental damages. Parts carry the manufacturer's warranty, which is usually one year.</small></p>				Estimate for anticipated repairs:
				Invoice Total

TECHNICIAN'S SIGNATURE

Funds Rec'd (date, kind, amount)

CUSTOMER'S SIGNATURE

Funds Rec'd (date, kind, amount)

Balance still due (if any)