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 NARDA[136] 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.

[136]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 55]. 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 114]. 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).

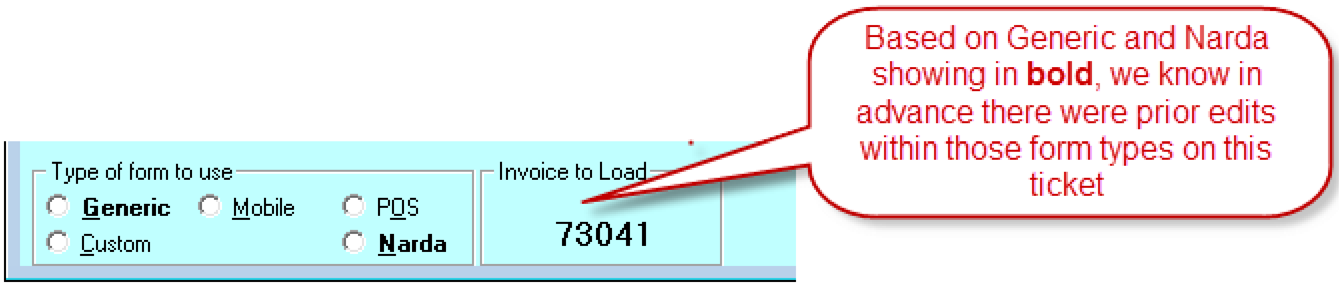
iii. Details on the Imported-data-Auto-Fill Function, and the Re-Load Alternative

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
POS
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.[137] 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.

[137] 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 Rossware'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 inernally-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.

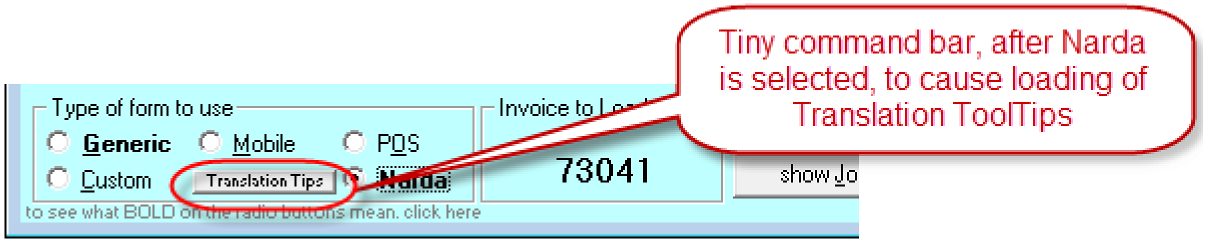
It's because of the above translation dynamics that, in some instances, you might find yourself in the conundrum of wondering: what box in Rossware's NARDA must "X" be in to show up in Box "Y" after the Rossware-assembled claim is handed off?

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.


Type of form to use
Generic
Mobile
POS
Custom
Translation Tips
Narda
to see what BOLD on the radio buttons mean. click here
Invoice to Load
73041
show Jo
Tiny command bar, after Narda is selected, to cause loading of Translation ToolTips

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.

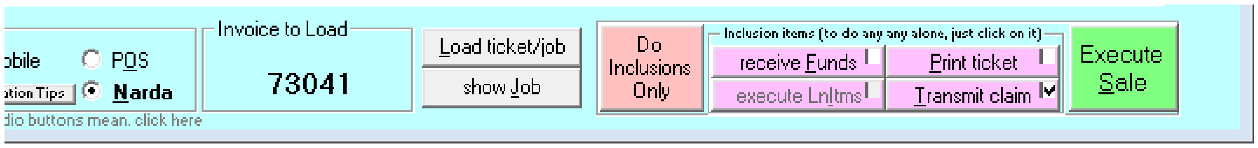
Thus, we fully expose the translation that happens on our end. If the need arises, you'll need to seek help from the processing entity for exposure of what's happening on theirs. Most typically, though, you'll find success easier to achieve. We're explaining these resources because they are ultimately 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).[138]

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,

[138] 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.

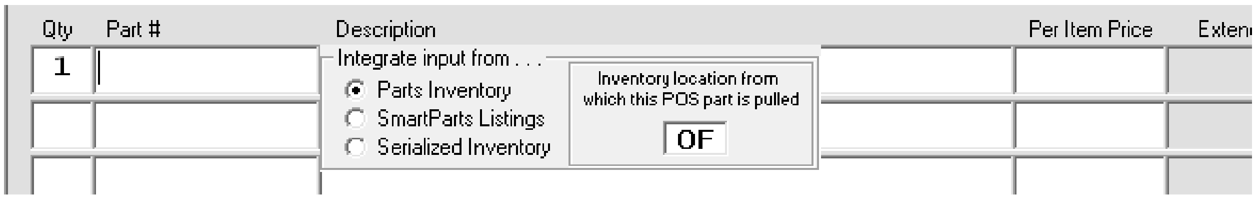
Aside from the above, please notice that just as soon as you conclude a Direct-POS transaction, the interface goes right back into a mode waiting for input of a sales person's two-letter code, to begin the next transaction. It is thus a system always-ready to perform for a never-ending succession of such transactions.[139]

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:



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.[140]

[139] 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.

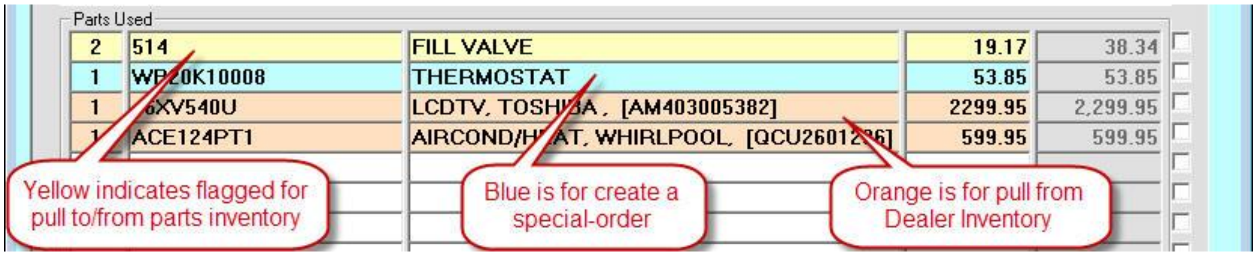
[140] SD-Dealer is one of the supplemental programs that may be acquired to work with ServiceDesk, but is not actually part of it. It's specifically designed as a mechanism to manage serialized inventory, and is basic but elegant. If you are interested, please contact Rossware for details.

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.[141]

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).



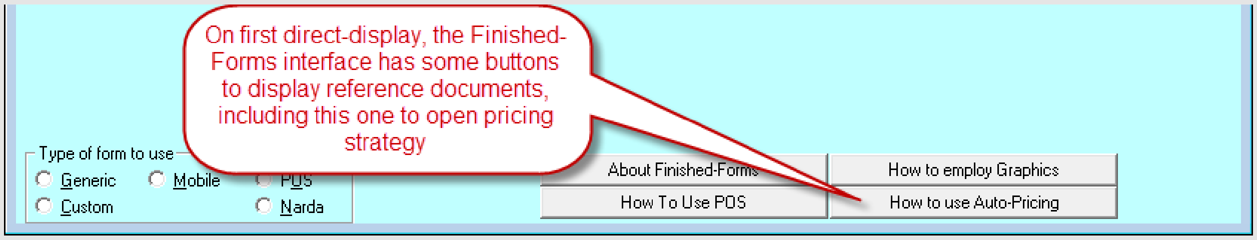
Parts Used

2	514	FILL VALVE	19.17	38.34
1	WP20K10008	THERMOSTAT	53.85	53.85
1	6XV540U	LCDTV, TOSHIBA, [AM403005382]	2299.95	2,299.95
1	ACE124PT1	AIRCOND/HEAT, WHIRLPOOL, [QCU2601296]	599.95	599.95

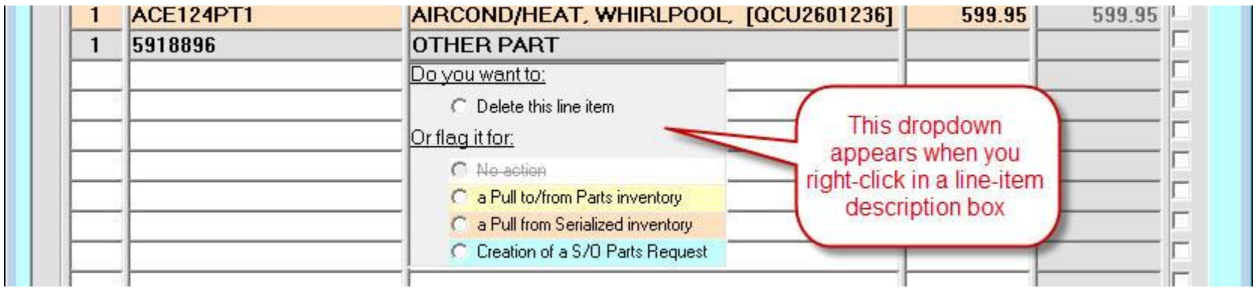
Yellow indicates flagged for pull to/from parts inventory
Blue is for create a special-order
Orange is for pull from Dealer Inventory

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:

[141] 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:


On first direct-display, the Finished-Forms interface has some buttons to display reference documents, including this one to open pricing strategy
Type of form to use
Generic
Mobile
POS
Custom
Narda
About Finished-Forms
How to employ Graphics
How To Use POS
How to use Auto-Pricing

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.

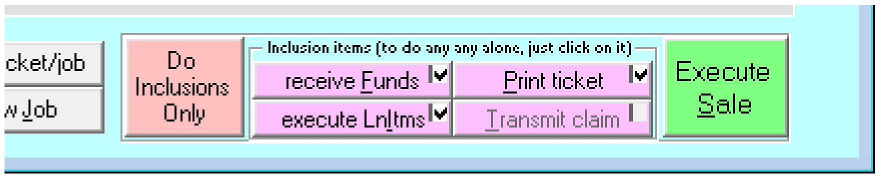
Our overall point is you can use such mechanisms to accurately populate line-items within your POS form, so that it accurately reflects what you're selling to your customer, and how the interaction of such sales should relate with other elements as being managed within ServiceDesk. However, nothing actually 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 LnItms*:"



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-carets 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 131).

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 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".

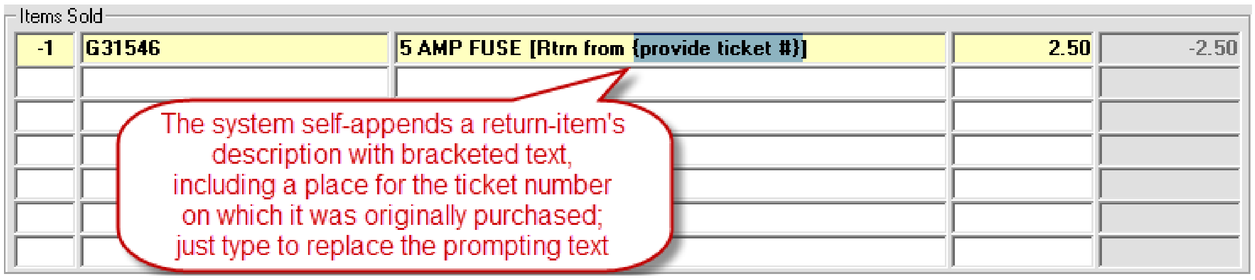
Please also feel free on this new ticket to include items that you are newly selling to the customer. In other words, there is no problem with a single new ticket involving both items being accepted in return (negative quantities indicated) and items currently being sold (positive quantities indicated).

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 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]

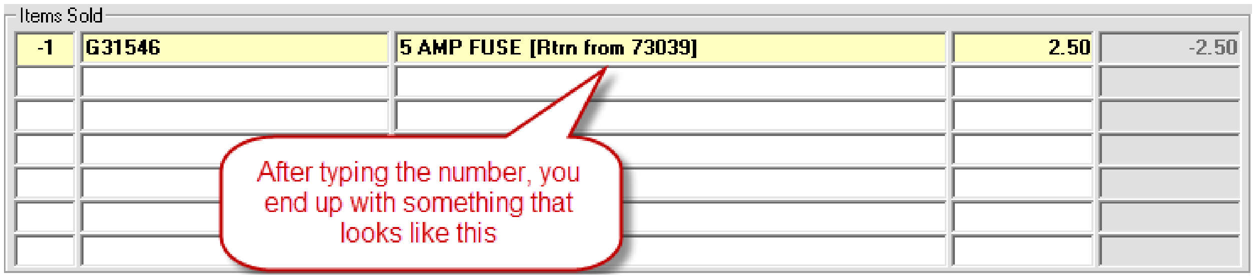
[142] We'll readily admit that for true integrated completeness, these added auto-ties should be present. It's also true, though, that ServiceDesk is primarily a service-management system, as opposed to being the utmost/best-possible POS system. If not for the fact so many other developmental demands 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

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 is but one potential rough-spot that may arise on the financial/money side of returns. It's if you decide to refund money back to the customer by writing a company check. It's rough only in the sense that 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 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.

vi. Customizing the Finished-Forms Interface

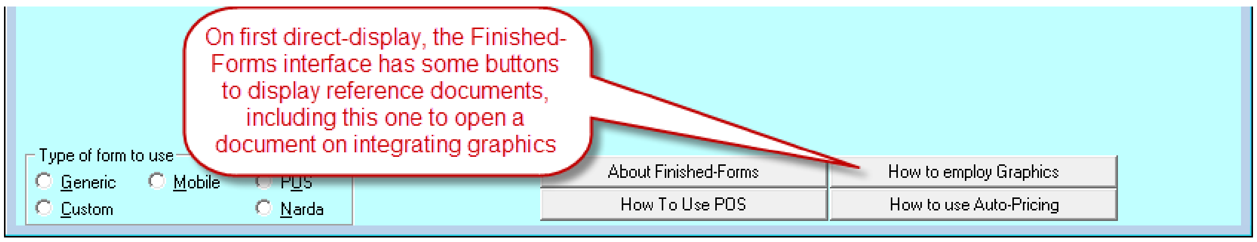
We want to again distinguish FinishedForms from the up-front ticket-creation context. In the latter, we've invested intense programming capital to make an output that is almost infinitely (and very easily) customizable to

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.

[143]Exceptions 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.

user-preference (see page 267). 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):


On first direct-display, the Finished-Forms interface has some buttons to display reference documents, including this one to open a document on integrating graphics
Type of form to use
Generic
Mobile
POS
Custom
Narda
About Finished-Forms
How to employ Graphics
How To Use POS
How to use Auto-Pricing

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*."