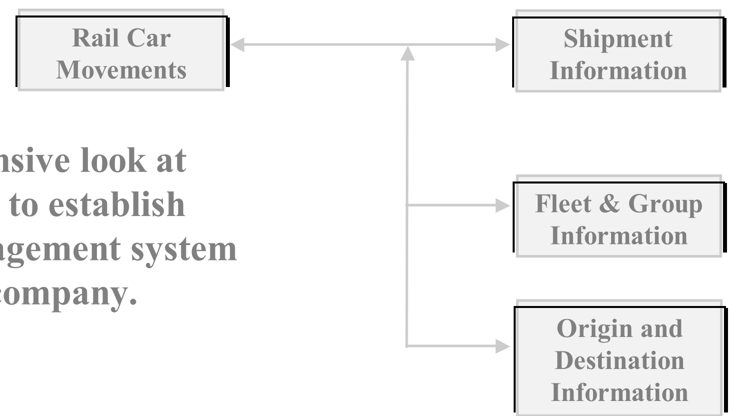
Rail Fleet Management Systems

A Rail Shipper's Primer



A comprehensive look at
what it takes to establish
a rail fleet management system
for your company.

CEVA
13354 Midlothian Turnpike
Suite B-102
Midlothian, Virginia 23113
804-378-2095

NOTICE

The information contained herein is the sole property of any shipper who has ever tried to control a rail fleet. Not one word of this document is proprietary and you may feel free to reproduce or transmit in any form or by any means, electronic or mechanical, any part of it without the express written permission of CEVA or McKamie Wilson. Our only request is that if you come up with any great ideas related to rail fleet management as a result of reading this booklet, you let us in on it or pass them on to someone else who can use them.

McKamie Wilson
CEVA Operations Center
04 Apr 2003

CEVA
13354 Midlothian Turnpike
Suite B-102
Midlothian, Virginia 23113
804-378-2095

Print History: April, 2003
January, 2003
May, 2002
November, 2000
March, 2000
February, 1998
May, 1995

Rail Fleet Management Systems

A Rail Shipper's Primer

Contents

Contents (Cont'd)

Contents (Cont'd)

Contents (Cont’d)

1. Understanding the Problem

If you think you have been getting along fine without a computerized rail fleet management system, think again. You've been wasting both money and resources!

Why? Typically, the purchase price of a hopper car or a tank car is between $55,000 and $65,000. The monthly lease payment on that same car can be $500-$600. For a fleet of around 150 cars, if you lease the rail cars, you have a $1 million dollar annual expense; if you own the rail cars, you have a $10 million dollar asset. In either case, that's a lot of money when you are operating with no controls, no historical records, and no forecasting ability.

Rail Fleet Management Systems: A Rail Shipper's Primer

If your company ships products which require more specialized transportation (for example, stainless steel tanks or rail cars with special insulation, heating, or cooling characteristics), you could be paying as much as $1,500 for the monthly lease or $150,000 as the purchase price. Now, for that same 150-car fleet, you're talking about an annual expense of $2.7 million or an asset of $22.5 million. At those prices, simply reducing your fleet size by one car or squeezing an extra trip per year out of a car can yield tremendous savings.

The information in this booklet is the product of more than twenty years of experience in dealing with corporate officers, transportation department managers, traffic managers, rail carriers, third-party solution providers, consultants, and corporate computer folks. We are not trying to sell you on any particular computer system or information system architecture. However, we are definitely trying to sell to you the concept of using computer systems to support rail logistics. Along the way, we'll try to bring a little light and a little levity to the very serious subject of managing rail transportation.

So, just sit back and relax. This is not rocket science. We're not going to load you down with buzzwords, acronyms, rail jargon, or computer techno-speak. Although it is impossible to talk about either rail management or computers without using some specialized terms, we will explain the few we do use as we go along.

The Starting Line

In our experience, the real problem in creating a rail fleet management system is that the computer folks and rail folks have never had an opportunity to work together. The need for good rail management has been around for years. After all, the business shipping community has been using rail transportation in this country for well over 125 years. But, computers and computer applications have rarely been specifically designed for the transportation group.

We are offering this booklet primarily for those transportation people who have been thinking about how a computer might help them do their job better. If you are a computer systems analyst who has been given the responsibility of providing a computer system for the rail traffic department, you will find this booklet to be a valuable reference. Also, if you are a corporate officer or an information systems manager, this booklet will quickly help you become conversant with both parties involved in the creation process. If you don't know anything about rail management or computers, this is a good place to start. The more you know about your rail-related processes and computer possibilities, the easier it will be to get a rail management system up and running in your own company. There is a boatload… no, make that a trainload… of stuff to learn!

Here is what we're going to do in the chapters that follow:

- Give you a general idea of what traffic people encounter in their daily rail management processes.
- Tell you what kind of movement information and reporting structure already exists so you can take advantage of it.
- Give you some guidelines for building a basic computer rail tracking system that can help you get your rail information organized.
- Describe some ways to use the basic system as a foundation for a full-fledged rail fleet management system that could easily save you the cost of the system in the first year of its operation.
- Summarize the available support services including rail management software, third-party solutions providers, and industry groups.

Two Great Truths

From day one, you will discover that there are two constants in this new world you are entering:

- Transportation people are concerned only with keeping their rail cars moving efficiently from origin to destination. Generally, they don't know anything about computers...AND THEY DON'T CARE.

- Computer people are concerned only with keeping their computer systems running efficiently from morning till night. Generally, they don't know anything about rail cars...AND THEY DON'T CARE.

You have probably heard about the book that declares, "Women are from Venus, Men are from Mars". Everyone knows that men and women are very different. But, we also know that men and women can work well together when they focus on a common goal. Compared to getting men and women to work together, getting rail people and computer people to work well with each other should be a snap. Yeah, right.

So, given the two constant truths above, did you notice the important common point? No, it's not that they both don't care. Rather, both rail managers and computer systems people are primarily interested in the efficiency of their respective tasks. This is, of course, a good trait for employees to have. And, it is this common theme that we are going to expand upon in the rest of this booklet.

As we stated in the previous section, the two groups probably have never really had a chance to work together as a single unit with a common goal. If you use the concepts we offer in this booklet, you should be able to blend the best talents you have from these opposite worlds and come up with a system that will provide exactly the information your transportation department needs.

Putting the Team Together

Let's assume that you have ignited a few fires of interest in computerizing some rail management functions. The first thing we need to do is put together a "logistics systems design" team composed of two or three people from the transportation department and two or three people from the information systems department. Obviously, you will be selecting members from both sides of our two great truths dilemma, so you want people from both sides who have energy and enthusiasm, but more importantly, you want people who can work well with others. If you select the team members appropriately, you will eliminate half of your problems before you've even started.

In our experience, it seems to work out best if you select a team leader from the transportation side. The slant of the project, if there is one, tends to lean toward the views of the project leader. Also, computer programmers will struggle to participate in the design effort if the language of the design stays in the realm of the transportation group even if they don't grasp all of the concepts and problems at first. This is because they know that the ultimate success of the project depends on how well they can translate the team's design into working reality. If an information systems person heads up the project team, the design directions tends to stray toward the technical aspects of software and hardware and the transportation members can very quickly "zone out" and become invisible participants in the planning and design effort. One additional recommendation for the team leader: this person should have the respect of management and the authority and ability to make design decisions for the transportation department. If not, the rail system design will take five times longer than necessary. At a minimum!

! **A note for the Transportation team members**: You should be positive and direct and honest. This is a chance to learn some valuable computer design skills, but this system is being built for your department and you are going to be using it. Computer people can get very focused on the bits and tweedles of system logic, but they will always recognize and respond to sincerity and respect. If you have questions, ask them. If you don't understand the answers, say so and ask for an explanation in English. Remember, the only stupid question is the unasked question.

! **A note for the Information Systems team members**: You should be positive and be open to new ideas. Try to stay out of the mindset that this project will be just like one you have done before, but with a few menu changes. If another team member asks you a question, never say NO immediately. Think about your response carefully, from their perspective. If you have to research the answer and get back to them later, tell them that, **and then do it**.

Information Distribution

Now that our design team is ready, we need to get all the interested parties together and start a general discussion of what would happen if the rail logistics information could be made more easily available to other groups. Set up a "blue-skying" meeting with representatives from the three departments or groups described below. In the meeting, you should formally introduce the logistics system design team. Make sure everybody knows that the team is there mainly to listen and to provide answers to "Can we...?" questions posed by the other members of the group.

The following is a series of questions you might use to get the discussion focused on the high level nature of the meeting. We will concentrate on the three major areas of your company that will be affected by a rail fleet management system (other than your traffic and computer departments): Sales and Marketing, Production, and Management. This is not a meeting to discuss the different codes you will use to identify a product or the formats of the reports. That will come in due time. What you are trying to do here is find out the different "needs" and "points of view" relative to the information that flows from the logistics process.

The first group on the list is the Sales and Marketing department. You probably already know exactly how much it costs your company to manufacture a single unit of your product and how much you sell it for. But, do you know how much it costs to ship a single unit? Do you know how much it costs to sell a single unit? Do you know the difference in the cost to ship a single unit of your product to a client on the West Coast as compared to the cost of shipping a single unit to a customer on the East Coast? How long does it take to get there? How long does your East Coast customer hold your rail cars at his facility before he releases them? Is he storing the product in your rail cars until he's ready to use it? If so, how much is that costing your company? How many trips per year do your cars make? Could they make more, thus allowing you to reduce your fleet size? Have you noticed any trends in either sales (customer or product related) or production (supply, product, or corporate related)?

Next, we need to look at the information from the Production point of view. In today's environment, you can no longer use the "Dorito-theory" of management: "We'll just make more...". There are production limitations including, but not limited to, manpower, equipment, and cost of raw materials, and there are chains of supply involving inventory, storage, and delivery. What is the production cycle time of your products? Does this fluctuate for any reason? What are the limiting production factors? Do you have enough cars to complete the shipments? Will they be available when you are ready to load them? Are the cars sized correctly for the loads you are delivering?

Finally, we have some questions for the Management group. This is, in general, a group that may have representatives from several different areas of your company. Are you meeting delivery schedules? Are your customers satisfied? Are your shipments cost effective? What information do you need to answer most customer service questions? Do you have any complaints from customers? From production? From transportation? Can you identify shipments individually? By customer? By product? By corporate division? By geographical area? Do you need more information? What is it? Who needs it? Why? What's your gut feeling about the shipping situation in general? Can you support that feeling with facts?

Remember, the emphasis on this meeting is the discovery of information, who uses it and for what reasons. This meeting may very well determine the direction you take for the rest of this project. If your logistical information requirements outside of the transportation department are inconsequential, then you probably only want to concentrate on a small but dependable tracking system. If you have a lot of interest in the information across several departments, then you will need a more robust system capable of supplying information to a diverse set of users.

The Mysterious Art

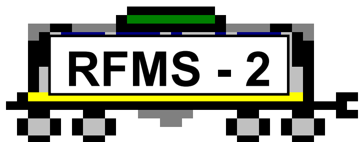
Now that we have the blue-sky sessions out of the way, let's bring the design process back down to earth. Computer systems, relatively speaking, have just emerged from their infancy. For most of the time that computers have been used in corporations, the programmers and systems analysts have concentrated their efforts on crunching numbers and handling employee information. This was their best area for most efficient use. It is only in the last few years that computers have entered the rail management arena. More often than not, it was an adventurous traffic person who found that a spreadsheet program on a personal computer could help record information about volumes of product delivered, the rail car initials and numbers, scheduled and completed inspections, or counts showing the number of rail car shipments by customer or by product.

Most traffic groups operate like a subsidiary of their own company. Other departments do not understand what goes on in "transportation" AND THEY DON'T CARE as long as there are rail cars lined up for loading and the shipments arrive on time. The customers are happy therefore management is happy therefore transportation is doing its job. Rail management has been treated as a mysterious art because it is *reactive* management (management by exception), and its successful application is totally dependent upon the experience and the dedication of the traffic managers.

But now times are changing. Corporations are tightening the reins on all aspects of their business. Production lines are being scrutinized, computers are being upsized or downsized to better fit the information service requirements, employee benefits are coming under the microscope, and training is being cut back. Retirement programs are taking the experienced traffic people out of the department before they have had the opportunity to pass on their knowledge. The corporate belt is being cinched in a notch at a time, and the transportation experience is disappearing at a frightening rate.

At long last, computers are being introduced into transportation, not as a bone tossed to a hungry dog, but formally, with corporate support and dedication and, most importantly, with a separate line in the company's budget. The timing is perfect. The search for efficiency in business shipping practices comes along just when the speed and power of computer systems can match the requirements of busy transportation users. As traffic managers begin to see what a computer can do for them, the ideas for more and better uses will burst forth like flowers in the springtime. After the computer systems people implement their first successful transportation system, they will take on a new attitude. It's nice to know that the work you do is appreciated.

For all of this to happen, you have to start someplace. The rail people have to be introduced to the computer people as partners and a common goal has to be established before the process can begin to blossom. That's what this booklet is designed to do. We will provide enough information for you to build a small rail tracking system. Once the pieces are in place, we'll show you how you can expand this foundation into a full blown rail management system. The rest is up to you. And soon, the mysterious art becomes the art of efficiency.



2. Basic Rail Management

At a glance, shipping via rail seems like a simple process: you dump your product into a rail car, the rail carrier pulls the car to your customer's location, and they take your product out of the car. What could be simpler?

The next time you think something is simple, imagine trying to tell somebody how to tie a shoelace...over the telephone. Did you say the string on the right went over or under the loop on the left. Or, how about another simple thing: have you ever tried to follow a recipe for stir-frying vegetables? You didn't get the pan hot enough the first time either, did you! Simple things aren't always so simple.

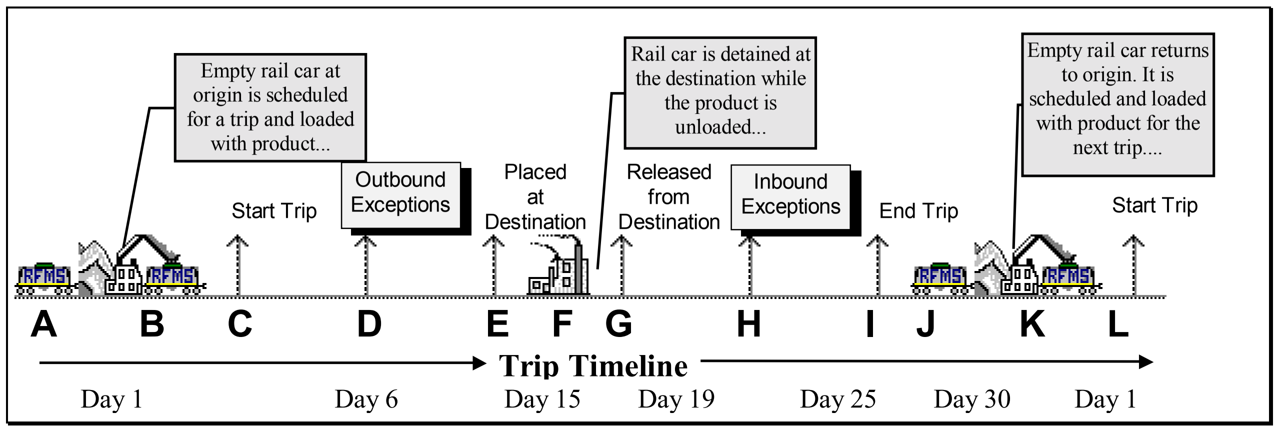
Rail management is like that. To most people outside the traffic department, shipping by rail appears to be a no-brainer. After all, the railroads do most of the work. All you have to do is get the stuff in the rail car and then wait for it to be delivered. Just like UPS. Yeah, right.

The truth is shipping products by rail is a lot like baby-sitting a two-year old kid. You know something is going to happen, you just don't know when. In fact, everything usually goes perfectly...until you turn your back for a moment. The next thing you know you're standing knee-deep in the worst mess you've ever seen. That's when you have to react with a high degree of skill and efficiency. It's called *management by exception* and it is practiced to a fine art by most rail transportation managers.

The trick to managing by exception is to recognize when you have a problem and to respond <u>immediately</u> with the corrective action. To do this repeatedly, day in and day out, takes experience and dedication. For example, when you suddenly notice that a rail car is moving north on CSX and it should be going south on Norfolk Southern, who do you call first, the CSX or the NS? Knowing the answer to this takes time. Not knowing the answer is why rail transportation management seems to be such mysterious work.

Understanding Reality

To begin the process of computerizing rail management, everyone involved must first understand the realities of moving your products over the rails. There is a lot going on out there. Take a look at the following diagram.



In general terms, rail cars are in constant movement, cycling from an origin point to a destination point and then back to the original or on to another origin point. They should only stop moving when they are loaded, transferred from one carrier to another, or unloaded. The daily cost of a rail car is the same whether it is moving or standing still.

The logistics events in this diagram are labeled "A", "B", "C", etc., and they move from left to right along a timeline that encompasses an average trip for a rail shipment. It doesn't matter whether you use tank cars, flat cars, hoppers, or gondolas, ship products to customers or receive material from suppliers. All rail shipments follow a pattern similar to the one shown above. It's only where you are located along the timeline that causes your perspective to change.

A *trip* for a rail car is one complete cycle according to your perspective. If you own or lease your rail cars and you are shipping material that your company produces, this diagram probably describes your perspective exactly. A trip for your cars would be the **A** to **J** cycle that includes the loading of a car at the origin, the movement to a destination, the unloading at the destination, and the movement back to the origin.

However, if you are receiving material shipped via rail cars assigned to you by a rail carrier, you might be interested only in the **C**, **D**, **E**, and **F** portions of the diagram. From this perspective, a trip would include the movement of the loaded rail car from its origin at your supplier to your facility where it is unloaded. Your only other concern might be the **G** portion if, perhaps, a rail carrier is slow to remove cars from your rails, thus preventing you from bringing in another loaded car.

Another possibility is if you own or lease rail cars that you send out empty to a supplier's facility where they are loaded and returned to your plant. If this describes your situation, then the trips your rail cars make start the cycle at the **F** point and end with **E**. (In this case, you should swap the word pairs "origin and destination" and "inbound and outbound" in the diagram to make it fit your situation.) Now, follow along as we describe the events in the diagram as they occur across the timeline. You'll quickly begin to see the enormous complexities that rail managers face every day.

A. The empty rail cars at the plant are scheduled and prepared for loading. In some cases, this means cars have to be cleaned to remove any product residue from a previous loading. They are always inspected to make sure that all the doors and valves are working properly and that they close securely. Particular attention is paid to specially equipped cars that are designed to protect your product for extended periods of time.

B. The rail cars are moved to a loading spot and filled with product. There was a time when "monitoring the loading process" meant having a man stand on top of the rail car with a stick in his hand. When the level of product in the car reached a line painted on the bottom of the stick, the car was considered to be full. There are now better ways to do this job. For example, some companies are beginning to utilize weighing or filling devices that are monitored during the loading process to make sure each rail car is loaded consistently. Also, computers can be used to calculate product expansion, contraction, or settle values more accurately. This is not so much a factor at load time as it is a parameter of the forecasting process.

C. The rail cars are *released to the primary carrier*. That means the rail carrier comes onto your plant and pulls the rail cars into a switch yard where they can be added to a train. This is considered to be the beginning of the outbound segment of a trip. Trains arrive and depart a switch yard or terminal according to established schedules, just like airplanes land at and take off from an airport. Some cars, destined for local facilities, are dropped from a train as it enters the switch yard while other cars, leaving local facilities, are added to a train as it leaves the switch yard.

D. Your cars travel along the outbound leg of your trip cycle. Literally dozens of things happen to the cars as they move. These happenings are called *actions*. If an action occurs that is not a part or your plan for that rail car, it is called an *exception*. Imagine that you are driving coast to coast on the interstate highways. If you take a wrong turn or get off one exit too soon, that's an exception. And, you correct it as soon as you realize the problem. The same is true with rail traffic. One of your cars can get dropped from its train by mistake. If it gets attached to the next train that comes along going in the same direction, the exception is fixed after only a short delay. However, it could get attached to the wrong train and go miles in the wrong direction before it gets corrected. It could get derailed. It could be damaged. It could need a *running repair* (also called an *in-service repair*) like brake shoes or couplers or an axle. It could have a spill. It may even need some repairs to external parts like doors or handles, internal parts like the valves or the special conditioning equipment that protects your product. These are all exception possibilities. These exceptions and the problems that go along with correcting them are the reason traffic people have gray hair. (Unless, of course, they also have teenage children! That's a different kind of exception and requires a different booklet, much longer than this one.) If a rail car is transferred from one carrier to another (e.g., CSXT to NS, or BN to UP), the switch yard or terminal (identified by city and state) where the transfer took place is considered to be a *junction*.

E. The rail cars arrive at the destination terminal where they are dropped from the train and prepared for movement onto the rails of the company receiving your products. Again, car movement between the switch yard or terminal and the receiving company follows a predetermined schedule. In most urban settings, trains come and go all day long, but movement to and from local facilities usually takes place at night, especially if the railroad tracks cross (automobile) roads and highways.

F. The rail cars are unloaded at their destination company. Sometimes a company will bring the cars straight onto a spot and unload them, and sometimes the rail cars will sit for hours or days before they are unloaded. This inactive period is called *detention* and some suppliers charge their clients for the delay. Others use the threat of detention charges as an incentive for the receiving company to get the car unloaded. Still others build the cost of the delay into the shipping costs or the cost per unit of the product. If you don't do something to let the receiving company know that you are aware of the delay, they will tend to take advantage of you, storing the product in your cars until you complain. This is not finger pointing, it's just reality.

G. After the rail cars are unloaded at the receiver, the rail carrier is notified. This movement marks the end of the detention period. The cars are then pulled back into the switch yard and attached to another train. This is the beginning of the inbound, or return, segment of the trip. It generally takes longer for an empty car to get back than it does for a loaded car to go out, even if the cars traverse the same rails in both directions. The logic behind this is that loaded cars have priority over empty cars.

H. Your empty cars return along the inbound leg of your trip cycle subject to the same exception problems as described in the **D** portion above. Exceptions can occur anywhere, but the highest probability is that you will encounter a problem during either the outbound movements or the inbound movements of the trip. It makes sense because these two portions make up more than 90% of the rolling time of any trip cycle. If you ship to a client that is five days away and it takes that client ten days to unload your cars, most of your problems are still going to occur during the ten days of rolling time (five days out, five days back).

I. The rail cars return to the origin on the inbound train. They get dropped from the train at the switch yard. Subject to the rail carrier's schedule, they will be moved back to your facilities.

J. The rail cars are placed onto the tracks at your facility and this marks the end of the trip cycle. There is usually a delay here between the end of one trip and the beginning of the next trip. This delay is called *dead time*. If you have a high percentage of dead time as compared to actual trip time, you may have too many rail cars in your fleet.

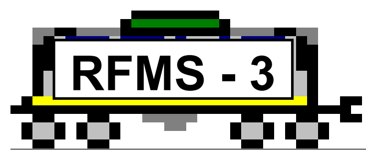
K. and L. This marks the beginning of the next trip and corresponds to the **A** and **B** portions described above.

A Model of Efficiency

What we have just described is the real activity involved with rail logistics. Now what we are going to do is *model* this real activity with the computer. No, modeling does not mean we have to build little rail cars out of balsa wood, wire, and that glue that pulls the skin off your fingers. It just means that we are going to recreate and track the movement of the cars and the situations they encounter with a computerized system using electronic representations: letters, numbers, and codes.

Each rail car has an identification code (its initials and numbers), each logistical event (**A**, **B**, **C**, etc.) will have a code and a description, each rail carrier has a unique set of initials, and each origin, destination, and junction has a specific identifier (city and state). Then, if we enter into the computer a series of the events we expect to happen, that series is called a *route*. The route is our plan for the next trip for a given rail car.

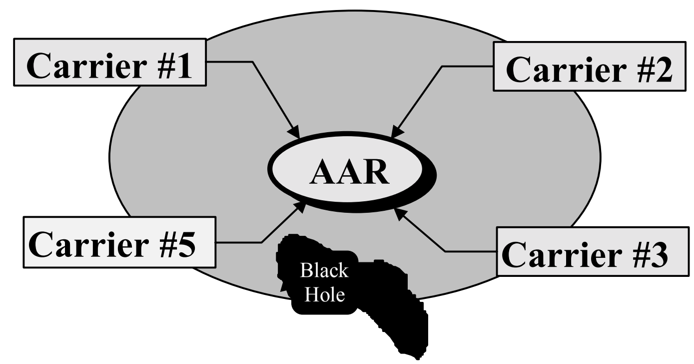
We can even get more sophisticated and attach times (usually measured in days, but hours are more accurate) to each separate event in our route. This will let us measure the actual movement of our rail car against our expected performance. Ah, life is good. We can do all kinds of things with the data. And, it's actually easier than you might think. Read on.



3. Rail Car Tracking

Railroad tracks go everywhere, in every direction. The tracks carry thousands of rail cars from companies, to companies, covering all points on the compass in the United States, Canada, and Mexico. It’s like ants in a huge anthill. So how can you keep tabs on a few cars of your own in the midst of all this seemingly unrelated activity? The railroads and the AAR do it for you.

The Association of American Railroads (AAR) receives movement-by-movement information on every rail car that rolls over most, but not all, tracks in North America. Even though the tracks are owned and operated by independent companies, they still report the moves to the AAR. In most cases, this reporting is done electronically, computers talking to other computers, through an established set of standards for electronic data interchange (EDI). However, in a few cases, the information transfer is not so sophisticated and the movement reports get to the AAR through whatever means are available to these slow reporters.



There are also some small short-line carriers that do not report at all. When your rail cars enter these *black holes*, the last information you get is from the reporting carrier where it met a junction with the non-reporting carrier. Your cars virtually disappear until they move back onto another reporting carrier's tracks. This can be very aggravating, especially when you are trying control your cars down to the last detail. But for now, it can't be helped and we just have to work around the problem. Fortunately, the slow reporters and black holes can't stop progress. They are just part of life's little quiver of slings and arrows.

If you want to know more about reporting carriers, you should contact the National Industrial Transportation League in Washington, DC. The NITL publishes a "guide to shippers, carriers, and other interested parties" called the **Rail Users' Manual**. This manual is invaluable in supporting the construction of any rail-oriented computer system.

THE NATIONAL INDUSTRIAL TRANSPORTATION LEAGUE
1700 North Moore Street, Suite 1900
Arlington, VA 22209
703-524-5011
www.nitl.org

CLMs, Codes, and Electric Things

As we said above, every time a rail car moves, information about that move is reported to the AAR. Each individual movement report is called a *sighting*. The sightings are transferred to the AAR in standardized electronic record formats called *Car Location Messages*, or *CLMs*. There are several different CLM formats, but basically, each sighting record contains the identity of the rail car which moved, the date and time of the movement, the city and state where the movement occurred, the rail carrier responsible for the car, and two special code fields.

The first code field indicates whether the rail car is loaded or empty. The code is "**L**" or "**E**", respectively. This simple piece of information is incredibly valuable to a rail manager. If you know your rail cars' movement characteristics, then the loaded/empty status of a car and its sighting location can tell you if the car is inbound or outbound, soon to be available, headed for a customer, just loaded, or just emptied.

The second code field is just as useful, but it's a bit more complex. It is formally called the *railroad sighting code*, but it is also known as the *action code* or *status code*. Regardless of what name is used, this little beauty tells you WHY the car was moved, or, in some cases, not moved. The sighting codes are usually self-explanatory. For example, the code "**H**" identifies a rail car that is "delayed or held". A good computer system should display the code along with the text description so that the system users never have to struggle to remember the more cryptic codes like "**K**", which is described as an "intermodal interchange". These railroad sighting codes are put into the sighting records by the rail carriers. These codes show activity, but they do not show responsibility. For example, there is no specific code showing a carrier-caused problem like "Sorry, we rolled your rail car off a cliff!" Likewise, there is no code showing shipper responsibility like "You jerk! You're asking us to put an unsafe car on our rails!" There is more information about the fields and codes in a CLM in the **Files and Formats** section.

All this is only of passing importance compared to the one little subtlety that makes rail tracking different from tracking trucks: *A record of every individual movement is transferred to the Association of American Railroads and you have the right to get a copy of every CLM that involves your rail cars.* Wow! Talk about power to the people! If the importance of this fact doesn't strike you at first, imagine being able to call an airline to find the last radar position showing Flight #1234 while it's still in the air. The CLM records allow you to follow each and every one of your shipments as they move.

What's the big deal? What makes these CLM records such explosive information? Glad you asked that question. Let's take a simple example. Suppose you are using the CLM information to follow a rail shipment from Biloxi, MS, to Minneapolis, MN. You get CLM records that show the incremental movements of the rail car away from the Gulf Coast, north through Mississippi, into Arkansas and Missouri. It dips into the western counties of Illinois and moves on to Iowa. And then you don't get any more movements for four or five days. Suddenly the rail car starts moving again and goes on to arrive at your customer in Minneapolis without another delay.

Some people might not give this slight hitch in the getalong a second thought, but because you are a dedicated and experienced traffic manager, you decide to do a little detective work. You review the history of shipments that have gone to this customer and you find that there are other cars that have experienced the "in-Iowa delay". In fact, it seems that the delay first started about four months ago. Before that, the shipment histories show smooth sailing through Iowa. So you call the carrier at the reporting location of the delay in transit. The carrier representative tells you that they have shifted some equipment away from the Iowa terminal and that you should expect all shipments through that location to experience a similar transfer time. Now you know what happened, the next question is what are you going to do about it?

Your response to this problem is what makes the CLM information so powerful and potentially explosive. Because of the CLMs, you have a history of this carrier's performance both before and after the equipment change was made at that Iowa terminal. You can positively prove that your shipments were affected by **X** days and you can show how that affects the shipment service levels agreed upon during your rate negotiations and that it changes the delivery schedules you have promised your customers. You may have to add another rail car to your fleet to cover the delay. At the very least, you may have cause to renegotiate your rates with that carrier. On the other hand, the carrier may not have realized the equipment change at that location was causing such a problem. Based on this new information you have brought to him, he may bring the needed equipment back into service.

Powerful things, these CLMs. And they are yours for the asking. So, how do you get the CLM records? The quickest and least expensive method is to use an electronic gadget called a *modem*. A modem is a device that allows your computer to connect to another computer across a standard voice telephone line. The purpose of the connection is to exchange electronic information. This process is fast and, once it has been set up properly, it's very reliable and relatively hassle-free. The term "modem" is actually a combination of two words, modulation and demodulation, which are the actions performed by the modem of converting the digital signal coming out of the computer to and from an analog signal which can shoot across a telephone line designed to handle sound (voice) transmissions. Modems are actually signal gateways controlled by *communications software* programs running in the computers at both ends of the telephone connection. Using your computer and the modem, you can call a telephone number set up by the rail carrier. This is called a *dial-in* connection.

A more modern approach is to connect to the rail carrier via the Internet. Using your Internet connection software, usually called a *browser*, you enter an Internet address (like www.thisrr.com) and click "Go". Your browser establishes an electronic link between your computer and the rail carrier's computer. The rail carrier's computer answers your computer's connection request and the communication software in both computers establishes an electronic conversation during which the rail carrier sends all of the rail movements to your computer where they are stored in a file. This is real magic and it makes anything David Copperfield does look like a bad card trick.

This electronic union (via a modem or the Internet) can connect you to most of the major rail carriers and third-party suppliers. You can refer to the **Rail Users' Manual** we described in the previous section to get the carriers' Internet addresses or dial-in telephone numbers and modem setup characteristics. Some of the carriers even offer specialized communications software packages that will allow you to simplify the connection process.

Files and Formats

The purpose of all this connection stuff is to gather the rail movement information, the CLM records, onto your computer. The CLMs are usually stored in a type of file that is commonly accessible to most computer software. There are two computer terms for this type of file: *text* and *ASCII*. These terms are used interchangeably. "ASCII" (pronounced *ask'-ee*) refers to the American Standards Codes for Information Interchange, a committee that defined the file structure. We'll just call it a text file.

A text file is sort of the Jeep of computer files. It goes anywhere. Most computer software can read and write text files as a part of their natural functions. There is a difference between the *structure* of a file and its *format*. Using the automobile metaphor we just started, the structure would define a vehicle with four wheels as opposed to three wheels or two wheels. The format would define whether we have a two-door or four-door, a stick or an automatic, a sports car or a station wagon. In the case of our CLM file, the structure tells us we are using a text file and the format tells us what information is contained in each CLM record. A *record layout* is a complete definition of each field in a file's record format showing its length and type.

The AAR provides several standard formats for CLMs, identified as CLM Text Options "Blank", "A", "B", "C", etc. If you really wanted to do such a thing, you could trace the family history of these formats back to the days of punched card-oriented computers. Some formats contain less information than others and some formats contain weird codes instead of city and state values. We are going to concentrate on the one format that seems to provide the most and best information: the CLM "D" format. This is sometimes called the "destination" CLM format because the basic sighting information is supplemented with the rail car's destination city and state.

All of the data fields shown in the CLM format in the box below are available in movement records collected through third-party CLM providers. Although the CLM supplier may not refer to their record formats using the AAR designation, the data fields will still be there. There is one very important thing for you to remember: **THE THIRD-PARTY CLM SERVICES CHARGE FOR THE CLMs**. There's nothing wrong with this, of course. It's a very American thing to do. But you have to be aware that there is a monthly fee for their services and you will have to make room for this cost item in your budget. Currently, the individual railroads do not charge for the CLMs collected directly from their systems. This may change.

AAR CLM Format (CLM "D")

The transmission from your CLM source should contain the following data found in the standard AAR CLM "D" format:

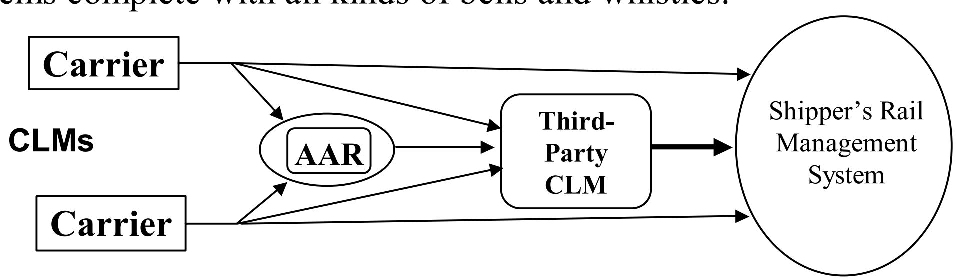
1. Rail Car ID – Vehicle Initials and Numbers
2. Sighting Date and Time – Date and Time of Movement Sighting (Look for Year-2000 compliance in the dates!)
3. Action Code – AAR Standard Movement Transaction Code
4. Location City and State – Location of Movement
5. Loaded/Empty Status – Vehicle Load Status ("L" or "E")
6. Railroad Identifier – Railroad Initials (AAR Standard)
7. Train/Junction Railroad Identifier – Train Number or Junction Railroad Identifier
8. Destination City and State – Vehicle Destination
9. Transmission Date and Time – Date and Time of CLM Transmission (in the CLM's transmission header)

! Another thing to keep in mind is that the CLMs form the backbone of any rail fleet management system you construct. Large or small, select a CLM source you can rely on, one with which you feel comfortable, and one that will be there when you need help with the collection process. Getting the CLM source connected and starting the flow of rail movement information is the first step in implementing a system. The data has to be accurate and reliable. Everything depends on the CLMs.

Sources and Solutions

As we said above, you can dial in to the individual railroads or you may prefer to go through one of the third-party solutions providers to get your CLMs. There are several different companies that specialize in providing CLMs and supporting information to shippers. You'll find a list of third parties in any of the several logistics and rail transportation periodicals. Some of these third parties can install off-the-shelf rail fleet management systems complete with all kinds of bells and whistles.

Flow of CLMs



If you don't want a packaged rail system, some of these companies would be very happy to help you create a system designed specifically for you. Whether you need a small system that simply provides record selection capabilities and a few reports or a full-blown system that gets its fingers into your existing order entry, inventory, and customer service systems, the third-party solutions providers are usually willing and often uniquely qualified to build a system for you or to suggest who can build such a system.

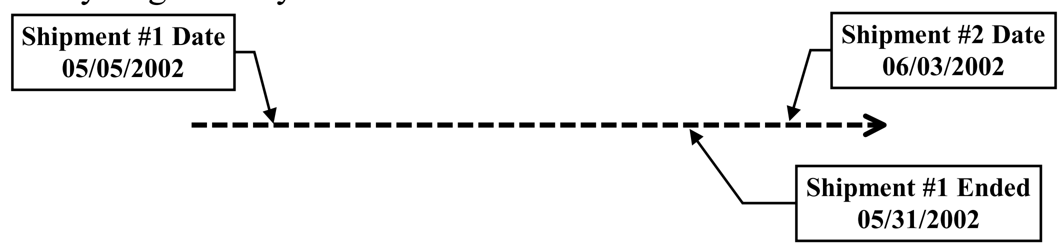
Since it's not our intention to sell packages for these vendors or to recommend any particular service or supplier, we have included at the end of this booklet a brief description of each service, a contact name, and a telephone number. Then, you can use the resources of your company to analyze the various offerings for fitness and acceptability in your situation. It is our intention, however, to describe the basics of a rail system, so let's get back to it.

Information You Provide

While the CLM information may be the backbone of a rail tracking system, it is left to you to add the brains. A series of CLMs for a rail car is nothing more than footprints that show where the car has traveled. If you add some shipping information, you will begin to see the system come to life.

Add a shipment date, a customer name, a product code or description, the origin and the destination and immediately you have a pattern and purpose related to the CLM footprints. Of these additional pieces of information, perhaps the most important is the shipment date. This date shows when the rail car started moving on its current trip.

It also indicates when the previous trip ended. Using the starting date and the ending date of the same trip, you can determine the length of the trip. Given a history of trips from a specific origin to a specific destination, you can determine the average performance of your rail cars. With some very light statistical calculations, you can determine the expected trip times with a high degree of accuracy. From that you can develop estimated times of arrival from any origin to any destination.



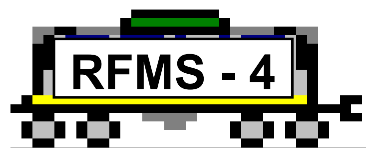
Assuming you now know the start and end dates of your trips, you can determine from the CLMs the middle points of each trip, that is, when each rail car arrived at its destination and when it was released for its return to origin. You can easily determine this using the load status (remember the **L** or **E**) of the car. And if you know when it arrived at and departed from your customers' locations, you can calculate the delay at each customer. For example, assuming that your car ships out loaded, the time differential between the last loaded CLM and the first empty CLM is the approximate delay at the customer. Again, using a little statistical magic, you could begin to predict how long each customer delays your cars and you could make some very intelligent decisions concerning shipment costs and customer priorities.

Remember, earlier we mentioned that shortly after you begin using a rail tracking system, the traffic people start sprouting ideas like flowers in springtime. You will hear exclamations like this: "Hey! Look how long it takes to get cars through Bracketville!" or "Wow. I had no idea that Billabong Industries took eight days to unload our cars!" We refer to this as the "Hey–Wow!" syndrome. It comes from having seen these kinds of reactions to the information produced by even the most basic tracking systems. Every transportation manager knows about shipment begin and end dates and customer delays and how to manipulate the numbers, it's just that it suddenly becomes easy to see and do. When something is easy, it tends to get done. And soon, the information just rolls out of the system.

While it is easy to do, it does take time to set it all up. The information age doesn't come cheap. However, if your traffic department is currently logging every trip in a notebook or even in a spreadsheet, it will not take any more time to enter the same information into a computerized rail management system. On the other hand, once you get started, you will probably find more things you can do or want to do. More calculations, more estimates, more reports, more information, more selections. All of this means MORE WORK.

Somewhere along the way, you will have to draw a line between what is necessary and what is desired, between what produces better information and what produces more work.

We call this the WORK VS REWARD dilemma. Humans have been struggling with this dichotomy since we first fell out of the trees.



4. System Requirements

We are at a crossroads now, so let's take a quick review. We have given you a taste of what you can do with a rail management system, you've probably had a few meetings so corporate is interested, you have introduced your logistics design team composed of both transportation and computer folks, you've had the blue-sky meeting and everyone was surprised at the interest in and need for the information. You know a lot more about the CLM sighting process, CLM formats, and shipping information than you did before. You may have called a few carriers or third parties about the mechanics and costs involved in collecting CLMs. A few of you have even checked out the telephone or Internet connection equipment available in your company. (If you did do this, it's time to get out your "Technical Wizard" button and hang it on your shirt or blouse. Congratulations!) We're getting close now, Jim, you can feel the excitement in the air!

In this chapter, we are going to introduce "system requirements". This is a generic term we will apply to everything connected with the construction of a computer system. First, we will discuss the computer hardware: video screens, disk drives, memory, printers, networks, all the good stuff. Then, we will briefly step in some software. Most companies have established standards, or at least vaguely formed directions, on this subject so we have to tread lightly here. Next we will talk about data base techniques and define a few terms so we are all talking the same language. Last, we will offer some guidelines for designing the system.

If you are a computer person, this is the part where you get to relax a little bit. The previous chapters probably stretched your feeble brain cells with all that talk about CLMs, trip times, and loaded/empty status codes. If you are a rail person, you had it easy in the last chapter, and now you should prepare to be brain-stretched!

Hardware Basics

We are going to get the hardware recommendations out of the way first. This is a difficult area in which to generalize. There are hundreds of possible choices you can make because every company has a different philosophical approach to information technology and that means every company has different computer equipment requirements. So... look around, see what you've got, and prepare to use it.

You probably thought it was going to be more sophisticated and difficult, but it's not. We could give you a long list of hardware items starting with "get the latest and fastest computer system available", but you would probably ignore our recommendations and use your existing computers anyway.

If you are building the rail management system on anything larger than a personal computer, your company probably has formal guidelines and standards for the computer equipment in its user departments, so you can skip this section. If you don't know what kind of computer hardware equipment you have, take this little two-question test:

1. Is your computer equipment located somewhere other than on or under your desk? YES or NO
2. Does the person who runs your computer equipment have the word "Operator" in his or her title? YES or NO

If you answered "YES" to either of these questions, the rest of this section probably does not apply to you and you should skip to the next section in this chapter. If you choose to continue reading this section, you do so at your own risk. You could easily get sucked into the technical vortex from which there is no return.

If you don't already have computer hardware, or if you are starting this whole project from scratch and you really do need some direction as to what equipment to get, well, have we got some advice for you! We will lay it out simply. Buy quality.

You can see that we are trying to avoid giving you specific advice on the selection of computer equipment. With all of the different computer environment options available today, it is nearly impossible to do such a thing. You could be using IBM-compatible personal computers, Apples, UNIX, local networks, wide-area networks, remote workstations, or remote connections. It's just impossible to put generalized, usable recommendations for every situation in a booklet like this. However, so you will not feel like we have short-changed you, we will offer this brief checklist. Just call them suggestions based on experience.

- **Use a high resolution video monitor.** Most video equipment purchased new in the last year or so has clean, crisp, clear character displays. You want to avoid blurry characters or unfocused display screens. Color is highly recommended. But, it is better to have an easy-to-view black-and-white monitor than a faded, fuzzy color display. Traffic people have enough headaches without getting them from their equipment.

- **Use disk drives that provide <u>fast</u> data access.** Any rail-oriented computer system is going to be using a lot of disk space and the level of disk activity will be very high. As far as speed is concerned, get a disk drive with the fastest access time money can buy. As for capacity, get a disk drive that is at least two times larger than you think will be satisfactory. To make a rough estimate of the disk space needed, multiply the number of rail cars you will be tracking in the system times 100 (the approximate number of characters stored in each CLM record) times 25 (the average number of CLMs received per car per month) times 12 (twelve months in a year) times **Y** (the number of years of rail data you need to keep immediately available). This will give you the base amount of disk space (in characters) that will be used as data storage. We opened up this booklet with an example using a fleet size of 150 cars, so let's do the math using that number and we'll say we want to keep three years of active rail data. Remember, this is just the active rail data. You will need additional disk space for programs, reports, selection files, support files, and other things that go bump on the disk. When you have calculated the absolute highest amount of disk space you will need, multiply it by five.

$$\underset{\#Cars}{150} \times \underset{\#Chars}{100} \times \underset{\#CLMs}{25} \times \underset{\#Mos}{12} \times \underset{\#Yrs}{3} = \underset{\#Chars\ in\ CLM\ History}{13{,}500{,}000}$$

- **Use the fastest computer you've got.** We said earlier that rail logistics is a management by exception activity. When the exceptions hit the fan you need answers, ***NOW!*** You might be in the middle of writing a memo to your customer services manager when you get a call about a hazardous material spill in Big Swampy, Virginia. You need a list of rail cars in that area, the products they are carrying, and the customers to which they are going. And, you need it ***NOW!*** So, you want immediate access to the selection features of your computer system. The last thing you want or need is a computer that takes several minutes to start a program and even longer to search your CLM files. A good rule of thumb used to determine if your system is too slow is: if, on a quiet, easy day you find that you are tapping your pencil on your keyboard while you wait for your system to start, then your system is too slow. If this happens to you, your system will not be able to react quickly enough to help you when you really have problems. And, for a rail manager, anything that doesn't help you just gets in your way.

- **Use a laser printer connected locally.** This may seem ridiculous at first, but it fits right into the "speed" theme. Right now, laser printers offer the best print quality and the fastest print speed available to computer users. If you need to print out a list of all the empty cars available for loading to fax to your production manager ***NOW!, y***ou don't want to magnify the fuzziness of a dot matrix printer with the fuzziness of a fax, you don't want to have to decrypt the random blobs of an ink-jet printer,, and you certainly don't want to wait on a slow printer. If you do have a laser printer, but it is the department's group printer to which you and five other people are connected, Murphy's Law states that you will need your list of available empties immediately after your boss's secretary begins printing an update to the corporate policies manual.

- **Fast, faster...Speed, more speed.** Have we emphasized this enough? The computer people are nodding their heads in agreement because they have heard about all they want, but the rail people are shaking their heads, saying "No, give us more...give us more!" Traffic management is a dynamic, reactive business. Traffic folks need tools that help them get their jobs done as quickly as possible. If a tool slows them down, sooner or later they will quit using it. A rail system is a terrible thing to waste, not to mention the money, time, and resources that went into its development. So, it has to help them, and to do that it has to be fast and effective. If they need more memory in their computer to make it run faster, get it. If they are connected to a network and the network is slow, consider taking them off the network and connecting them to an outside rail management service or, at least, building a new set of tools that can be used locally. Do whatever it takes to make sure that the job gets done. It is important to the rest of the company, too.

If you are a personal computer user, we can make the following minimum hardware recommendations. If you connect to the Internet directly or through a network, you will need to add any additional connection hardware.

1. **Computer** - A really fast CPU running the most stable operating system with just a ton of memory.
2. **Disk Drive** – The biggest hard drive your computer will support.
3. **Video Monitor** – The biggest high-resolution screen you can afford.
4. **Backup Unit** - Internal or external does not matter, just get one.
5. **Modem** - Internal or external fax/modem with data compression and transfer speeds of 56.6 Kb or greater. You are better off with a cable modem, DSL or ISDN line, or other direct high-speed Internet connection device.
6. **UPS** - Uninterruptible power supply sufficient for powering the PC and video monitor for 30-60 minutes in standalone mode.
7. **Printer** - High quality laser printer with a high page-per-minute rate.

Software Strategies

Take two aspirin and call us in the morning. Software choices always cause headaches. The choices you make depend on the four goals you set for your system: speed, capability, ease of use, and cost. There's just no easy way to generalize these things. If you already have your software selected and your design steps planned, go ahead and skip to the next section. If you need some direction, read on.

You will probably need three different types of software: a database manager, a development language for the basic system, and a database manipulation tool for special data selection and reporting. Sometimes the three types can be found in one package, but more often than not, it will take separate tools. The database manager controls access to the rail information. The development language is used to build a compact, efficient system that let's us manage the day-in/day-out processing and solves 90% of the users' information problems. The database manipulation tool is used for one-time jobs and "gotta-have-it-now" solutions.

Most companies have some very definite rules and regulations regarding the software that gets used on the departmental computer systems. Software decisions are based not only on the hardware platform on which the programs will be run, but also on the technical level of the information department's staff, their commitment to supporting the user departments, the corporate budget, and a host of other factors.

As a part of the software choice, there are certain qualities or features that you should look for when you design your rail system. You will want to use a database manipulation tool that the users can learn and use quickly. No matter how perfect a system is and no matter what functions you have included, there will always be a need to get at the information in a way that was not part of the original design. It's just the nature of the beast. The last thing a computer programmer wants to do is run back and forth to the transportation department creating little one-time add-ons. Likewise, the last thing a rail manager wants is to wait on some programmer to finish a little gotta-have-it-now job. So, your users should have the ability to *query* or *filter* (perform selective data access on) the CLM data and create their own reports.

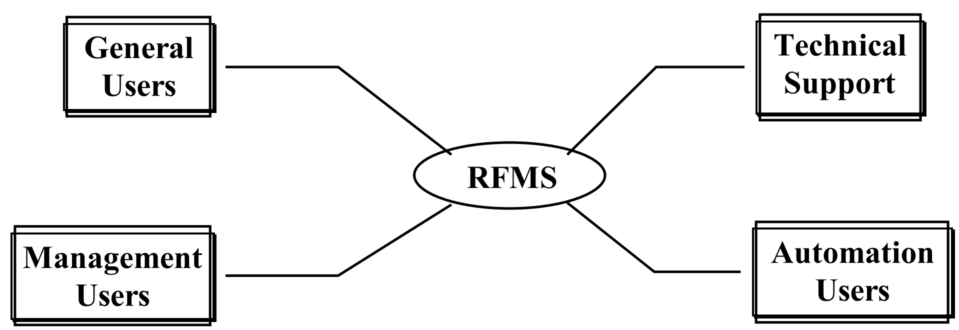
The hardware section above emphasized speed and that has to be a large part of the software equation. When the alligators are snapping their teeth, no one wants to sit and wait while their system slowly loads into the computer and grinds through megabytes of data. Therefore, the basic rail system should be as fast as possible. A little slowness in the special one-time reports is acceptable, but the main system and the gotta-have-it-now jobs must burn up the wires with their speed.

However, speed is a relative thing. It's a fact that Ferraris go faster than Jeeps on the Autobahn, but Jeeps can traverse those rocky back roads better than Ferraris. You have to choose the right tool for the job. You need to apply that same wisdom to your choice of software. The term *platform* refers to a specific combination of computer hardware and software, and some computer programs run faster than others on a given platform. Platforms run the gamut from personal computers up to mainframe computers. A database management program which runs on a PC will have a different set of functional characteristics and capabilities when compared to a database program running on higher-level computer platforms. You have to choose the right software to go with your hardware or you have to get new hardware to match your choice of software.

One last software requirement before we move on: the basic system must be written using software that is expandable. The basic tracking system we are describing here is not a "throw-away" project. It is designed to be the bottom block of an upside-down pyramid on which a full rail fleet management system can be built. The reason you have not heard the word "prototype" used in this booklet (and you will not hear it again) is because it is rare that such a thing really exists and even more rare that corporations allow them to be constructed, other than to define a new user interface. Most programmers already know what language they are going to use, how to control the database structure, and how they are going to display information on the screen. They know this because they have already used the language, the database, and the interface in other programs or they have IS (Information Systems) guidelines which specify what to use, what to do, and how to do it. The programs they write for the basic tracking system will become the *initial functions* of the full system when the transportation department decides to expand the system to include more features. This means the original development language, the database manipulation software, and your RFMS programs must be able to handle added processes, screens, and reports. Your programmers will probably never get a second chance to create the basic system.

What the System Must Do

If you were floating around out there somewhere in the space shuttle and you had the opportunity to look down at this rail fleet management computer system we are trying to create, it might look something like this:



From our lofty perspective, we can see that our system is going to be used by at least four different groups. "Used" is perhaps the wrong term because only the **General Users** are actually going to use it. "Viewed" is more appropriate, or "dealt with", or "looked at". Whatever.

- The **General Users** get into the system, print the reports, look up the rail cars and the shipments, and query the data for specific information.
- The **Technical Support** folks administer to the needs of the database and programs, tuning it, tweaking it, making changes, and performing backups and fixes as required.
- The **Automation Users** are not even human. They are the special computer-controlled processes required to collect the CLMs, feed the input to the system update programs, and run the backups and archive processes.
- The **Management Users** do not actually operate the system, but they read the reports and react to the information generated by the system by making decisions.

These are four very different perspectives of the rail system, however, there are certain key elements they have in common. All of the users expect the system to:

1. Engender a high level of confidence in both the service and the data reliability. The system must provide all users with fast response times and accurate information. If a **Management User** needs a report of loaded cars placed at destination every morning at 8:30 AM, then that report has to be there and it has to be correct, consistently.

2. Be simple to use. The interface should be as simple as possible. For example, pressing the "Print" button allows the user to print from a list of reports.
3. Be fast. For all users, the system must start quickly, the functions must be easily and immediately accessible (without long menu ladders to climb up and down), the selections, data generations, and reports must produce results within an appropriate time frame, and the users must be able to exit the system or change to another application quickly and without fuss.
4. Focus on a function quickly. The users must be able to change from one process to another without losing the original train of thought. Also, when the focus changes, the intuitive results should be displayed. For example, it you are looking at the most recently received CLM for rail car "ABCD000123" and you change focus to the shipment information, you should be looking at the shipments for the same rail car.
5. Be flexible. The system must be able to adjust to new requests for information or respond to new selection criteria without undue change or challenge.
6. Provide different views. A user should be able to press a key or click on a button to see all the CLMs, press or click again to see only the CLMs for a specific rail car, or press another key or click another button to see the shipment information for the same rail car.

7. To work with existing computer hardware, networks, and operating systems. In other words, the rail system should have a user interface that is consistent with other applications running in the same environment so that neither the **General Users** nor **Technical Support** has to learn new methods unique to this project. This is a tall order when you realize that computer hardware and software changes almost completely every eighteen months. You know that mouse you've been using for several years? Get ready to take it off your desk and begin talking directly to your computer. Uh-hunh. It's coming.

The list above reflects the requirements from a general perspective of all four user groups. The following list focuses more on the technical side. The programs in the system should be designed to:

1. Satisfy the users' needs. Otherwise, what's the point of building the system. However, the keyword here is "needs". There may be a whole stack of "wants" out there, but your logistics design team will have to decide what to include and what to leave out.
2. Operate within the users' skill levels. You don't want the rail system to be so basic that the quick-learners are bogged down in menus and option selection lists. On the other hand, you don't want new users to have to learn such esoteric keystroke combinations as **Ctrl-Alt-Genuflect** just to access rarely used

functions. There is a middle ground offered by the newer graphic user interfaces (GUIs). Use the technology.

3. Be usable in both your current and future computer environment. Hey, nobody said this stuff was going to be easy. You just have to try to do the best you can. For example, ten to fifteen years ago who would have guessed that DBase would disappear from the A-list of personal computer database managers, or that IBM would actually provide a very powerful database manager capable of competing with Microsoft. How do *you* pronounce "SQL"?
4. Be usable on your corporate network. Some things are easier than others. However, unless your computer network is left over from the '60's, you can get most software to work well with most networks right out of the box.
5. Fit reasonable cost/time specifications. The trick is to do a really good job on the design; the rest will fall into place naturally. Stay within your corporate knowledge and skill capabilities on the design specifications and then stay within your design specs as you build the system.

You probably noticed that there was some overlap between the two requirements lists. You'll find that nature is full of these interesting overlaps and it enforces the natural order of things by highlighting those parts of the system that require concentrated attention. Simplicity, flexibility, efficiency, and speed are the key ingredients of any rail system.

What the System Must NOT Do

Sometimes in that special frenzy that overtakes system designers, we forget that there is an opposite side to the list of features included in a computer application. In the on-going effort to include everything a user says he wants, the resulting program can become an inefficient, plodding maze of "really neat things" that never get used. What starts out as a quick-in/quick-out solution provider turns into a bottomless morass of pop-up and drop-down menus, dialog boxes, finger-wrenching keystrokes, and answers to questions that no one will ever ask.

Here is a list of what NOT to include in your system:

- **Complex or fancy display screens** - Keep the displays focused on the subject. Most users know what they want when they sit down at the keyboard and start using the system. If they get caught up in screens that look like most Internet sites or the cover of one of those magazines you see when you are standing in line at the grocery store, they forget what they were doing or they quit caring. It's easy to get lost in screens that look like ransom notes or topographic maps. Also, "rainbow" displays on color monitors may show off the capabilities of the video monitor, but they just sidetrack or confuse the user with sensory input they don't need.

- **Intricate and elaborate menus** - Go directly into the primary system function when the system is loaded. Better yet, let each user identify the function that should appear on their screen at startup time and save it in that user's "characteristics" list. Use the function keys, buttons, or a toolbar to provide direct access to highly used system features. Avoid selection lists that display selection lists that display selection lists... blah... blah... blah.
- **Heavy system overhead** - Systems that are burdened with tons of features, dozens of database tables, and piles of graphics often put a demand on the resources of a computer to such a degree that nothing else can be done until the system is terminated. Lighten up! If necessary, create several different system entry points and limit the features to those related to that entry point's subject. Also, avoid leaping into a function which processes for an extended period of time without providing an on-screen description of the process and an exit option. Users should be able to select YES or NO before they send their computer into a two-hour information generation process.
- **Operator (user) overhead** - Repetitive functions dull the human senses. Provide as much automation as possible for the daily collection update and report processes. Also, if you find that you are doing any function the same way every time, automate it. For example, it's ridiculous to require an operator to press the "Yes" button to start the monthly archive process. Instead, have the computer

recognize that today is the first Saturday of the month and execute the archive function automatically.

Go or No-Go

After all that we have talked about so far, it's time to decide if you are ready to begin the construction process. Do you have the blessings of management? Do you have the support and interest of your user community? Do you have the attention and dedication from your information systems department? Do you have the equipment? Do you have the time? Do you have the money? Do you have the skills? Do you remember the name of Red Ryder's Indian companion?

These questions are not raised to denigrate the quality or expertise of your transportation or computer personnel. Not at all! It's just that you need to weigh these factors before you take on the construction of a computerized rail system. You don't want to spend a lot of time and money only to reach the limits of your company's human, financial, or technical resources half way through the project. If that happens, everybody loses.

The basic rail tracking system described here is intended to be a short project of about four months or less (after the design is completed). The only difference between long projects and short projects is the level of corporate commitment. A short project can (and often is) just as intense as a long project. The intensity just doesn't last as long. Everything else is the same. Management has to be committed to the project, the information systems group has to be committed, and the user departments have to be committed. And the commitment level has to be 100% or the project will never survive. If management does not give its total approval, the project will lack the needed pressure from above and it will never be completed. If the computer people are not *gung ho*, the system will be a dud or it will never be refined to usable perfection. If the users don't care, why start the project in the first place?

(His name was Little Beaver. And if you knew that, you're getting way old.)

Terms of Endearment

Before we go on, we must first learn how to talk to each other. Specifically, it's time to define a few good computer terms. This will make it easier to describe our system. If you have ever used a computer, you have heard these terms and most of them have a common or business analog to which you can relate.

For example, everybody has a concept in mind when you use the word *file*. Business people immediately think of a filing cabinet holding a bunch of manila folders grouped in such a ways as to aid in the selection of specific information. That means one folder may contain employee information, another folder contains customer information, and yet another folder contains product information. Computer people immediately think of files containing the same segregated groups of information, but the holding device is not a folder in a filing cabinet. Instead, it is a disk drive connected to a computer. Regardless, the concepts are similar enough that no one questions the use of the term.

For our purposes here, a file will be considered the same thing as a *data file*, a *database*, and a *table*. If we were picking nits, there is a difference, but we aren't so there isn't. Got that? We will treat them as synonymous terms and we will try to limit ourselves to using the two database-oriented terms *database* and *table*, since we are describing a database project. The term *database* can be used to refer to all of the tables in our system as a collected set. Not surprisingly, this is also known as a *table set*.

In our rail tracking system, a table is stored on the computer's disk drive and is composed of one or more *records*. A record is composed of *fields*, and fields are composed of *characters*. For example, in the file room (a disk drive) you will find a file cabinet (a database) with a drawer (a table) labeled "Customers" (CUSTOMER.DB). In that file drawer there is a folder (a record) for each customer which contains name, address, contacts, phone and fax numbers (these are fields) which are made up of letters, numbers, and symbols (collectively called characters). In the rail system, we will have many tables in our database, but our main table will be the CLM table. This table is composed of a one unique record for each movement of each rail car. The CLM records contain fields that describe an individual movement: rail car initials and number, sighting date and time, status codes, location city and state, and the railroad identifier.

There will be several tables in our system and they will contain related but separate information. You have probably heard the term *relational database*. It is used to describe the tables in a database whose records have one or more data fields in common. In other words, they are related because of this common field. For example, in our CLM table each record has the rail car's initials and number so that we can keep all of the movements for a specific car together. We will also have a shipments table. The records in this table will contain the rail car initials and number, the shipment date, the origin, destination, customer's name, and product among other fields. Since we have the rail car initials and number in the records of each table, we can relate a specific car's shipment to its movements for that shipment.

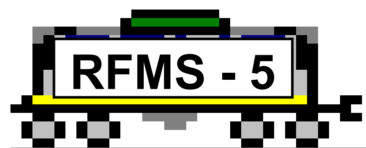
All of the tables in our system will be structured in such a way that every record is unique. We do this so that we can quickly go to an individual record. If we have a table called "RailCars", for example, where we have one record for each rail car we lease or own, we use the rail cars' initials and number field to make each record unique. To make the records in the CLM table unique requires three fields: the rail car initials and number, the sighting date, and the sighting time. Look at the table on the following page.

This table shows a series of nine records for three different rail cars. Notice that as many as two fields may be the same in one or more records, but never will all three fields be the same. The fields that create the uniqueness are called *key fields*. These key fields are, by design, the first three fields in each record. Most, but not all, relational database programs require this.

Rec#	RailCarID	Sighting Date	Sighting Time
1	DDCX 001234	05/09/2002	04:43
2	DDCX 001234	05/10/2002	18:15
3	DDCX 001234	05/15/2002	18:15
4	DDCX 001234	05/15/2002	18:30
5	DDCX 001235	05/10/2002	12:00
6	DDCX 001235	05/12/2002	12:00
7	DDCX 001235	05/14/2002	02:15
8	DDCX 001235	05/15/2002	21:10
9	DDCX 001236	05/15/2002	18:30

The key fields are also stored in a separate file called an *index*. The index file contains each record number used to locate a specific combination of rail car, date, and time. The database index is used by our database programs to look up an individual record just like humans look up a word in the dictionary or a name in the phone book. For example, most telephone books use *LastName* and *FirstName* as the two key fields. When you look up **Joanna Smith**'s telephone number you first find the page containing **Smith**, then you scan the names looking for **Joanna**. Our rail tracking programs will do the same thing except that it will take three steps to find an individual record in our CLM movement table. Step one locates the rail car, step two selects the date, and step three finds the time. Using our table above, if we looked for the rail car "DDCX 001235", the sighting date "05/15/2002", and the sighting time "21:10", the program would display or print the movement information contained in record number **8**.

We will explore the various relationships between the tables in our database later. For now, it's important that you understand the concepts we have just described. If there is any confusion, talk openly with your logistics design team members. The computer people will be able to give you some other examples that will help to firm up the shaky ground. The rail people should also define any specific terms we have not covered. Make sure that everybody understands all the terms before you continue.



5. Building the Basic Rail Tracking System

Everybody is ready now. Management is in their offices nodding their heads in total approval. The users are out there at their workstations staring at the screen waiting for rail information to pop up. The programmers are sitting at their computers with skilled fingers poised over their keyboards, anxious to carve out some executables. The logistics design team members are lurking in the shadows with copies of the design plan in their hands waiting to be asked a question about the architecture.

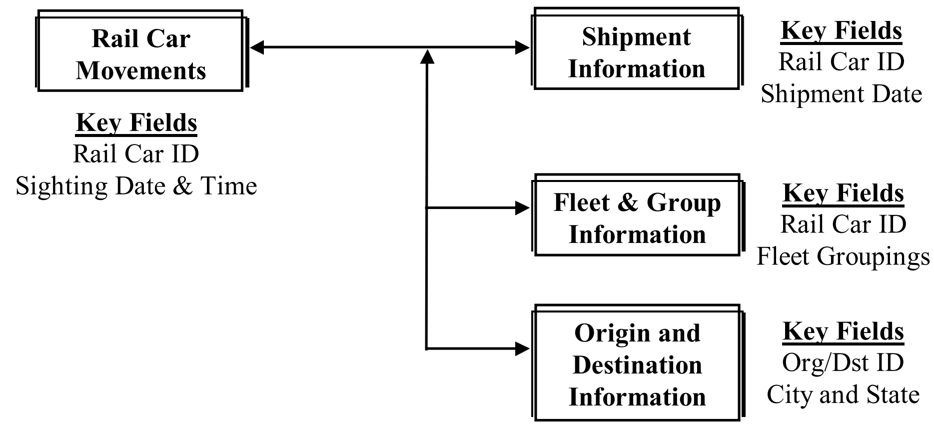
On your marks.... Get set....

Wait a minute! Design plan? What design plan? We don't have a design plan. Oh, no! We've got more work to do.

In this chapter we will lay the pieces out in front of you so you will know how to put them together. If you have asked all the right questions around your company, you will have all the right answers so this will simply be a matter of putting pen to paper. We want to create a design document that will completely describe our new rail tracking system. This will be the blueprint from which the programmers will crank out the code that will tell the computer how to collect and manipulate the data that will form the reports and screens that the transportation department will use to produce the information that the company needs. And, the leg bone is connected to the knee bone.

The Basic RFMS Model

It's time to actually do something constructive. We have defined and described the separate pieces of this puzzle and now we start putting the pieces in place. The diagram below is an overview of the database elements of our system. This type of diagram is referred to as a *model*.



The model shows the minimum table set needed to create a functional rail fleet management system. These four tables will enable you to create a powerful daily tracking system and also construct a solid foundation for a full-blown fleet management system. Let's spend some time on the details so you can see how this database structure can be used almost immediately. Then, in the following chapter, we will give you some pointers on how to extend this system to produce the trip statistics that will help you measure your fleet's efficiency, forecast fleet utilization, and trim costs appropriately.

The primary table in any rail system is the Rail Car Movements table. This table contains one record for every move your rail cars make. The data used to update this table is the Car Location Messages (CLMs) that you will collect from either the rail carriers or a third-party CLM source. We are basing our system on the AAR (Association of American Railroads) standard CLM "D" format. As we said in a previous chapter, this is sometimes referred to as the "destination" CLM format, so called because the basic sighting information is supplemented with the rail car's destination city and state. The mechanics of the communications process is left to your computer gurus. You may call as many CLM sources as needed to cover your fleet of cars, but we recommend that you collect all of the CLM records into a single file. Then copy the CLMs into a separate archive file just in case you need to rerun the update process.

This CLM data is stored in the Rail Car Movements table with only a slight amount of reformatting necessary to allow the first three fields to be the "Rail Car ID", "Sighting Date", and "Sighting Time". These fields will be our *key fields* and they will be used to generate an index to each record in the movements table.

Getting the CLM data into the Rail Car Movements table will require some form of program control. This is called an *update* program. We'll leave the programming language up to your computer folks, but generally, you are looking for the fastest update process possible. It should also do some basic data validation. The following fields should be checked. Any invalid data should be logged on an exception report.

- **Rail Car ID** - Each rail car's initial and number are combined into a 10-character field. The first four characters must be alpha characters (upper case letters or spaces). The letters must be left justified with no spaces between the letters. If there are less than four initials, spaces must be used as padding characters on the right to fill to four characters. The remaining six characters must be numeric characters (numbers 0 – 9). The numbers must be right justified. If there are less than six numbers, zeros must be used as padding characters on the left to fill to six characters.

- **Sighting Date** - The date of the sighting must be in a comfortable (for the users) format using numbers for the month, day, and year. CLMs with dates in the future should be considered invalid and ignored. The one possible exception to this rule is if you decide to use the ETA CLMs (Action Code = 3) transmitted by some railroads. We recommend that you log the ETA CLMs on the exception report, but do not add them to the Rail Car Movements table. To do so will cause your system to get confused and occasionally spit out data hairballs like a Persian cat. Oh, and of course the date must be kept in a Year-2000 compatible format.

- **Sighting Time** - The time of the sighting, like the date, should be saved into a format that is easily read by both the computer and the human users. In our examples, we will use the standard military time display format (HH:MM). Most computer software provides a *time stamp* format combining the date and time into a data field that can be easily validated and manipulated by the computer's date routines.

- **Action Code** - The movement action code is a one- to three-character code that should be validated against the list of possible codes provided by the NITL's **Rail Users' Manual**. Action codes with less than three characters are left justified and padded to the right with spaces to fill the three positions. CLMs with invalid codes should be logged on the exception report but not added to the movements table. If the action code is invalid, the CLM is nothing more than electronic flotsam.

- **Loaded/Empty Status** - This one is easy. It is a one-character code and if it's not **L**, it had better be **E**. Anything else is junk. In fact, if any one or more of the five fields described so far contains invalid data, log it as an exception and throw it away. You don't want it near your good movements. You may want to check with your CLM source and let them know if you receive bad data very often.

- **Railroad Initials** - The railroad transmitting the movement record is identified by a unique set of initials. This is a one- to four-character field. Initial sets with less than four characters are left justified and padded to the right with spaces to fill the four positions. The initials can be validated using the list of reporting railroads found in the **Rail Users' Manual**. CLMs with invalid fields should be logged but it is neither necessary nor desirable to throw away the CLM. First of all, the railroads put bad initials in their CLMs about as often as Ed Jones misspells his own name. Second, if you log the exception, you can do a little extrapolation from other similar trips and correct the initials yourself. Notify the railroad if you get bad initials very often.

- **Location and Destination** - Both location and destination are two-field combinations: the city field and the state field. The spelling of the city is covered by standard conventions. The **Official Railway Guide** will keep you up-to-date on the city names. However, you will find that the same carrier will spell a given city's name in sixteen different ways. One time it will be "ESTLOUIS", then next time it will be "EASTSTLOU", and the time after that it will be "EASSAILOU". Wait till you see what they can do with "APALACHICOLA", "OPELIKA", and "MASSAPONAX", not to mention "BLENNERHASSET" or "HAVRE DE GRACE".

> We have included a description of an optional function that will rename misspelled city names as they enter your rail tracking system. See the **City & State Translation** section in the chapter on **Extending the System**. This is a handy process, but it is, by no means, required for the successful implementation of the system.

As we have said before, the Rail Car Movements table is the heart of any rail tracking system. You could stop here and still be able to help the transportation group do their work better. But we're on a roll now, let's don't stop. There are three other tables we need to add to our system: the Shipment Information table, the Fleet Information table, and the Origin/Destination Information table.

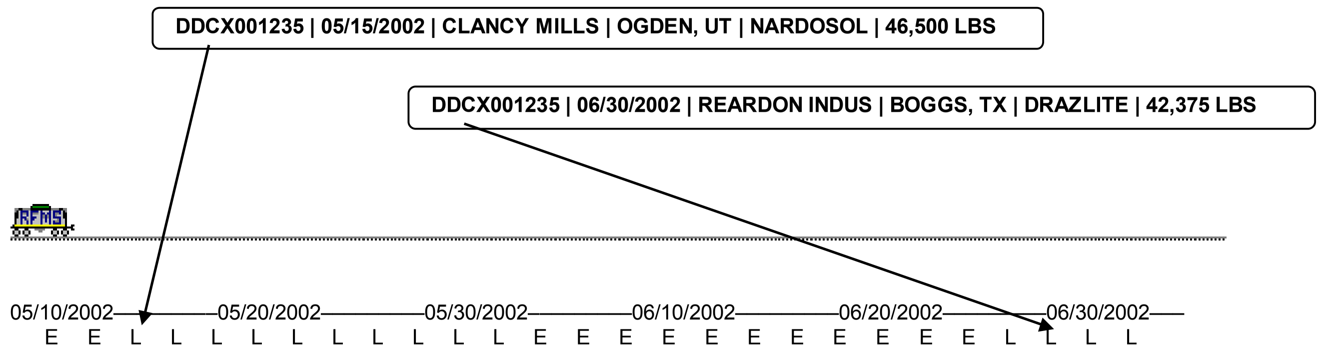
Shipment Information

If we take the Rail Car Movements table and add something as simple as the shipment date to it, we've suddenly got a whole new ball game. Whereas before we just had a stream of empty movements for each car followed by a stream of loaded movements, now we have a time element that tells us when and where each shipment started.

Why is this important? Because using that shipment date as a time marker we can spot the first movement record that was actually a part of a specific shipment. Looking at this movement in combination with the shipment info, the Loaded/Empty status tells us if the rail car is going out loaded with product or returning empty. Knowing the sighting location city/state of that first move gives us the origin point, and the destination city/state tells us where it's going. If we see the movement records change from loaded to empty, we know where (the sighting location on this movement should be the same as the destination from our first movement) and when the product was delivered. More importantly, we know that <u>it was delivered</u> because now we have an empty car coming back to us.

If you create a Shipment Information table and store in it such data values as the customer's name, the shipment number, the product code or description, the volume or weight of the product, a waybill number, a route, and some notes (just in case you need to remember that there was something different about this shipment), well, now we're really cooking. We will add some very important fields to this table when we extend the system, but this will do for now.

The key fields in the Shipment Information table are the rail car initials and number and the shipment date. These two fields make each shipment record unique because no rail car can ship out more than once on a given day. The shipment records are directly related to the movement records by using these fields. Take a look at the time chart below to see how this works.

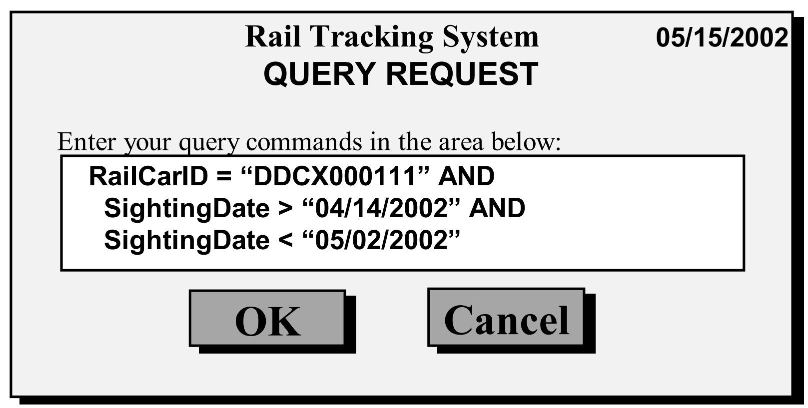


Fleet Information

The addition of the shipment information is the one thing that changes our little project from just another database containing movement records into a real information system. But, you will find that not everyone in your company wants to see all the information about the rail cars on every report or screen view. In fact, in some cases, you may want to limit a user's view of the system to only a few rail cars or only a small subset of the CLMs or the shipment information.

The general structure of the system should allow you to ask and answer a question like "I need to see all of the movements between April 15th and May 1st for the rail car DDCX000111." A program would handle this request through a direct lookup on the rail car's initials and number along with the sighting dates using a *dynamic structured query*. The "dynamic" part of this term means that the request for information is built directly from the user's input at the time it is needed. The "structured" part means that the program controls the search using a standard, repeatable operation. Your program might have a "button" you can press which causes a query dialog box similar to the one shown below to appear on your screen.

Notice that the "query command area" contains the field names and request values that will answer our question. The result of this request will be produced on the screen when the user presses the **OK** "button. You should use the same query command structure, if not the same dialog box, whether you are printing the resulting information or displaying it on the screen.



There are many other questions that might be asked and many other ways to group the resulting data. Even though we are moving toward the "paper-less office", we are not there yet and in some cases we may never get there. After all, the human eye will still focus on a list of items on the printed page quicker than when the same list appears on a display screen. Perhaps it is because there are no command buttons or menu options to distract the brain on a piece of paper. For that reason, you will want to distribute a certain amount of information in printed form to the users of the rail data. And, since not everybody wants to see ALL the rail cars ALL the time, we will use the Fleet Information table to help us group the rail car sighting information into predetermined sets.

For example, the regional customer support people will probably want to know the current location of each rail car moving to the customers in their region, but they are not interested in empty cars at the plants. The production people at each plant will want to know how many rail cars are currently available for loading at their plant, but they don't care about bad ordered cars in transit. The shipping people will want to know how many cars have not yet been loaded and how many empties will become available for loading by Thursday afternoon, but they don't care about the cars placed at destination.

The Fleet Information table contains several fields that will help us do this selective reporting quickly by using the relational capabilities of our database structure. The Fleet Information table uses the rail car initials and number as its key field, therefore we have only one record for each rail car in our system. In addition to the rail car identifier, the table has:

1. An **Active Code** field which contains the active status of the rail car. You might want to use the following codes: "A" for active, "I" for inactive, "S" for a shopped car, and "P" for a car that is held pending an inspection or and on-site repair or modification.
2. A **Fleet Identifier** field which identifies by code or text the group designation of the car. Some companies group their cars by origin, some by product service, some by division, and some by car type (tank, hopper, etc.).
3. An **Ownership** field that contains a code value showing that the car is leased, owned, assigned, or temporary.
4. Two service date fields: one showing the date the car was **Placed In Service** and the other showing the date the car was **Removed From Service**.
5. A **Group Identifier** field that can be used for dynamic grouping on the daily reports. This is explained in more detail below.

The fields described above will allow you to create reports containing both movement and shipping information or any subset you can think of. You can add more fields if you need them to create semi-permanent groupings.

Of all the fields in the Fleet Information table, the most interesting and most versatile is the Group Identifier field. This is the field that allows you to generate reports and displays using on-the-fly groups (i.e., categories that you create as you need them).

What you need to do first is create two or three general reports which are sorted on the Fleet Information table's Group Identifier field. Most report generators don't care what data values are in a group field when you create the report, so we are going to take advantage of that fact. We'll create a report that looks something like the one shown on the next page.

Using this report structure, it doesn't matter what values you place into the Group Identifier field. The report groups will automatically change depending on those values. The basic report will always look the same, but the rail cars will be grouped differently each time you alter the value in the Group Identifier field for each rail car. The report uses the relational capabilities of the database to connect the RailCarID field in the Fleet Information table to the RailCarID field in the Rail Car Movements table and pick up the movement information.

Now, to make this thing work the way we want it to, you will need a special program to spin through the movement records and set the Group Identifier value into each record in

Rail Tracking System 05/15/2002
YOURGROUP REPORT

XXX-Your Group Identifier-XXX

RailCarID	SightingDate&Time	Actn	L/E	RR	Location	Destination	ShipDate
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy
xxxxxxxxxx	mm/dd/yyyy (hh:mm)	xxx	x	xxxx	xxxxxxxxxxxx	xxxxxxxxxxxx	mm/dd/yyyy

Number of Rail Cars in XXX-Your Group Identifier-XXX: **8**

the Fleet Information table. This program should locate the most recently received CLM record for each rail car from the Rail Car Movements table and then locate the most recent shipment record for each rail car from the Shipment Information table. From the related fields in these two records, the program can select or calculate the data values which will then be saved in the Group Identifier field in the record for that rail car in the Fleet Information table. This process should not take more than a minute or two for each grouping category. There are, of course, other ways to accomplish this task, but this is a very direct and simple approach.

! **Programmers hark!** Here's your chance to display a few of your creative skills. There's a bunch of different ways to skin this cat and you can probably come up with some top-notch ideas of your own. There are three things to shoot for: speed, reusability, and simplicity. It goes without saying that you want the process to complete as quickly as possible. You also want to be able to create one process that is capable of doing several related tasks. Finally, you want the user to perform a minimum of keyboard aerobics to complete each task. In fact, it is preferable for the processes that complete the information collection and report production to execute with no operator intervention at all. Think about what you can do to help.

Here are a few samples to show you how you might use this feature. All of these are based on the Group Identifier collection process described above.

- **Non-Movement Days Groupings** - Subtract the sighting date in the Rail Car Information record from the current date in the computer. The resulting value is the number of days since the rail car last moved and it should be placed into the Group Identifier field. Users should be able to specify a threshold value for the reports which answers the question "I want to see all the cars which have not moved in **X** or more days. If the report is run without operator intervention, the threshold value should default to a predetermined value like 2 days.

- **Sightings by Origin Groupings** - Place the word "Origin: " plus the origin city/state from the Shipment Information record into the Group Identifier field. This will cause the most recent sighting for all rail cars leaving from common origin points to be grouped together.
- **Sightings by Destination Groupings** - Place the word "Destination: " plus the destination city/state from the Shipment Information record into the Group Identifier field. This will cause the most recent sighting for all rail cars going to common destination points to be grouped together.
- **Sightings by Product Groupings** - Place the word "Product: " plus the product from the Shipment Information record into the Group Identifier field. This will cause the most recent sighting for all rail cars with similar product identifiers to be grouped together.
- **Loaded/Empty Cars Groupings** - If the Loaded/Empty status code from the Rail Car Movement record equals "L", then place the word "Loaded" into the Group Identifier field. If the Loaded/Empty status code equals "E", then place the word "Empty" into the group identifier field. This will cause the most recent sighting for all rail cars in each load status to be grouped together.
- **Empty Releases Groupings** - Place the words "Empty Release from " plus the location city/state from the Rail Car Movement record into the Group Identifier field for every recent sighting record that is currently empty but was loaded on any CLM received prior to today. This will cause the most recent sighting for all rail cars released from common destination points to be grouped together.
- **Bad Ordered Groupings** - Place the words "Bad Ordered at " plus the location city/state from the Rail Car Movement record into the Group Identifier field for every recent sighting

record that has a "B" in the first position of the Action Code. This will cause all bad ordered rail cars to be grouped together by location. If you want to provide even more "bad order" detail, you can refer to the **Rail Users' Manual** for the complete bad order Action Code sub-class. In all honesty, though, most railroads do not transmit the sub-class codes, and if they do, the codes are inconsistent.

- **Loaded/Empty Swaps Groupings** - Place the words "Swapped to " plus the load status from the most recent sighting plus the word " from " plus the previous load status into the Group Identifier field for every recent sighting record which has changed load status today. This will cause the most recent sighting for all swapped rail cars to be grouped together by their swap direction: Loaded-to-Empty cars appear in the "Swapped to E from L" group and Empty-to-Loaded cars appear in the "Swapped to L from E" group. This is also a handy report to review on a daily basis to make sure that you don't have any CLMs with bad L/E codes. You will occasionally find a stream of empty CLMs for a rail car with one lone "L" stuck in the middle. This is obviously an erroneous sighting status, but it happens and this report will help you spot it. Just another example of management by exception, and it is something you could easily miss it if you only saw the most recent CLM for every car each day.

Origin and Destination Information

This table is designed to help you watch the moving cars and throw up exception flags when they hit the begin and end points on the route. You could also add enough information to highlight slow junctions or switch yards.

The basic fields in the Origin and Destination Information table are:

1. The **City** and **State** fields are the two key fields in the table. They are used when the system needs to look up a specific location to determine if it has arrived at the destination or returned to the origin.
2. The **Origin/Destination Code** field identifies whether this location is an origin or a destination. Of course, it is possible for a location to be both origin and destination, so the code values might be "O" for origin, "D" for destination, and "B" for both. If you have better ideas for code values, use them.
3. The **Load Status** fields indicate the anticipated loaded/empty status of the rail car when it moves to or away from the origin/destination location. It is used as an error-catcher. For example, if you were shipping a product from Atlanta to Memphis, the expected load status on the move away from Atlanta would be "L". The expected load status when the car arrives in Memphis would be "L" and when it moves away from Memphis the expected load status would be "E". If anything different shows up on the CLMs, up goes the exception flag.

! **Programmer's note:** We will discuss in one of the next sections of this chapter the kinds of things you will have to do in your programs to check for the exceptions. You traffic folks will probably have many additional conditions for which you should check, so make sure you set up your programs so that you can make quick changes to the code.

Data Access Principles

Most current day computer systems are designed and programmed using a modular approach. The separate modules are roughly oriented toward the functional and service requirements of the system. For example, in our system, the CLM Update module will read a CLM record, check the fields for validity, access the Rail Car Movements table, check for an existing record, and update the database table. Viewed from above, the express purpose of the CLM Update is to add the individual CLMs to the Rail Car Movements table. But, if we look closely at the data service-oriented parts of this function, we will find that there are repeated accesses to each database table for specific purposes (e.g., locate the previous record, delete the current record, insert a new record, and locate a specific record).

The "data access module" is considered a service module because it will be required by all other modules in the system. Getting the data access module to work correctly early in the development life of the system will save hours of individual programmer effort.

The data access module should be built as one of the first steps in the overall construction of the system. It should be maintained separately and religiously. All table access requirements should be written into the data access module. When any function-related modules need a record from any table in the system, the data access module is used. If a new access method is required (even if it is a one-time use), the new access procedure should be added to the data access module. This will make the system easier to maintain in the future.

! When Module A uses Module B in this manner, Module A is said to *call* Module B. This makes Module A the *calling module* and Module B the *called module*. To increase the reusability of the data access module, the calling module could supply the name of the database table and other special values pertinent to the requested process in the called module. These values are called *parameters* or *call parameters*.

Let's look at some of the processes that should be included in our data access module. Remember, these processes should be created as generic operations using the call parameters to identify the database table and specific function requests and to pass and receive values like key fields and operation return codes.

- **Table Open** - Prepare a database table for program access.
- **Table Close** - Remove a database table from program access.
- **General Record Locate** - Locate the first, next, previous, or last record in the specified database table and move to it.
- **Specific Record Locate** - Locate a record in the specified database table using the key field value supplied by the calling program.

- **Insert a Record** - Insert a record into the specified database table in front of the current record, behind the current record, or at the location specified by the key field value supplied by the calling program.
- **Copy a Field from the Previous Record** - Copy the value in the current field in the previous record into the current field of the current record. (This is really handy for data entry.)
- **Copy and Insert a Record** - Save all field values in the previous record, insert an empty record behind the current record, and copy the saved field values into the new record. (As above, this is primarily a data entry function.)
- **Delete the Current Record** - Remove the current record from the specified database table.
- **Delete All Matching Records** - Remove all records matching the key value supplied by the calling process for the specified database table.
- **Change a Common Field Value** - Change a specified field to a specified value for all records matching the key value supplied by the calling process for the specified database table.
- **Return the Record Number** - Identify the record number of the current record in the specified table and return it to the calling process.

- **Return a Record Count** - Return the total number of records in the specified table to the calling process.
- **Table Exists** - Return a value to the calling program indicating whether or not the specified table exists.
- **Table In Use** - Return a value to the calling program indicating whether or not another user or process is currently using the specified table.

There may be other data access procedures that you will want to add to your data access module. It may seem like a simple thing to create a specialized module like this, and it is. But, it is certainly not frivolous. The use of a data access module makes your RFMS independent of any database manager. Since all data is accessed through this one module, it is the only thing that would have to change if you migrate your system to a new database.

Operational Events

A system, by definition, is a repeatable flow of successive events arranged in a logical progression. That's the hard way of saying "first you do this, then you do that". This section will describe the "this and that" you need to do and the order in which it needs to be done.

CLM Collection - Since our rail fleet management system is based on the individual rail car movements made available from the rail carriers, the first thing we need to do is collect the CLMs. As we said in the previous chapter, where and how you get the CLMs is your business. Rail cars move 24 hours a day, seven days a week, but generally speaking, the collection should be done only once, maybe twice, each day. Depending on your computer platform, the CLM collection may be accomplished by packaged communications software. The process is often set up as a timed process and, along with the CLM Update described below, run in an automatic mode (no operator intervention required).

CLM Update - The CLM Collection and Update processes are the operational hub of any rail fleet management system. While the CLM collection can be implemented using an off-the-shelf communications package, the CLM Update is a specialized program, written by you, to read the CLM records according to the format you have selected from your CLM supplier, validate the CLM data fields, and add the movement data to your Rail Car Movements table. This is where you will provide most of the exception analysis. This program should be set up as part of a timed, automated process following the CLM Collection. You should also be able to run these two processes manually and independently, if desired.

It is important that the collection and update be intelligent enough to handle all types of data exceptions and operational errors without operator intervention. For example, the

CLM Update should never generate a message on screen that requires the user to reply with a key press or mouse click. It defeats the purpose of an automated process if the update stops until the user responds to the message "The rail car ABCX012345 was not found. Do you wish to add it? (Yes or No)". Also, any reports should be directed to a disk file and not printed while the process is running. This will prevent the system from hanging in the middle of the update waiting for more paper to be put in the printer. The reports can be automatically printed as the last step of the automated process.

The records that come from your CLM source are updated as discrete units of information. That is, there is no information in the CLM record that is dependent upon previous or subsequent CLMs. When the Rail Car Movements table is updated, however, there are certain things you should do as each CLM is added:

⇒ Determine if the rail car exists in your system. If it does not exist, add it to the Fleet Information table with any predefined (default) field values. For example, you might want to add new cars with a fleet identifier of "NEW RAIL CAR!" so that it will clearly show up as an exception.

⇒ This would be the place to adjust the location city/state and destination city/state field values if you are using the City & State Translation function as described in the **Extending the System** chapter.

⇒ Check the preceding movement record for this rail car. Bring forward the ShipDate and, possibly, the Destination City and State. If there is a destination in the new CLM, you may want to save it or ignore it depending on the status of the Loaded/Empty field. For example, if you have a loaded shipment going to a customer, you probably want to save the destination city and state from the CLM into the Rail Car Movement record if the Loaded/Empty status of the CLM is "L". If the Loaded/Empty status is "E" for the same shipment, you may want to ignore the CLM's destination and leave the original loaded destination in the Rail Car Movements records. If you are using the Shipment Information, you may not want to bring forward the destination city/state at all. This is a decision your logistics design team will have to make.

⇒ Check the location city and state to see if the rail car is leaving an origin or arriving at a destination. You may want to mark the record with a special code value indicating this status.

Shipment Update - You can create an automated or a manual process connecting your order department or your shipping department to the rail system. If your shipment information is already computerized, you could build a "linking" process to transfer the shipment data from your current system into the Shipment Information table of our rail system. If your shipments are not computerized, you will need to manually enter the shipments into the Shipment Information table.

In either case, the part of the program that handles the Shipment Information record update should place the shipment date into the ShipDate field of the Rail Car Movement records for the associated rail car that have dates equal to or later than the shipment date. This will connect the shipment for that rail car with each of those movements and allow a relationship to be established between a group of movements and a shipment (and vice versa). You will see the importance of this when we begin to expand the system later. You should also decide if the destination city and state in Shipping Information should override the destination city and state in the Rail Car Movements. If so, locate the first movement for this shipment and update the movement record with the ShipDate and the destination city and state.

If you have an automated connection to your shipments, you will probably want to update the Shipment Information table once each day. If you are entering the shipments manually, you may want to move in and out of the data entry process as your time and workload permit. On the other hand, if you have a lot of shipments you should have a user whose responsibility it is to update the shipments all during the day.

! If you are supplying the origin and destination information through your Shipment Information, you may not need to pay your CLM source the extra fee for supplying destination information on the CLMs. On the other hand, you may choose to use the "destination check" method described in the **Exception Analysis** section below, in which case, you should continue to receive "destination" CLMs.

Exception Analysis - OK, you've collected the CLMs, updated the Rail Car Movements table and the Shipment Information table, and you're ready to do a little management by exception. Every company will have its own set of rules that it uses to determine when a rail car is an exception, but we will give you a few of the most common indicators.

⇒ **Non-Movement Days** - Back in the **Fleet Information** section of this chapter, we described a question that every transportation manager needs to have answered: "Where are the cars that have not moved in **X** days?" The **X** value changes from day to day, but your organization probably has a critical **X** value that could be used as the default value for an automated daily report. You should also be able to specify any **X** value you want if you decide to run the report manually.

⇒ **Movement without Shipment Orders** - If you are maintaining your Shipment Information religiously, you could select all the rail cars that are currently moving away from origin without a corresponding shipment. This is a bit more complex than the other exceptions because it is dependent upon several critical elements of the system. In order to determine this exception condition, you must be checking the origins and destinations and marking the cars in the Rail Car Movements table when they leave the origin and arrive at the destination (and return to the origin, if your cars go on circular trips).

- ⇒ **Bad Ordered Cars** - This is an easy one. Just check the current movement record for a "B" in the first position of the Action Code field. If it's there, it's a bad ordered car. You may want to extend this exception category to include delayed or held cars. If so, check the Action Code for a "B" or an "H". The "H" is the "Car Delayed or Held" railroad sighting code. More sophisticated systems will create a history of the bad ordered/held cars and perform a trend analysis once or twice each year.
- ⇒ **Release From Bad Order** - You could just check the current movement record for a "G" in the Action Code field. If it's there, it's a bad order release. However, you are depending heavily on the railroads to transmit "good" order CLMs. A more thorough method would be to check the preceding movement of every rail car's current CLM for a bad order. Assuming that the current CLM does not continue the placement of the car in the bad order/held classification, it must be moving again which means it has been released from its previous delay condition.
- ⇒ **Loaded/Empty Swaps** - This exception turns out to be very handy at spotting interesting problems. Check the preceding movement of every rail car's current CLM for a Loaded/Empty status different from its current status. If it is different, then the car has gone from loaded to empty, empty to loaded, or there is an error in the CLM itself. If you have provided the correct Origin/Destination

information, then you can check the current location against the origins and destinations to see if the car is placed correctly or is truly an exception. Sometimes the railroads do transmit a CLM with an erroneous load status. It will appear as an exception. You could probably ignore it, but you may want to check with the carrier to make sure the status is or is not correct. For example, if you have a loaded shipment going from ATLANTA to PHOENIX and it suddenly appears as an empty in LITTLE ROCK, it's time to phone the carrier.

If you extend the capabilities of this exception to include a beginning and ending date specification, you will come up with a powerful tool for collecting outbound shipments or completed shipments. For example, if you normally ship out loaded cars and you use this function to collect a list of all the loaded-to-empty swaps between "05/01/2002" and "05/31/2002" from your Rail Car Movements table, you just listed all of the shipments which completed their deliveries during the month of May.

⇒ **No Billed Cars** - If you find an "N" in the Action Code field in the current movement record of each rail car, you have found a car without a waybill. This is an exception condition. If a "no billed" car also has a Loaded/Empty status of "L", you've got a big-time exception. For one thing, the "N" sighting code is a dinosaur that refuses to die. The "N - No Bill" railroad sighting code was supposed to have been replaced by the expanded bad order reporting codes ten years ago. But they keep showing up. Can you say "slow to change", boys and girls? We thought you could.

⇒ **Destination Check** - Here's an interesting exception. If you are getting the "destination" CLMs and you are maintaining your Shipment Information table, you can check the destination city and state values when they appear in the CLMs against the destination city and state in your shipment records. If they don't match, you've got an exception. Also, you can compare the location city and state in the CLM that moves it away from its destination against the destination city and state in the shipment record. If they are not the same, it's probably not an exception, but it may be an indicator that you should verify the city and state values you are using in your shipments. If it's just a different spelling, check out the section on **City & State Translation** in the next chapter.

⇒ **Carrier Check** - This exception requires that you build a little intelligence into your rail system. If your rail cars move in a specific geographical area all the time, you could create a list of all railroads in that area. Checking the railroad initials in each CLM against the railroads on your list would produce an exception condition if the railroad is not found on that list. If you want to expand this capability, read the section on **Route Analysis** in the next chapter.

⇒ **New Rail Cars** - If you are running only temporary or assigned cars, new rail cars in the system would not be considered to be an exception. Still, you might want to be notified when a new car is added or you might want to perform some additional editing when it first appears. If you own or lease your rail cars, this becomes more important. You would definitely want to be notified when a new rail car enters your system.

⇒ **Future CLMs** - A "future CLM" is a CLM record that appears with a sighting date in the future (defined as tomorrow or more). Future CLMs usually come in with an Action Code of "3". The railroads call this an advanced ETA. They should be examined carefully when you first begin using the system. If you find them useful, then you can remove them from the exception classification and create a report grouping for them. We do not recommend that you store them in the Rail Car Movements table. They are not real (even if they appear to be accurate) and they will only gum up the works. Use them as reference – they can be helpful, but don't store them.

⇒ **Bad CLM Data Fields** - There are only a few things that can go wrong with the CLM data:

1) Bad Characters in the Rail Car Initials or Number fields - Log the CLM as an exception and throw the thing away.

2) Bad Date or Time Values - Log it and throw it away.
3) Bad Action Code - Log it, but save it with "???" in the Action Code field.
4) Bad Loaded/Empty Status - Log it, save it with "?" in the Loaded/Empty status, and check with the railroad.
5) Bad Railroad Initials - Log it, save it with "????" in the Railroad Initials field, and compare it to the preceding and subsequent CLMs for that car or against the moves by other cars on the same route.

Any CLMs that were saved in the Rail Car Movements table that had question marks in the data fields should be retained no more than a week. If it hasn't been fixed by a subsequent CLM or updated manually by you, throw it away. It is rare that you will find a bad CLM. The carriers have a very good accuracy rating.

Daily Reporting - There are as many options to choose from here as there are programmers in heaven. Well, maybe more than that. In the **Fleet Information** section of this chapter, we described how to create a generalized report that would cover many different groupings. You can extend those groups to include additional reports that suit your specific needs. Try to settle on a fixed group of reports that can be produced automatically each time the CLM Collection and Update process is run. This may include grouped reports or reports based on the exceptions described above.

Because rail fleet management is so much a management by exception process, you must include a powerful query facility for your system users. The ability to ask complex questions about the data in the system and get valid answers is second in importance only to the reliability of the system. If your users are required to dig out their answers with a pick and a shovel, the system is no more useful than if it locks up three or four times a day. Either way, it won't be used. When you select your database manager, keep in mind that you will need a good querying tool and a good reporting tool. Train your users extensively in the use of these tools and closely watch the way they use them. Then, if you can design a programmed function which saves the users time, do it. If you can't, stay out of their way.

You will need three kinds of reports: predefined automated reports, predefined manually selected reports, and dynamic manually selected reports. If your query and report tools can provide all three kinds, great! If not, try to hide the differences from the users so that they only have to learn one set of query commands and one report construction set.

General Use - Any rail fleet management system is primarily designed to provide information to the transportation department users. It is natural, however, to expect other departments to want access to specific parts of the system that may help them do their jobs. Who gets to see what and do what will vary with every implementation of the system, but we can give you some general guidelines on providing system access or new functions.

- ⇒ **Define what is needed before you provide it.** If a user requests a new function, don't forgo the principles of good systems design in your quest to provide the function. Talk with the user and find out exactly what is being requested. Determine if it is already in the system. Determine who else might use it. Let the development team determine if it is really needed. If it is, design the function. Go through a series of check-off meetings with the user in the design and programming phases. Get the user to help test the new function. Provide documentation, training, and support when the function is made available.

- ⇒ **Keep the functions simple and focused.** This has been a design goal throughout this project. Multi-colored information boxes strewn about on a screen can look really neat, but if you can put everything a user wants to know in a single line, do it. On the other hand, think about what the users are asking for and what they really need. Sometimes users don't know how complex or how simple their request is, so it is up to you to help them out. For example, if they ask for a report which shows only RailCarID, ShipDate, Origin, and Destination you should suggest that they might also need the ability to select specific values in all those fields. They may have actually only wanted to see these fields for a specific destination, not realizing how huge a stack of paper they would have gotten if they had received exactly what they requested.

⇒ **Make the entry and exit speedy.** This ought to go without saying by now. Speed! You gotta have it! Nobody willingly uses a system that putts along.

⇒ **Provide quality support to those who need it.** Be prepared to commit your resources to training the users, providing telephone and desk-side support, and written guides. This goes for the original basic system and any extensions you make. We said above that nobody uses a system that putts along. It is also true that nobody uses a system they don't know how to use.

⇒ **Limit the ability to change real data.** Every user who can change data will change data. Either by accident or on purpose, they will do it. So, if you don't want users to change data, give them their own copy of the data and let them trash it all they want. This is especially true if a particular user does not need to make changes to the tables in our RFMS. Let them eat copies!

⇒ **Provide access to network users.** Once the system is implemented, advertise it. Let it be known that it exists and that authorized users can be added to the system quickly and easily across the network. The mailroom clerks probably have no use for a rail fleet management system, but people in shipping, production, sales, and other departments may need some information out of it.

Never forget though, when somebody says they need access to the system, the first words out of your mouth should be "OK, why?" This implies that you are willing to give them access if they can give you a valid reason why they might need to use it. This is true… isn't it?

⇒ **Provide access to on-line users.** Let your customers get into the system. In these days of expanding corporate partnerships, many companies are providing their customers with a gateway into their computer systems. With just a few extra pieces of equipment, you could set up your system so that the companies to which you ship products can connect their computers to your new rail system and look at their incoming shipments. But you want to make sure that CLANCY MILLS cannot see the shipments to REARDON INDUSTRIES.

While there are several different ways to do this, one of the easiest methods is to print a report to a disk file and allow the customer to look at only the data in that report file, not the actual data in the system's table set. You do this by creating groups of rail cars based on the customer names and then printing the report for each customer group. For more on how to do this, refer to the section in this chapter called **Fleet Information**. If you don't want to let your customers into your system, you can create the reports just as we described and then email or fax them straight out of the computer to your customers on a daily basis. This can be automated so that it requires none of your time or attention.

⇒ **Keep the system alive.** When you finally get everything up and working, your users trained and happy, and you have received a bonus for all your hard work, it ain't over! Like every other living thing, a computer system evolves over time. When your users see what can be done, it may – it should! – change the way they do their work. That, in turn, may change the information they want to see in the system. Be ready! And, be willing. But don't forget the principles of good system design. Listen, analyze, think, and then ask why.

Documentation

"This manual is lousy!"

This is, perhaps, the most persistent general complaint we have heard from users about all the rail systems we have seen over the last twenty years. It does not matter whether it's a tiny little one-table tracking program or a full-blown, multi-thousand dollar packaged system, the documentation is usually the weakest point.

There are several reasons for this. First, the rail system vendors will spend all of their development money putting features in the system, saving the documentation for last. And then they don't have any resources left to write the manuals. Second, the budget for documentation is always underestimated. It actually takes almost as much time to create good, usable manuals as it does to develop the system. Third, most developers are technical folks and techie-types don't always make the best writers. Science fiction, maybe. Cyber-action-drama, probably. But, support manuals? No way! The problem is, there's often no one other than the lead technical programmer who understands all of the nuances of the system well enough to write the manuals. Fourth, by the time the system is ready for the manuals, everybody on the developer's staff is sick to death of looking at the thing. Nobody wants to spend another three to six months picking at it. It's like a Thanksgiving turkey on the next Wednesday. Fifth, the developer never looks at the documentation the same way the users do. Most developers view the manuals as the last hurdle that keeps them from marketing the system. The users see the manuals as their only salvation when they have a problem at 4:30 PM on a Friday afternoon. It's a developer's conundrum, but it's the user who bears the weight. Sixth, most developers are not rail people, they are computer people. As a result, they don't know enough about the rail tracking or fleet management to relate the functions of the system to the work performed by the rail users.

In truth, most developers do not realize that the documentation is the first impression their system makes on the users. The color of the binder, the design on the cover, the typeface used in the printing, the white-space on each page; these are the things the user notices long before the system’s first screen appears. It’s that important!

But what about you? After all, this is just a simple rail tracking project, an in-house system. You don’t have the resources to write a manual, right? You’d better find them! If you don’t, you are going to be spending more “resources” in telephone and desk-side support than you ever thought possible.

If you don’t want to write an honest-to-goodness manual, write down the individual process steps and distribute them to the users. Ask the users to jot down the things they do as they do them, and then you can formalize their notes. List the information you presented to management. Collect the course outlines you used in the first few training sessions. Record the questions that were asked and the answers that were given. (Research the answers to make sure they were correct.) Get the transportation members and programmers who were on the logistics design team to do the same thing and then have the best writer you can find organize the information and create a presentable package.

You don't need to have a ton of on-screen help or a CD-ROM supporting the system. (You can't read a CD in the bathroom!) You just need to provide good, solid, printed information. The things the users want to know about the system are:

1. What must they do with the system to get their job done?
2. What can the system do to make their jobs easier?
3. What can the system do to make their job better?
4. How should they do all of the above – step-by-step.

Archival and Backup

It takes about six months of enthusiastic operation before the lightning bolt strikes you. Hey! This database is getting HUGE! Yeah, well, don't say we didn't warn you. Remember our formula for calculating the size of your database from back in the previous chapter?

#Cars X **#Chars** X **#CLMs** X **Months** X **Years** = **Size of CLM History**

It does accumulate. Quickly! There will soon come a time when you need to remove records from the database tables and save them in a separate file, and you are going to want to do it in an organized manner. This is called *archiving*.

Most database management software will provide a method for removing the records via a query, but you should create a programmed method which can be done selectively by the users or automatically at predetermined intervals: weekly, monthly, quarterly, annually, etc. The only two tables in our project that will need to be archived initially are the Rail Car Movements table and the Shipment Information table. These are dynamic tables, as opposed to static tables like the Origin/Destination table (which only changes when we add a new customer or plant site).

Archiving is relatively simple. In the case of the Rail Car Movements, you just delete all the records within a given date range. Almost. The Rail Car Movements table must retain the last sighting record for each rail car. If you remove all records for a given rail car, you will not know where that rail car is located until it moves again. There are a few other things: 1) don't archive a car during a shop trip; 2) don't archive a bad ordered car; 3) don't archive temporary cars until they are no longer in your service.

There are more sophisticated methods of archiving the CLMs. For example, you could use the Shipment Information table and remove all the rail car movement whose ShipDates are within a specified date range, keeping in mind that you have to leave the last sighting record for each rail car in the table. This is called *archiving by trip* and it is usually easier to find old movements when they are grouped this way because it keeps all of the CLMs for each trip together in an accessible clump. (Note: *clump* is a highly technical, semi-Latin term meaning "a really big group").

We recommend that when you archive, you save the records in a table structure similar to the original, but with a name that helps identify the nature of the archive group. For example, if you archive by quarter, you could identify the archive file as "CM1995Q1" which stands for "Car Movements for the First Quarter of 2002". (Of course, if your platform allows long file names, this is not a problem.) Using this technique, you could easily find a series of movements for a given rail car within a specified time period. It's a good idea to keep as much movement information in your active database as possible. The reasons for this will become more obvious when we begin to describe the extensions to the basic rail tracking system in the next chapter.

We have talked about archiving, now let's discuss backing up your data. A *backup* of a data file is a copy of the original, made at a specified time interval, and stored in a safe place. A backup does not require any record deletion nor does it require any fancy programming. You just need to remember to do it. If your local power company zaps your computer with a power surge, it can destroy your disk storage. If you have been faithful to your backup regimen, this would only be a minor annoyance. Assuming that you have spare equipment, you could be up and running in as little as an hour.

Backing up your system is just like exercising for fitness, you have to do it consistently for it to pay off. You might even want to combine two or more backup methods. For example, every morning you make a copy of your rail system onto another physical device (i.e., copy from drive "C" to drive "D") and every Friday you copy your system to a backup tape or a removable hard drive. This procedure will protect your data as much as possible. Like the Nike ad says, "Just do it!"

Record Layouts and Event Diagrams

Record layouts are used to name and describe the fields in the records of the database tables. We have listed the fields in each table along with the field descriptions, characteristics, and a range of possible values. The characteristics of a field include the character composition (alpha only, numeric only, or mixed), the length, and a key field identifier (**k1**, **k2**, etc.). We have also included information for the City & State Translation table as described in the **Extending the System** chapter. This should be one of the first extensions you make to the basic system.

Following the record layouts is an expanded version of the basic RFMS model that we presented in a previous chapter. In database terminology, this is called a *schematic*. In the schematic, as opposed to the more compact model diagram shown earlier, the relationship characteristics between or among the tables are shown using special symbols.

The relationship between any two tables is defined by its cardinality and optionality. *Cardinality* expresses a relationship in terms of the type of link: one-to-one, one-to-many, or many-to-one. It is denoted on the schematic by the termination of the relationship (connecting) line. If the line ends with a single contact point, it is said to have a cardinality of "one". If the line ends in multiple contact points (i.e., it looks like a crow's foot), it expresses a cardinality of "many". Optionality expresses that a relationship may be either "mandatory" or "optional". A *mandatory* relationship is denoted by a bar that crosses the relationship line and indicates that the table **must** contain a related record. An *optional* relationship is indicated by a circle. A good example for both cardinality and optionality is exemplified by your relationship to your social security card and your credit cards. The relationship between you and your social security is a cardinality of one, whereas you relationship to your credit cards has a cardinality of one (you) to many (more than one credit card). You must have a social security card, therefore you have a mandatory optionality. You may chose to have or not to have a credit card; this expresses an optional relationship. There are other expressions of relationship, but these are the most often used.

The last set of diagrams defines the "flow" of events through the system. Beginning with the collection of the CLMs and ending with an archive, the symbols highlight the procedural steps of the basic RFMS. These are not meant to be definitive flowcharts. You may want to produce flow diagrams expressing more detail as you begin to define your own bells and whistles.

Rail Car Movements Table

Field Name	Description	Characteristics	Range
RailCarID	The initials and number of the rail car.	AAAA###### **k1**	Minimum:"A 000000", Maximum:"ZZZZ999999"
SightingDate	The date the rail car was sighted.	mm/dd/yyyy **k2**	Any valid date value
SightingTime	The time the rail car was sighted.	hh:mm **k3**	Military Time, 00:00 thru 23:59
ActionCode	The railroad sighting code, AKA the AAR standard movement transaction code.	XXX	One to three characters, see **Rail Users' Manual**
LorE	The code indicating the loaded or empty status of the rail car.	X	"L" or "E"
RailroadID	The AAR standard initials identifying the sighting carrier.	XXXX	One to four initials, see **Rail Users' Manual**
LocationCity	The city where the rail car was sighted.	XXXXXXXXXX	One to ten characters representing city
LocationState	The state where the rail car was sighted.	XX	Valid two-character state abbreviation
Junction	The junction railroad identifier or train number.	XXXXXX	N/A
DestinationCity	The city where the rail car is going on the current leg of trip.	XXXXXXXXXX	One to ten characters representing city
DestinationState	The state where the rail car is going on the current leg of trip.	XX	Valid two-character state abbreviation
ShipmentDate	The date of the shipment associated with this rail car.	mm/dd/yyyy	Any valid date value

Shipment Information Table

Field Name	Description	Characteristics	Range
RailCarID	The initials and number of the rail car.	AAAA###### **k1**	Minimum:"A 000000", Maximum:"ZZZZ999999"
ShipmentDate	The date of the shipment associated with this rail car.	mm/dd/yyyy **k2**	Any valid date value
ShipmentTime	The time the rail car was shipped (Optional Field).	hh:mm	Military Time, 00:00 thru 23:59
LorE	The code indicating the loaded or empty status of the shipped rail car.	X	The only possible values are "L" or "E".
Customer	The name of the customer to whom the rail car is being sent.	XXXX...XXXX (Length as required.)	Defined by User
Route	A short text description of the rail car's route.	XXXX...XXXX (Length as required.)	Defined by User
Product	A short text description identifying the product shipped.	XXXX...XXXX (Length as required.)	Defined by User
ProductVolume	The volume of product shipped.	###,###.###	0 to tare weight of this rail car
UnitOfMeasure	The unit of measure of the product shipped.	XXXXX	Predefined as "GALS", "TONS", "LBS", etc.
DetentionCode	A code value identifying the detention characteristics for this customer and/or product.	XXXX...XXXX (Length as required.)	Defined by User

Shipment Information Table continued on next page...

Shipment Information Table (Cont'd)

Field Name	Description	Characteristics	Range
ShippingCode	A code value identifying the shipment characteristics for this customer and/or product.	XXXX...XXXX (Length as required.)	Defined by User
OriginCity	The city from which the rail car shipment originated.	XXXXXXXXXX	One to ten characters representing city
OriginState	The state from which the rail car shipment originated.	XX	Valid two-character state abbreviation
DestinationCity	The city where the rail car is going on the outbound leg of trip.	XXXXXXXXXX	One to ten characters representing city
DestinationState	The state where the rail car is going on the outbound leg of trip.	XX	Valid two-character state abbreviation
Shipment#	The sequential number (or code) identifying this shipment.	XXXXXXXXXX	Defined by User

Fleet Information Table

Field Name	Description	Characteristics	Range
RailCarID	The initials and number of the rail car.	AAAA###### **k1**	Minimum:"A 000000", Maximum:"ZZZZ999999"
Status	Status code values established by user (Active, Ownership, Car Type, etc.).	XXXX...XXXX (Length as required.)	Defined by User
FleetID	A code value or text description identifying the fleet group for this rail car.	XXXX...XXXX (Length as required.)	Defined by User
OtherGroupings	A code value or text description identifying the other groups for this rail car. This may be filled in dynamically.	XXXX...XXXX (Length as required.)	Defined by User
Other Possible Fields	Tare weight of car, mileage credit rate (all countries), description of car, service dates, cost per month, etc.	XXXX...XXXX (Length as required.)	Defined by User

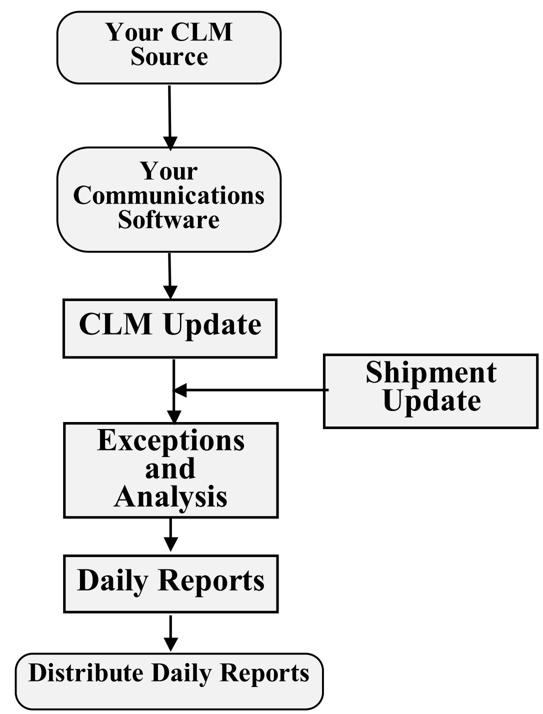
Origin and Destination Information Table

Field Name	Description	Characteristics	Range
LocationCity	The city identifying an origin or destination.	XXXXXXXXXX **k1**	One to ten characters representing city
LocationState	The state identifying an origin or destination.	XX **k2**	Valid two-character state abbreviation
O/D Codes	The code value indicating whether this is an origin or a destination.	X **k3**	The only possible codes are "O" or "D".
LocationCarrier	The AAR standard initials identifying the carrier for this location.	XXXX **k4**	One to four initials, see **Rail Users' Manual**
ArrivalLEStatus	The code indicating the loaded or empty status of the rail car as it arrives this location.	X	The only possible codes are "L" or "E".
ArrivalActionCode	The expected railroad sighting code values for arriving movements.	XXX,XXX,XXX, XXX,XXX	Up to five different one to three character codes, see **Rail Users' Manual**
DepartureLEStatus	The code indicating the loaded or empty status of the rail car as it departs this location.	X	The only possible codes are "L" or "E".
DepartureActionCode	The expected railroad sighting code values for departing movements.	XXX,XXX,XXX, XXX,XXX	Up to five different one to three character codes, see **Rail Users' Manual**

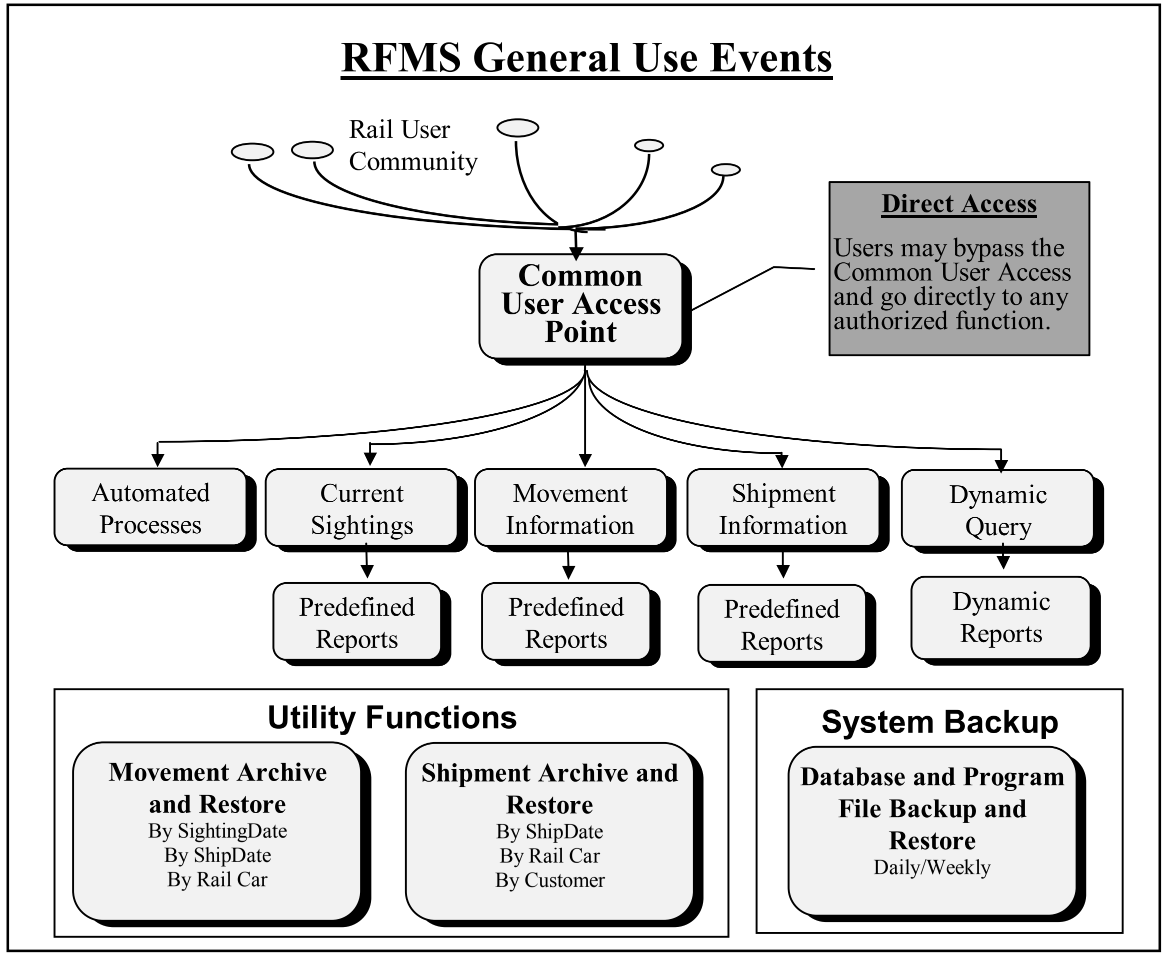
City & State Translation Table

Field Name	Description	Characteristics	Range
TranslateCity	The incoming CLM city where the rail car was sighted.	XXXXXXXXXX **k1**	One to ten characters representing city
TranslateState	The incoming CLM state where the rail car was sighted.	XX **k2**	Valid two-character state abbreviation
LocationCarrier	The AAR standard initials identifying the sighting carrier on the incoming CLM.	XXXX **k3**	One to four initials, see **Rail Users' Manual**
LocationCity	The city to which the incoming city will be translated.	XXXXXXXXXX	One to ten characters representing city
LocationState	The state to which the incoming city will be translated.	XX	Valid two-character state abbreviation

RFMS Daily Events



Automated Daily Events in the Basic Rail Fleet Management System


RFMS General Use Events
Rail User Community
Common User Access Point
Direct Access
Users may bypass the Common User Access and go directly to any authorized function.
Automated Processes
Current Sightings
Movement Information
Shipment Information
Dynamic Query
Predefined Reports
Predefined Reports
Predefined Reports
Dynamic Reports
Utility Functions
Movement Archive and Restore
By SightingDate
By ShipDate
By Rail Car
Shipment Archive and Restore
By ShipDate
By Rail Car
By Customer
System Backup
Database and Program File Backup and Restore
Daily/Weekly

6. Extending the System

Here is the fun part! This is where we really begin using the system to deliver some magical information. But, before we get into it, let's take a moment to review what we have created thus far.

If everything has gone according to our plan, we have a working rail tracking system capable of following our rail cars from origin to destination (and back). We can identify rolling exceptions and delay problems quickly and we can supply reliable shipment and movement information to all corporate groups who have the need to know. Our customers are happy because they receive (or access) daily reports showing the current status of each of their shipments. We have been using the system long enough that we have built up a sizable amount of movement and shipment history. The users have become educated in the operation of the system and they are beginning to generate ideas of their own for additional functions and reports. Life is good.

It's time to take the next step. This one is not as big as deciding to build the original system, but it will produce the most impressive responses throughout the company. We have a working system, and that's cool. Henry Ford built decent cars. But, haven't you ever wanted to sit back in the handcrafted leather seats of a new Mercedes-Benz, or listened to the purr of a Porsche Turbo Carrerra engine and dreamed of a two-hour run on the Autobahn, or imagined being buckled into a Formula racer as it snaked gracefully through the esses at Watkins Glen. Well, it's time. We're about to turn our Model T into an Indy racer. Gentlemen… and ladies, start your engines!

City & State Translation

We have mentioned this function a few times in previous chapters and, by now, you have probably been using the basic tracking system long enough to realize why it is necessary. It is an unfortunate fact of life that if there is any possibility of spelling the name of a rail location city more than one way, it will be done. You could call this the rail correlation to Murphy's Law. Back in Chapter Five, in the section on **The Basic RFMS Model**, we gave you the example of three different ways that railroads report sightings at East Saint Louis. It does not matter whether this is typical or even factual, if it happens at all, anywhere, we have to be prepared to deal with it. That's what the City & State Translation function is designed to do.

Our window to the outside world is the CLM record. It is what powers the system. Our CLM Collection process simply gathers the records and turns them over to the CLM Update process for validation. The City & State Translation function is a part of the CLM Update. After all the other the field values in the CLM record are validated, the City & State Translation function is called to check the location and destination city and state fields for compliance with our choice of spellings. The city and state fields are used as lookup values to locate the TranslateCity and TranslateState in the City & State Translate table. There are only two possible results of the lookup:

⇒ If a matching pair of fields is found, the LocationCity and LocationState fields from the City & State Translation table replace the original CLM fields.

⇒ If a matching pair of fields is not found, the CLM is saved into the Rail Car Movements table using the LocationCity and LocationState values that came in on the CLM. From time to time, you should check all of the city and state combinations that come in on the CLMs and add any new spellings into the City & State Translation table so that this new spellings will be translated correctly in the future. Also, don't forget to correct the spellings already in the Rail Car Movements table.

If you think we are translating these spellings because we are neatness freaks, you are wrong. We are doing it because the system extensions we are about to make are based on consistent data values in all fields, but especially in the city and state fields. This is very important. That's why we listed this as the first extension to the basic tracking system. In the **Record Layout** section, we established the first three fields in the City & State Translation table as key fields. This is optional, but many database management programs require lookup tables to be keyed this way.

Bad Order History

Everybody who receives CLM information looks at the bad orders. There are two reasons for this: first, it's easy; second, it's important. We have already talked about a report that looks through the current sightings and gives you a list of all of the bad ordered cars. We have also discussed a second report that looks for the bad order release code so you will know when your bad ordered cars start moving again. But, now we are going to do a little more.

We have access to the history of the rail movements for as long as we want to keep them. Sooner or later you will need to archive the movements because they accumulate so quickly. When you archive them, you lose easy access to the logistics events that happened in the past. This makes it difficult to analyze specific events as part of a trend. Since controlling bad orders is vital to good traffic management (especially repetitive bad orders) and since they are so easy to deal with (from the computer's point of view), we will show you a way to isolate the bad order events so you can perform the trend analysis any time you choose.

We need to add a new table to our system. We'll call it the Bad Order History and it should look like the table shown on the next page. As you can see, we have set up data fields so that we can see the dates, times and sighting codes for both the bad order and the release from bad order. This allows us to keep all the pertinent information together in one record.

Now, we need to write a program that runs as a part of the Exceptions and Analysis process in the daily automated events. This program will scan the current sightings (and several records prior to the current sighting) looking for new bad orders. If found, the new bad orders will be added to the Bad Order History table.

Bad Order History Table

Field Name	Description	Characteristics	Range
OpenBO	A code indicating whether or not the rail car is currently bad ordered.	X **k1**	The only possible codes are BLANK or "C".
RailCarID	The initials and number of the bad ordered rail car.	AAAA###### **k2**	Minimum:"A 000000", Maximum:"ZZZZ999999"
BODate	The date of the bad order associated with this rail car.	mm/dd/yyyy **k3**	Any valid date value
BOTime	The time of the bad order associated with this rail car.	hh:mm **k4**	Military Time, 00:00 thru 23:59
BOCode	The railroad sighting code indicating the bad order.	XXX	One to three characters, see Rail Users' Manual
LorE	The code indicating the loaded or empty status of the rail car when bad ordered.	X	The only possible codes are "L" or "E".
RailroadID	The AAR standard initials identifying the sighting carrier of the bad ordered rail car.	XXXX	One to four initials, see Rail Users' Manual
BadOrderCity	The city where the rail car was bad ordered.	XXXXXXXXXX	One to ten characters representing city
BadOrderState	The state where the rail car was bad ordered.	XX	Valid two-character state abbreviation
ReleaseDate	The date of the release associated with this rail car.	mm/dd/yyyy **k3**	Any valid date value
ReleaseTime	The time of the release associated with this rail car.	hh:mm **k4**	Military Time, 00:00 thru 23:59
ReleaseCode	The railroad sighting code indicating the bad order release.	XXX	One to three characters, see Rail Users' Manual

RouteCode	A unique code identifying this route (Optional Field).	XXXX...XXXX	As defined by the user

Next, the program will scan the "open" bad orders to see if the associated rail car has been released from the bad order condition. If it has, the bad order is marked as "closed" and the date, time, and sighting code of the release are stored in the history record. You can now write queries and reports to selectively analyze the Bad Order History records. For example, you may wish to see how many bad orders were reported on the **BN** between 01/01/2002 and 03/31/2002. Or, maybe you have a hot spot on one of your traffic lanes where rail cars seem to get bad ordered more often than elsewhere. Since we now have one record for each bad order, you can summarize the delay times and produce numbers that will help you control the efficiency of your fleet. Over time, an analysis of the Bad Order History will yield the mean time between failures, average delay per bad order, expected bad orders per year, etc.

! Just as a side-note for you programmers, many of the extensions we will be talking about in this chapter are "scan" function like the Bad Order History. These scans require time to read the complete set of current sightings movements. You may find it beneficial to group as many of the exception and analysis operations as possible into a single program so the movements only need to be scanned once. Also, you can create hierarchies of functions and speed the process up using the table structures. For example, you don't need to scan the Rail Car Movements two times for the Bad Orders: once for the Bad Order/Release Report and once again for the Bad Order History. Do the Bad Order History scan first and then base the Bad

Order Report on the Bad Order Information table. It contains everything you need for the report and it's faster to access because there are fewer records in the table!

Trip Analysis

Some things are more fun than others. This is one of our favorites. Perhaps it is because the results are so gratifying. The Trip Analysis function will generate the information that we will use in many of the other system extensions. It is not complicated, but there is a lot of work to do. Fortunately, it is all done by the computer. We will use the SightingDate and SightingTime fields in four specific movement records relating to a single shipment for each rail car. By subtracting the date and time in one movement record from the date and time in a later movement record we can determine the interval of time between the two movements. This interval is usually expressed in DAYS, but it could be expressed in HOURS and MINUTES if you need a more precise measurement for your trip metrics.

We are going to make a few assumptions here. Yes, that's a dangerous thing to do, but there's no way around it. We are going to assume that you have set up the Shipment Information table the way we described it, and that you have been faithfully entering your shipments into the basic system. You also should have about six months or more of rail car movements and shipments in your database for the trip calculations to be meaningful.

We need to design a program that will search through the Rail Car Movements table using the RailCarID and ShipmentDate fields in the Shipment Information table as the basis for marking the beginning and ending movement records for each <u>completed trip</u>. Notice, we did not say completed shipment. That's a very different animal from a completed trip. A "shipment" is considered complete when the empty rail car is released from the destination (customer) after the product has been delivered. A "trip" is not complete until the rail car returns to its origin and is made available for a new shipment.

! Please remember we are using our original trip "perspective" that we established in Chapter Two. Our perspective is that we are using owned or leased rail cars that travel <u>outbound loaded</u> with product to a customer. At that point, the cars are unloaded and they <u>return empty</u> to our origin. The cars are making circular trips and the "destination" is always considered to be the customer's city and state. If your perspective is different, you can make the mental adjustments in the remainder of this function description. If left-handed people can learn to tie their shoes and drive stick-shift cars, you should be able to do this with no trouble. No disrespect intended.

In addition to marking the beginning and ending movement record for each completed trip, we need to mark the movements that identify the placement at destination and the release from destination. Finding these two points is not difficult, however, providing enough computerized intelligence for the program to consistently mark the correct placement points can be a real hair-puller. The first step is to find the loaded-to-empty movement pair. Mark the empty movement as the release from destination; the railroad sighting code is not significant on the release. Now, look backwards through the stream of loaded movements to determine which movement defines the best point of placement at destination for you.

! If you are wondering exactly what we mean by *marking* the movement records, here is the explanation. To mark a movement means to flag the record in such a way that you can access the information in each of the four $\mathbf{T_N}$ time reference (see the description below) records for a given shipment quickly and easily. Here are some ways to do this:

1. Create a permanent or temporary table to hold the RailCarID, ShipDate, and four sets of date and time fields for each shipment. This could be called the Trip Metrics table. If you are using a relational database, this is the most efficient choice. The only drawback is that you will lose the access to the shipping information when you archive the shipments.
2. Add the four sets of date and time fields to the existing record layout in the Shipment Information table. This is a good choice if you are not using a powerful database (or if you archive the shipments often) because you have all the information you will need for generating trip reports in the one record.
3. Create a flag code field in the record layout of the existing Rail Car Movements table that can be used to identify a record using the $\mathbf{T_N}$ markers as flag code values. This uses less disk space than the previous two methods and it may be a valid option for non-relational database users. However, each time you want to calculate the time intervals for a given trip you will have to pass all the movement

records for the entire trip to collect the marked records. Also, you will lose your trip metrics when you archive the movements.

We are being a bit ambiguous here because not every company looks at the placement point the same way. There are three railroad sighting codes that should help us: **D** – Arrival at Destination, **Y** – Constructive Placement, and **Z** – Actual Placement. Unfortunately, the railroads do not always use these three sighting codes as defined, and sometimes not at all. So, you will have to decide how to logically mark your placements. You need to select which sighting codes to use as your primary placement point and then you will need to define a fall-back option for those times when the sighting codes you expected do not appear in the movement stream.

Once you get your placement logic figured out, the rest is easy. You have marked the beginning of the trip ($\mathbf{T_1}$), the placement at your customer ($\mathbf{T_2}$), the release from your customer ($\mathbf{T_3}$), and the return to origin $\mathbf{T_4}$). Now, watch what we can do with these numbers!

The calculations in the table on the next page are generated for each completed trip. Then, the individual trip calculations are analyzed to produce statistical values that will become a basis for estimating arrivals, releases, and returns. You may even wish to create an "after the fact" exception log. This is a separate report which lists the trips that: 1) appear to be outside of the normal transit times, 2) contain bad order or delay movement sequences, 3) have no shipping information, 4) have no Loaded-to-Empty swap, or 5) have only a portion of the movements that make up a complete trip.

In addition to the other calculations, the dead time is calculated. *Dead time* for a rail car is considered to be the idle time at your product loading point following the return from one shipment and prior to the next shipment. All of the calculations in the table use the date field to produce results expressed in units of time, specifically, DAYS. To increase the precision of the interval measurements, simply include the SightingTime field and express the results in HOURS and MINUTES.

The five trip interval metrics are for each shipment. To determine the average trip times for all trips, sum the individual shipment metrics and then divide by the number of shipments. Let's try an example. Suppose you look through the **Complete Trip** interval values for all of your shipments and you find 23 days here, 34 days there, 27 days somewhere else. In all, you have calculated values for 126 shipments. To determine the average trip time, add all of the **Complete Trip** values and then divide the sum by 126. Obviously, the fewer shipments you have, the less dependable this average will be. And vice versa.

Trip Analysis Calculations

Description of Calculation	Formula
Outbound Trip Segment - That portion of a complete trip which covers movements from the origin point to the placement at customer.	$T_2 - T_1$
Delay at Customer Trip Segment - That portion of a complete trip which covers the time between the placement at customer and the release from customer.	$T_3 - T_2$
Return Trip Segment - That portion of a complete trip which covers movements from the release from customer back to the origin point.	$T_4 - T_3$
Complete Trip - That portion of a complete trip which covers movements from the release from customer back to the origin point.	$T_4 - T_1$
Dead Time - The idle time between trips, specifically from the last return movement to the first outbound movement of the next trip.	$T1_4 - T2_1$

You could increase the accuracy of your trip measurements by applying some of the general rules of statistics (queuing theory) to the interval metrics. Mean, variance, standard deviation, and the first and second moment calculations could produce some highly reliable trip numbers. You could then use these values to help predict how long a rail car will take to get to a customer, when that customer will release the car, and how long it will take to get the car back to the origin. Almost any spreadsheet software package could be used to produce impressive charts that could become very helpful to your customer service folks, not to mention your fleet forecasters and budgeteers.

Since most companies ship to more than one customer and the customers are located at geographically random points, you need to group the shipments in a way that defines your traffic lanes. Then use the shipment interval metrics to form an accurate analysis of each lane. Just as a suggestion, using the Customer Name, Destination City and State, and Origin City and State as the group values, you can get a quick snapshot of your traffic lanes. You may want to drill down a little deeper, but this is a good place to start.

We are going to do a lot more with these trip metrics in other system extensions described below. But for now, here's a list of what you can do so far (And remember, we only had to write one additional program to mark the rail car movements.):

- Group the trip metrics by customer and run your statistical calculations to produce information about the **Delay at Customer**. You may decide that a customer who delays a rail car for an unusually long time should pay more for the shipment. Every additional day of delay by a customer costs you, if nothing else, at least the daily cost (leased or owned) of the rail car.
- Group the trip metrics by various time periods to determine if you are having any seasonal slowdowns. If the **Delay at Customer** is longer, you may need to deal with that customer. If either the **Outbound Trip** or the **Return Trip** increases seasonally, you will need to talk to the railroads.
- Group the trip metrics by product to see if there are any product-related handling problems showing up. If you see something unusual, you may have to dig deeper

to find out which railroad, which customer, or which product is responsible. You could write another program to mark unusual intervals between movements to help support your trip metric discoveries.

ETAs

Let's say the Trip Analysis extension produces some really solid information and your customer service people have come to depend on the trip charts you developed for them. Every month you deliver to them an updated chart with recalculated trip statistics, and every day you deliver a report showing the current locations of all rail cars listed by customer. You could do more.

You could use the actual trip statistics values to calculate an updated ETA on their daily "car locations by customer" report. In addition, since we have the trip metrics separated by trip segment, you could produce a variation of that same report for the traffic person responsible for getting the customer to release the car on time. And another variation for the production person who is responsible for scheduling cars for loading. And yet another variation for the rail maintenance person following cars to and from the car repair facilities.

First, we need a place to keep the ETA values. Create a table containing the fields shown in the box on the next page. We will call this the ETA Information table. As you can see, the ETA values are grouped according to the customer's name and location and the origin of the shipment. Since each shipment contains the same fields, we should be able to establish a relationship between the Rail Car Movements table and the ETA Information table using the Shipment Information table as the connecting link.
You need to add three more fields to the Shipment Information table:

1. **ETABase** - This field contains the date that marks the beginning of the ETA period. The ETABase is the ShipDate for outbound cars, the Placement at Destination date for cars located at the customer, and the Release date for returning cars.
2. **ETACode** - The ETACode field tells us the current segment of the trip. Values for this field might be: **O**utbound, **D**elay at Customer, or **R**eturn.
3. **NewETA** - This field contains the newly calculated (or updated) ETA date for the current trip segment.

ETA Information Table

Field Name	Description	Characteristics	Range
CustomerName	The name of the customer to whom the rail car is being sent.	XXXX...XXXX **k1** (Length as required.)	Defined by User
DestinationCity	The city where the rail car is going on the current leg of trip.	XXXXXXXXXX **k2**	One to ten characters representing city
DestinationState	The state where the rail car is going on the current leg of trip.	XX **k3**	Valid two-character state abbreviation
OriginCity	The city from which the rail car shipment originated.	XXXXXXXXXX **k4**	One to ten characters representing city
OriginState	The state from which the rail car shipment originated.	XX **k5**	Valid two-character state abbreviation
OutboundETA	Statistical interval representing the time it takes to travel from the origin to the destination.	####[.##]	Numerical value in units of days or hours/minutes.
DelayETA	Statistical interval representing the time the customer holds the car before releasing it to the carrier for the return trip.	####[.##]	Numerical value in units of days or hours/minutes.
ReturnETA	Statistical interval representing the time it takes to return to the origin from the destination.	####[.##]	Numerical value in units of days or hours/minutes.

It is not strictly necessary to add the three new fields to the Shipment Information table. If you prefer, you may create another new table called ETA Base. Since each rail car can have only one ETA value at a time, this table would contain the RailCarID as the key field. Then, add the three fields described above to that table. If you are trying to keep the number of tables down to a minimum, you could add these three fields to the Fleet Information table because it also uses the RailCarID as the key field and, therefore, has only one record per rail car.

You will need to write a program to calculate the NewETA value and place it into the Shipment Information table (or the ETA Base table). The new program should be the last process in the daily update series. To do this for each rail car, first locate the current rail car movement (the most recent record). Then, using the SightingDate in that record, calculate the NewETA as follows:

⇒ If this is a new shipment there will not yet be any ETA values, so you can save the ShipmentDate as the ETABase and place an **O** into the ETACode field. Use the customer, destination, and origin fields in the Shipping Information record to link to the ETA Information table and pick up the OutboundETA value. Calculate the NewETA date using this formula:

NewETA = Today + (OutboundETA – (SightingDate – ETABase))

⇒ If the location city and state equals the destination city and state, the car can be considered placed at destination. (Here's another reason why the city and state names must be spelled correctly and consistently!) Save the SightingDate as the ETABase and place a **D** in the ETACode field. Using the Shipment Information as the link to the ETA Information, pick up the DelayETA value and calculate the NewETA using this formula:

NewETA = Today + (DelayETA – (SightingDate – ETABase))

⇒ If the Loaded/Empty status of the current rail car movement has just changed to "E", the car is considered to be released from customer. Save the SightingDate as the ETABase and place an **R** in the ETACode field. Linking to the ETA Information as before, pick up the ReturnETA value and calculate the NewETA using this formula:

NewETA = Today + (ReturnETA – (SightingDate – ETABase))

⇒ If none of the above conditions has been met, the NewETA should be recalculated using the ETABase for the current trip segment. Link to the ETA Information table and pick up the appropriate ETA metric value based on the current trip segment (O/D/R). Then, calculate the NewETA using this formula:

NewETA = Today + (O/D/R-ETA – (SightingDate – ETABase))

Note: If, at any time, the result of **(O/D/R-ETA - (SightingDate - ETABase))** ***goes negative, the value of NewETA should be set to indicate that the ETA is worthless. A negative value here means that the rail car on this trip segment is so far behind schedule that an ETA cannot be calculated with any accuracy.***

The programming for this extension is tricky and you are going to have to put in all kinds of logical checks and balances to make sure the ETA is reliable, but it can be done. Once calculated, however, it can be placed on any report or screen that needs to show the estimated time to the next event. The method we used to calculate the ETA is not very sophisticated and you may be able to develop a formula that adjusts to unexpected situations like Bad Order and Held conditions or even just long delays at junctions. Experimenting is half the fun. Good luck! And if you come up with something great, let us know.

Route Analysis

An old Chinese proverb says, "When all things in the world seem beyond your understanding, first you must find one thing you can understand." Then again, maybe it was Key Luke in the "Kung Fu" television series who said it. Whoever. The bottom line is to build something meaningful, you've got to have a solid foundation first.

This is also true when you try to compare one thing to another. One of the things you are comparing has to be a known thing. For example, to answer the question "Is water wet?", first you need to know what "wet" means. (If you knew what "water" was, you wouldn't have asked the question in the first place.)

This may sound frivolous, but it does have a point relative to route analysis. *Route analysis* is the art of knowing when a rail car is <u>not</u> in the right place. It's a bit magical, but it is also within the realm believability for a traffic manager (a human being) to glance at a list of rail car movements and quickly determine which cars are in the right place and which are in the wrong place. The cars in the right place are immediately dismissed from thought, and all cognitive effort is directed toward correcting the location of the cars that are in the wrong place – this is management by exception. What we need to do is figure out how our traffic manager works this magic and then pass the intelligence and skill on to our computer program. This is not going to be easy, so we're going to do it in incremental steps.

STEP #1 - The first thing we need to do on the road to route analysis is create a solid foundation. To do this, we will use the Route field in the Shipment Information table to store a text description of the railroads and junctions encountered between the origin and the destination (and back) for this shipment. Most traffic managers have seen this "route shorthand" printed on their waybills. It looks something like this example:

CSXT-HAGTN-BN-BATH-CGNE

Translated, this means that a rail car is using the **CSXT** as the prime carrier from the origin to connect with the **BN** at **HAGTN** (Hagerstown, MD). The **BN** will then take the car to **BATH** (Bath, NY) where it will connect with the **CGNE** which will take the car on to the destination. The standard abbreviations for the railroads and junctions can be found in the **Official Railway Guide**.

Using the Route field we can see that our example shipment will go over three railroads: the CSXT, the BN, and the CGNE. Since most rail cars that go on round trips use reverse routing to get back to their origin, the railroads used on the return trip would be the CGNE, the BN, and finally the CSXT. So, we need to write a program (or add a process to the existing CLM Update program) that will verify the carriers listed in the Route field. Here's how it works: for every CLM record added to the Rail Car Movements table we need to access the appropriate Shipment Information record and pick up the Route field containing our waybill route description. Now, compare the Railroad Initials in the CLM to the carriers listed in the Route field. If there is a match great, if there is not a match, the rail car should be listed as an "off carrier" exception.

STEP #2 - We can increase the sophistication of the route analysis function by adding a new field (called CarrierList) to the records in our Shipment Information table. This field will be used to store only the railroad initials from the Route field. The CarrierList field would be set up when the shipment record is added, then, as each successive junction city is encountered in the sightings, the railroad initials for the preceding railroads would be dropped from the CarrierList field.

Using our example route above, the CarrierList field would initially be set to "**CSXT BN CGNE**" and any CLMs must be verified for the first carrier – **CSXT**. When we see the movement into Hagerstown, MD, we would remove **CSXT** from the CarrierList field, leaving the field value as "**BN CGNE**". Now, any moves for this shipment must be only on the **BN**. And so on. When the rail car switches from loaded to empty, the original value in the CarrierList field would be restored, but in reverse order: "**CGNE BN CSXT**". We handle the verifications of the railroad initials on the return trip in exactly the same way. This is still only an "off carrier" verification, but now we have reduced our list of acceptable carriers down to just one, so any variation at the carrier level from our route is caught immediately.

STEP #3 - This step takes us into knee-deep into the murky waters of route analysis. It also adds considerably to the complexity and maintenance of our rail system. First, you need to create a new table called the Route Information table. This table should look like the one on the following page.

Notice that we have added an Interval field. It will not be used until the next step. The CarrierList field we created in the Shipment Information table in the previous step should now be changed to RouteCode and its field type and length should be changed to match the RouteCode field in the Route Information table.

Route Information Table

Field Name	Description	Characteristics	Range
RouteCode	A unique code identifying this route.	XXXX...XXXX **k1**	As defined by the user
Movement#	A sequential number identifying the movement.	#### **k2**	Ascending number
Carrier	The AAR standard initials identifying the carrier responsible for the movement.	XXXX	One to four initials, see **Rail Users' Manual**
City	The city to which the movement was made.	XXXXXXXXXX	One to ten characters representing city
State	The state to which the movement was made.	XX	Valid two-character state abbreviation
L/EStatus	The loaded or empty status of a car on this movement.	X	The only possible codes are "L" or "E".
ActionsList	The expected railroad sighting code values for this movement.	XXX,XXX,XXX, XXX,XXX	Up to five different one to three character codes, see **Rail Users' Manual**
Interval	Interval representing the time it takes to complete this move, as a standard.	####[.##]	Numeric value in units of days or hours/minutes
Optional	A code value indicating an optional move.	X	The only possible codes are blank or "O"

After accumulating ten trips or more to a given customer (minimum for determining standards), you can create a set of expected movements from the origin to the destination (and back) organized by the RouteCode, a unique code value representing the trip. If there are variant moves, include them in the list, but mark them as optional.

As each shipment is entered into the Shipment Information table, the code identifying the route is put into the shipment's RouteCode field. Now, extend the program we used in Step #2 so that it will verify each incremental sighting against the list of moves in the Route Information table. Any movement not represented in the table should be logged as an "out of route" exception.

STEP #4 - This is simply the next logical extension to the program we created for Step #3. Using the Interval field in the Route Information table, add the standard travel time between each sighting to the records representing the moves for each route. Now, compare the actual movement time between the current sighting and the previous sighting against the standard time (the Interval field) and report each variance of, say, 20% or more (plus or minus) as a "[late/early] arrival" exception.

If you want to make the program a little more versatile, you set it up so that the percentage of variance is different for loaded cars (the outbound segment) and empty cars (the return segment). Also, the percentage of variance is totally up to you. If you prefer a tighter rein on your fleet, use 10% or even 5%.

Steps #3 and #4 are data entry intensive. There is no denying that. But they will give you a tremendous return on investment when you begin using the function. Plus, you will be able to see exactly how your cars perform on a lane-by-lane basis. If you have a problem in a lane, you will know where the problem is, how it affects your transit times and overall fleet utilization, and you will know whom to contact to get the problem fixed. In other words, once you begin to establish standards based on your current transit times, you can begin to fine-tune your traffic lanes to perfection.

STEP #5 - All the route analysis that we have done thus far has been on a *reactive* basis. That is, we have waited until a rail carrier has sent the next movement to us before we checked a rail car for an exception condition. If we want our route analysis to become more *proactive*, we need to create a procedure that checks each rail car in our fleet whether it moves or not. After all, using the Route Information table, we know where the rail cars are supposed to go and how long they are supposed to take to get there. Why wait for the carriers to tell us the car is late arriving?

To accomplish this, we need to add another program to the daily CLM process following the CLM Update program. This new program will check the current location of each car and compare the time interval between the current sighting and today's date to the Interval field on the next expected move in the Route Information table. Based on percentage values you set, if enough time has passed since the last movement, list the rail car as a "probable late arrival" exception.

It is important to consider the WORK VS REWARD ratio when dealing with route analysis. It is very easy to keep extending these programs so that you would flag every rail car that is more than an hour off schedule. However, you may be collecting CLMs only once a day which means you could be as much as 24 hours behind the actual movement of the cars. Yes, you could collect CLMs twice a day, three times a day, four times, or even once each hour. But, keep one eye on your goal and the other eye on the cost (money and time). The process must be a benefit, not a burden.

Load Scheduling and Analysis

This extension to the basic tracking system is most useful to the production people. You can add a program to the Daily Reports section of the automated daily events processing which uses the Origin/Destination Information table to help determine which rail cars have returned empty from their trips and can be made available for loading their next shipment.

If you want to provide for input from the production group, you could create a separate table where the list of these available cars would be maintained. The production manager could schedule a car for loading by marking the record for that car with a special "scheduled" code. When the car is actually loaded, it would be marked as "loaded". Then, when it is shipped, the appearance of the first movement in the Rail Car Movement table or the shipment record in the Shipment Information table would cause the car to be removed from the Available Cars table.

It would be a fairly simple matter to include enough information about the car in the Fleet Information table that the production manager could select a car for a particular shipment based on the length of the trip, the car's cost per month, and the car's mileage credit rate per loaded mile. It is a clever trick to let the accumulation of mileage credits pay a car's monthly lease or purchase costs. If you do it right, you could manage a fleet of rail cars that costs you nothing to operate or, possibly, even makes you money.

This Available Cars table could also contain track and spot numbers to locate the cars on your on-plant rails, if you have them. If you did not want to combine the information, you could create a separate table just for on-plant movements. Make the table available to several workstations around the plant and have your car movers update the "on-plant location" fields in the table. You could even automate this process so that all the car mover would have to do is enter a rail car's numbers (e.g., **001784**) and your program would pick up the date and time from the computer and the on-plant location from the location of the workstation. In fact, you could develop a complete rail sub-system that would spot your cars at various points on your plant and update the Rail Car Movements table on the fly. The cost of the sub-system could be rolled into the annual transportation budget on a "dollars per shipment" basis.

Now, let's talk a little bit about load analysis. One thing a lot of companies will do is load their available rail cars in anticipation of a shipment. This is done for several reasons: 1) it helps prevent peaks and valleys in production; 2) the product is ready for immediate shipment; 3) the company has determined that they can store their products in rail cars cheaper than in any other form of storage; 4) they have just fired up a new plant or product and the fixed storage containers are not yet ready; 5) the plant shuts down annually for one week to do inventory, cleaning, and maintenance; 6) the company maintains a set of loaded cars at a local terminal. Whatever the reason, you may be interested in how long the *early loadings* stay around the plant before they are actually shipped. To do this, just create a trip with "**EARLY LOAD**" in the Consignee, the date the car was loaded in the ShipmentDate, and "**EL**" in the RouteCode. Then, when the car is shipped (with a new shipment record containing the ShipDate of the actual shipment), the trip analysis will group the early loadings separately.

Another thing you can do with the load information is compare the volume shipped (from the shipment record) with the tare weight of the rail car (from the UMLER Information table – see the **UMLER** section below). The differences show the underloadings or overloadings (this is rare). You can create a report that does the calculations and produces the cost of underloading expressed in terms of dollars, extra cars required, or lost shipping days.

It is always interesting to see the looks on the production managers' faces when these facts are presented in a meeting. It's amazing how quickly a few pounds or gallons or tons here and there add up. Politically speaking, however, you may be wise to show your findings to the production managers before you drop this little bomb in front of your CEO. Nobody likes getting bushwhacked, and you may avoid getting a knife in your back somewhere down the line. Plus, you could form a very strong alliance with production if you give them your analysis and they are able to turn the situation around. They look good, you look good, and everybody wins.

Fleet Utilization and Forecasting

Once you have the positioning of the rail cars under control, it's time to start working on the efficiency of the operation. That's what fleet utilization is all about. However, determining the utilization of a fleet incorporates myriad measurements and we can only break the surface here. So we will not talk about a specific program to write or a table to create. Instead, we will just describe different projects you can do to get the measurements that mean the most to your company. Then, we'll get into forecasting.

Utilization by Count - Once you have established the basic tracking system, you can create a program or a report to do some simple counting for you. Based on the current sightings records in the Rail Car Movements table, you can summarize the following:

- Loaded Cars at Origin - These are the cars that have shipment records but which have not yet left the origin.
- Loaded Cars in Transit - These are the cars that have shipments and are moving, but they have not yet reached their destination.
- Loaded Cars at Destination - These are the cars with shipments that are currently located at the destination.
- Empties at Destination - These are the cars that have swapped to empty but they are still at the destination.
- Empties Returning - These are the empty cars that are moving, but they have not yet reached their origin.
- Empties at Origin - These are the empty cars that have completed their trips and are now available for load scheduling.

If you do this daily, you will begin to see the trends develop, especially if you keep a history. It's fairly easy to transfer this information into a spreadsheet program to create some nifty charts showing the daily, weekly, and monthly highs and lows. Look closely enough and you may be able to spot that problem you had last summer when you had so many empty cars you had to tell your carrier to hold off moving in the returning cars. And, there, see that dip on the graph, that's where you went for four days with no cars at all available for shipments.

Utilization by Comparison - Assuming that you have created a Route Information table and that you have populated the table with records containing standard intervals for your routes, you could easily begin comparing the actual performance to the standards expected. The best method of determining fleet utilization is sometimes considered to be the ratio of actuals to standards. When this ratio approaches a value of **1.0**, that is, the standard time divided by the actual time, your actual performance is close to what you expected or predicted it would be. If the ratio is less than 1.0, you need to accelerate the actual trip movements; if the ratio is greater than 1.0, you need to re-think the standard. Here are three ways of grouping the comparisons.

⇒ **Based on Delays** - On a daily basis, using the current sightings for each rail car, compare the number of days of non-movement (i.e., the current sighting date subtracted from today's date) to the interval value of the next expected movement for the associated route.

⇒ **Based on Carriers** - On a trip basis (usually done monthly), separate the various carriers and compare their performance to the standard intervals accumulated by carrier. This can provide some intriguing information which you can use as a focal point when you meet with your carrier representative. It can (and should) be used to support both good service ("A round of huzzahs and attaboys for everybody at BLAH railroad!") and bad service ("How can we work together to get this problem at Nowhere Junction solved?").

⇒ **Based on Trip** - As above, on a trip basis by month, you can break the trips down to their respective lanes and show their performance as percentages compared to the standards. Effectively, you are creating a numerical bell curve. Here's how that information might look on a report:

Monthly Trip Performance										05/31/95
Traffic Lane	**Trip Std**	**Way Early**	**-3**	**-2**	**-1**	**On Std**	**+1**	**+2**	**+3**	**Way Late**
RICHMOND, VA to ATLANTA, GA	13	1%	2%	6%	18%	**23%**	35%	10%	3%	2%
RICHMOND, VA to STGEORGE, AR	19	0%	6%	4%	11%	**17%**	27%	21%	7%	7%
RICHMOND, VA to WESTWATER, UT	26	0%	0%	7%	14%	**41%**	11%	19%	5%	3%

One last word on this subject: *utilization* and *efficiency* are <u>not</u> the same thing! It is possible to get 90% utilization out of your fleet and still be operating at an efficiency level of 10%. If you have never fine-tuned your traffic lanes, if you don't care how long a car takes to get to your customers, if you load and ship quickly only to divert in transit, all of your cars will be moving so your utilization will be high. But they are running without control, so your efficiency is low. In short, the definition of *utilization* is "working". The definition of *efficiency* is "working well".

Forecasting for the Masses - Enough about utilization, let's discuss forecasting. This is, perhaps, the toughest part of a fleet managers job and it makes your local TV weatherperson's job look like a piece of cake. It's hard enough to keep the fleet under control today, much less to try to predict what will be needed next month, next year, or the year after. Fortunately, we can help.

If you have completed the Trip Analysis extension, you have all the tools you need to tackle a decent forecast. Statistically speaking, you have an above average chance a producing workable numbers. Remember that old Yogi Berra line: "Ninety percent of [playing] baseball is half mental." Well, he was pretty close to the mark when it comes to describing rail forecasting: ninety percent of the effort you put into your forecast will produce half decent numbers. For each individual customer location, you need to develop solid statistical figures for certain trip elements. Combine these with valid forecast requirements and you can produce some very reliable fleet sizing values.

Here's how to do it. First, gather enough information from the forecast requirements and your RFMS trip metrics to calculate the following variables:

◊ The forecast delivery period (FDP). Usually provided as a schedule with begin and end dates or specified delivery arrival dates, the FDP is the number of days you have to complete the delivery of the product.

◊ The forecast delivered volume requirement (FDV). This is the total volume of product required by the customer during the forecast delivery period.

◊ The actual volume per trip (AVT). Take the volume of product you actually delivered during a previous calendar period similar to the FDP (last year, last quarter, or last month) and divide it by the number of <u>complete trips</u> in that time period. This yields the average volume of product carried by each rail shipment. Don't divide by the number of shipments! Remember, a completed shipment only covers a little over half of a completed trip.

◊ The actual trip time (ATT). Using your trip metrics for shipments to this customer within a similar previous calendar period, calculate the average time to complete a trip. This includes outbound time, delay at the customer, return time, and dead time. Also, don't forget to add in time for bad orders and delays. These calculations were described in the section on **Bad Order History**.

◊ The number of cars required (NCR). NCR is expressed in whole cars, rounded <u>up</u> (↑) or <u>down</u> (↓) to the closest integer value as you feel necessary. This will produce the number of cars required to complete the delivery of the product within the forecast period <u>to this one customer location</u>.

Here are our fleet forecasting formulas, listed separately:

⇒ Calculate the number of complete trips required to satisfy the delivery of the required product volume (TRPS)

TRPS↑ = FDV / AVT

⇒ Calculate the number of complete trips possible for each car within the forecast period (TPEC)

TPEC↓ = FDP / ATT

⇒ Calculate the number of cars required (i.e., Fleet Size)

NCR↑ = TRPS / TPEC

If you put it all together into one formula so that you can substitute values as required, you can solve for any individual unknown element. For example, if you have a fixed size fleet and want to calculate the number of days it would take to deliver a specified volume, you can bring the NCR value back onto the other side of the equation and solve for FDP. Here's what it would look like when you combine the previous calculations into one formula:

(FDV / AVT)↑ / (FDP / ATT)↓ = NCR↑

Rounding up and down as we have shown produces a very conservative fleet size estimate. Some people like this, others like to live closer to the edge. Obviously, you can not have part of a rail car traveling on a piece of a trip, but if you want a slightly tighter fleet size value you could hold the rounding until you have solved for NCR. If you are left with a fraction of a car, round up or negotiate a slight extension to the delivery period. Your option.

Once you have developed an NCR value for each of your customers, add 'em up. This sum is the total number of cars you will need in your fleet for the forecast time period. Most companies use a common forecast delivery period across all customers so that the total number of cars required is a simple sum. It is possible that you might have to make adjustments for one-time customers, seasonal products, emergency shipments, breakdowns, etc. This is why conservative numbers usually work best.

Repair and Maintenance

This extension to the basic project will be of considerable benefit to your rail car maintenance people. Of course, we are assuming that you have the maintenance responsibility for your cars. If you don't, go on to the next extension section.

For years now, most commercial rail car shop facilities have had the capability to supply their customers with the repair information electronically. Likewise, the rail carriers also send their repair records to the AAR. In a process similar to the CLMs, but not as widely used, the AAR collects maintenance records and provides them to interested (authorized) parties. This is called the Car Repair and Billing Information Service. You can receive electronic maintenance records on a monthly basis either by electronic connection or mail from RAILINC headquartered in Cary, NC (www.railinc.com).

Using the repair information, you could create a program that would update a maintenance history table containing information describing every aspect of the repairs made to your rail cars. Repair descriptions, wheel readings, parts removed and replaced, cost details, everything. It's all in there. Store it in your computer and you can have records that will support your maintenance invoices from the repair facilities. This is powerful information. It does for your maintenance group what the CLMs do for your rail trackers.

Once you have the maintenance information at your fingertips, you can generate a host of reports and queries. How many repairs on CSXT in 1999? How many brake shoes were replaced by ATSF? What are the average cost and the frequency of occurrence of repairs to your aluminum cars as opposed to your steel cars? What's the mean-time-between-failure of your cars that are older than five years?

After the basic maintenance function has been added, you could begin work on a "shop trips" function. True, you could just add a shop trip into the Shipment Information table as we suggested earlier, but there are subtle differences between trips to customers and shop trips. For example, the outbound trip to the shop is the same, but you might want to create a special ETA for notifying you when the repaired car is expected to be back in service. You could also create a process to watch for the first movement away from the shop so you will know when the car becomes available. Just like everything else we have done, let your needs percolate in your imagination. Creativity rules!

UMLER

"UMLER" stands for Universal Machine Language Equipment Register. This is a computerized file of all the rail cars, highway trailers, and containers used in interchange and commercial service. It is maintained by the Customer Service Division of the Association of American Railroads. UMLER is the official source of rate and descriptive information (height, width, tare weight, couplers, linings, etc.). The AAR provides programs to equipment owners for editing, updating, and accessing the information.

That's the party line, but what good does it do you. Truthfully, not everybody needs to keep a full set of UMLER information on their cars, however, it is often helpful to have access to specific pieces of information. Let's try a few examples. Suppose you are building a new loading facility for one of your plants and you need to know the maximum height and width of every car that will come through the facility. Look at UMLER. Suppose you are installing a weighing station and you need to know the gross, net, and tare weights for all of your cars. Look at UMLER. Suppose you want to calculate the possible mileage credit earnings for a rail car because you want to send the car on the trip that will earn enough to offset the monthly cost of the car. Look at UMLER. Suppose you want to calculate the maximum load capacity by volume or by weight for each of your cars. Look at UMLER.

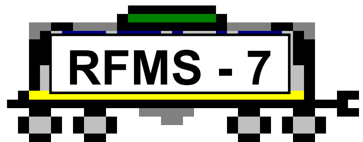
You can get the information from the AAR via RAILINC in many different transfer forms (magnetic tape, diskette, EDI, etc.). Once you get it, we recommend that you decide what pieces of the whole UMLER pie you will need, and then ignore the rest. We do not recommend that you create your own UMLER edit and update program. We have looked at it and there's just too much to do. Plus, changes are made to the specification ranges and validity tests about three times each year. Just keeping up with the changes would quickly become very expensive.

Since rail cars are subject to scheduled tests of various kinds (safety inspections, tank testing, etc.), you may want to create a companion function to the UMLER process for scheduling the tests and recording the results. This would be a fairly simple program.

You will need to define the required fields:

1. RailCarID (Of course).
2. Date fields for recording scheduled, rescheduled, expected, and completed dates.
3. Codes used to identify the tests or inspection events.
4. Codes and text fields where you can describe the tests.
5. Codes and text fields where you can describe the results.
6. Codes and fields used to describe each railcar as required for tests, inspections, and for HM-201 compliance.

Don't forget to provide a flagging mechanism for the exception process so you will know when a car is placed into inactive status and then when it is returned to full availability.



7. Summary

There is nothing left now but to do it. You can start with a few informal meetings to see how much interest is out there. You can knock on the door of your Information Systems group to see if you can get some early assistance. And, you can begin the research phase where you call the industry specialists in the list at the end of this chapter to see what they can provide and how much they will charge you.

Maybe it's the nature of the business, maybe it's that it is still early in the growth cycle, or maybe it's because when it comes to computer stuff everything changes so quickly that everybody understands we have to stick together simply to stay even with the computer beast. Whatever! You will discover that this is a very refreshing industry. Most of the people in this list either know or know of each other and, more importantly, they are aware of their own strengths and weaknesses. It is not at all uncommon for one company to recommend another just to help you out. And every one of these vendors loves to talk.

When you contact the vendors, let them know that you are just getting started. If you have read this entire booklet, you already know enough to answer most of their questions and ask a few intelligent questions of your own. Tell the vendors about this primer and let them know you have read it. (If they don't already know about it, refer them to us. They may be interested in reading it themselves.) You can use it as a common reference point from which to begin the discussions about your own system design. They may also have some very different opinions and recommendations. Good! What a dull world this would be if we all agreed on everything!

Options and Alternatives

There are, of course, other ways to approach rail fleet management. You do not have to build your own system from scratch. In fact, there are some pretty terrific alternatives out there and we encourage you to look around. "Rolling your own" RFMS is NOT always the best way. We went into considerable detail in the chapters preceding this one so that you would know what you are facing. If, after reading this booklet, you are comfortable with your skills and ready to tackle the job yourself, then good. We did our job and you now have enough information to start construction. If, on the other hand, you have come to realize that creating even the basic RFMS is too big a project for you, then great. We did our job exactly right! No truer words were ever spoken than Clint Eastwood's character in one of the ***Dirty Harry*** movies: "A man's got to know his limitations."

Let's examine a few of the alternatives to designing and programming your own system. Read through all of these options, even if you are sure of your preferences. Anytime you deal with a vendor company, it's a lot like getting married: you want someone you can talk to, you want someone you can live with, and you want someone who will always be there for you. Also, like marriage, working with a vendor is a give and take proposition. The vendor is going to give you the right to use his system, his support, and his expertise in the industry. And for that, you are going to give him money. Just remember, the best marriages work on a 50/50 partnership basis. In general, here are the alternatives to creating your own rail system:

- **PACKAGED SYSTEMS** – A packaged system is a pre-designed and pre-written software system that will provide you with a semi-instant solution to your rail fleet management problems. There are several good packaged systems on the market for all computer hardware persuasions. Your biggest problem will be in choosing the one that provides exactly what you want. You'll have to consider the following things:

a) The type of computer equipment the package requires – Will you have to purchase new equipment or upgrade your existing computer system? Will you have to change platforms (operating systems or interface programs)? How will it handle your network? How will remote users access the system? Will you need any additional software or hardware to support it?

b) How it will fit into your environment – Will your users have to be retrained on the equipment before they even start learning the new RFMS? Can the new package accept input from your existing systems? Who will do the interfacing?

c) The installation – How long will it take to get it up and running? Who is responsible for the installation? What about the connections to an existing system for external data (shipments, on-plant movements, etc.)?

d) The support – Will the package vendor be there when you need help? Does the vendor have the rail experience *and* the computer experience to give you the support you really need? Will the vendor help change the system to suit your needs? How good is the documentation? What about training? Can you change things yourself? What about updates or new functions? What about new versions?

e) The cost – Have you done enough research to know what you are getting? Is what you are getting worth what you are paying? Is everybody happy?

f) The future – How long will it be before you outgrow this system? Is the vendor keeping up with advances in computer technology? How important are you to the vendor's future? Have you calculated the COST VS REWARD?

- **MANAGEMENT SERVICES** – Hiring a vendor to handle your rail management is often the perfect way to go. Current corporate outsourcing trends require that you eliminate as much fat as possible from your environment. To do that, new equipment is not purchased, new systems are not designed, new people are not added to the staff, and sometimes even old staff members are allowed to leave. This can leave you with no other option but to "go outside" for services. But it may not be that way. You may have the perfect corporate environment, but maybe you don't have the expertise or the manpower to handle the rail management yourself. So, you turn to a service vendor for assistance and, for a price, they will provide you with what you need.

You can get just about everything you could possible want from a vendor company. From CLM collection to car brokerage, from traffic lane analysis to hazardous material handling, there are companies who can support you well. As before, there are things you should consider:

a) The credentials – Can the vendor do the job? Has the vendor done this work before? Have you seen the results? Have you talked with other clients? Will the vendor provide you with references?

b) The essentials – Will the vendor give you what you need? Will he provide related information upon request? Will he represent you at meetings with railroads or car brokers? Can your other departments work directly with the vendor or must everything go through you? Do you have the time? Do they work on-site or out of their offices?

c) The cost – Are you getting what you are paying for? Is the cost justifiable? Is this the best use of your money?

d) The future – Is this what you want to do? Is the relationship forever, or will the vendor help get you where you want to be?

- **COMBINATION SERVICES** – This is an interesting alternative that has appeared in the last few years. With the advent of powerful personal computers, rail-oriented personal computer systems, remote access, and the Internet, vendor companies have expanded the possibilities to include delicious, enticing combination platters of the two entrées we just described. Now, you can have someone else manage your entire rail fleet (as in the Management Services option) and still be able to have direct access and control over the data from your own computers (as in the Packaged Systems option). Sometimes, this option is the least expensive and best way to get into rail management.

To decide if this is the best approach for your company, you will have to combine the lists of considerations from the two previous options and answer the questions yourself. Just remember, you may be marrying two vendors at once: the service provider and the software (package) provider. It could work well; if your hardware entry requirements are kept to a minimum, your corporate responsibilities and involvement are kept to a minimum, your service charges seem reasonable, and the length of your relationship is kept to a minimum until you are comfortable with the relationship, it could be the perfect option. The trick is to decide where you want to be in three to five years. And, then keep your sights set on that goal. Any deviation should be your choice, not the vendors.

- **STATUS QUO** – Don't change a thing. This is an interesting option, but one you should consider as a plausible alternative. If none of the options we have presented, including creating your own system, suit your needs, then by all means stay where you are. The two simple tests we have emphasized throughout this booklet are: WORK VS REWARD and COST VS REWARD. If you put those into one formula, it would become (WORK + COST) VS REWARD. It's as simple as debits-on-the-left and credits-on-the-right. In any situation, doing nothing is still an option. The correctness of that option is for you to determine.

Problems, Problems, Problems

Okay, after a hundred and fifty-plus pages of this stuff, you may be saying to yourself, "My company doesn't work this way. We don't own or lease rail cars. We only care about cars in Arkansas. My rail cars only roll backwards." Relax. You're probably right. But, you can find some elements that fit. Use those as a basis and then go forward from there. Remember the Chinese proverb from the previous chapter? It's true. Do some research on your own.

First you think about it. Then you get other people to think about it. Then you all think about it together. Go into a room where you can draw charts and diagrams. Figure out what your company does now. Then decide what you want to do. Draw it up, write it down. Create a visual flow. The orders create the shipments that create the trips that define the routes. Who uses the information? Who needs the information? Do they need reports? Displays? Do they need it at all? Do they use it now? Would they use it if they could get it? Why? How? Why are those computer people so slippery? Did anybody tell them the meeting was today?

Nobody said this was going to be easy. Once you have the flow of information, both current and desired, written down, put some time schedules on it. Then put some cost figures on that. Now, go over it all again. Is it worth it? How about the WORK VS REWARD ratio? How about the COST VS REWARD ratio? Do you have the people? Do you have the time? Do you have the money? Do you have the need? Do you even care?

Create a checklist of your requirements. Call some of the vendors listed in the next section and see how they can help. If there is sufficient time available, corporate management may choose to begin their "rail experiment" by using an existing package or service that is both inexpensive to acquire and easy to implement just to get the information flowing. Then they use their experience as a gauge. They have seen what that package or service can do and they know that now they also want to do this, this, and this. You will undoubtedly go through the "build or buy" conundrum. And, only you can decide whether it is better to build it yourself, purchase an off-the-shelf solution, or link up with a service provider. **DO NOT LET THE VENDORS MAKE THIS DECISION FOR YOU!** They may try. After all, that's how sales people sell things. But, don't let it happen. Stay in control! You now know more than you did when you started, and that's probably enough to make a well informed decision.

Since the first publication of this primer, we have often been asked why we would give away all of our secrets. After all, rail tracking has been our business for over twenty years, why would we want to tell everybody how to do it. While we understand this point of view (usually held by the people who want to sell our services, by the way), we take the opposite view.

First, anything that helps make you a better rail manager is a good thing. Your job is tough enough as it is. Second, if you read this primer you will begin to appreciate the effort that is required to put together even a simple tracking system. And, if you appreciate that effort, you will understand the value of our ***Spotlight*** system or any of the other quality rail systems that are currently available.

There is a ton of work that goes into creating a good rail tracking system. Hopefully, this primer has given you an understanding of what is involved. If you want to build your own system, go for it! If you want help, call us. We will be happy to get you on the right "track".

Have a good day!

About the Author

McKamie Wilson is the president of CEVA Inc. The CEVA Operations Center manages and supports the ***Spotlight*** Rail Tracking service.

Specializing in rail logistics systems, Mac has provided network and database solutions to corporations throughout the United States and Canada for more than 25 years. He has spoken at industry conferences and university seminars on computerized rail management and related technical topics. As the senior applications designer and developer for NetCenter Corporation, he created the ***Spotlight*** Rail Tracking service. Mac also developed the **RaiLogix** Tracking System for Logistics Management Solutions in St. Louis, Missouri, the first **FleetWatch** Tracking System for AllTranstek in Downers Grove, Illinois, and the original **TRACE**-series of rail fleet management systems owned and marketed by MIS Traffic Services in Atlanta, Georgia. He has also developed three other major rail management systems for private corporations. He is fond of recalling how in 1981 he got his introduction into the rail industry: "I was just sitting in my office, minding my own business, when a client walked in wanting help organizing his rail information. Three months later we had a rail system up and running on an AppleII computer. My life hasn't been the same since!"

Mac and his wife, Connie, live in Midlothian, a 290-year old village just west of Richmond, Virginia. Midlothian is the site of Virginia's first railroad. In his moments of free time, Mac is an amateur photographer and an avid trail walker.

Contact Mac at mwilson@ce-va.com .

This version of

Rail Fleet Management Systems

A Rail Shipper's Primer

is produced by

CEVA

13354 Midlothian Turnpike
Suite B-102
Midlothian, Virginia 23113
804-378-2095