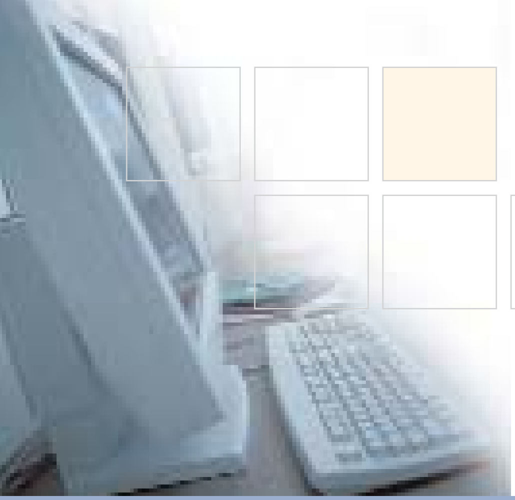

PROGRESS SOFTWARE | **Real Time** Division

Seven Principles of Effective RFID Data Management

TECHNICAL PRIMER

By Mark Palmer, mpalmer@progress.com

Though the recent publicity might suggest differently, Radio-Frequency Identification technology — RFID — is not new. RFID first appeared during World War II when Allied aircraft carried transponders that would acknowledge radar interrogations from friendly aircraft. Since then, both the size and cost of RFID tags have followed the progression of Moore's Law to where it's now feasible to attach RFID tags to a package of razor blades. But in fact, RFID has been used in commercial applications for years, from automated toll systems like E-Z Pass to automated manufacturing facilities. But in 2003, triggered by mandates from Wal-Mart and others, a media feeding frenzy ensued. Such mandates have validated that RFID just might enable a new era of business optimization where a Proctor and Gamble can instantly know its inventory stock levels, a Gillette can eliminate razor blade theft, and a Wal-Mart can squeeze even more cost from its supply chain by reducing the labor associated with bar-code scanning. This value, accumulated millions of times daily, will add up to billions. And that's just the beginning: visionaries see RFID impacting anything that involves physical items where keeping track has value: baggage tracked by airlines to improve security; Alzheimer's patients monitored in real time; pharmaceutical shipments to curtail counterfeiting.

But every opportunity carries its challenge, and there are many challenges posed by RFID. One of the biggest hills to climb is dealing with the flood of data RFID generates: in-store RFID implementation at Wal-Mart has the potential to generate as much data in three days as is contained in the entire U.S. Library of Congress (based on estimates from analyst Jim Crawford from Retail Forward). And it's not just a problem for companies the size of Wal-Mart — even modest RFID deployments will generate gigabytes of rapidly changing data a day. The volume and velocity of RFID data will exceed the capacity of existing technology infrastructure.

Capturing large volumes of data at high velocity is just the first step. Woody Allen once said: "I took a course in speed reading and was able to read *War and Peace* in 20 minutes. It's about Russia." If all you learn from massive volumes of RFID data are the most general conclusions then the value of that data is lost. It is as if one looked at a 250 gigabyte disk drive filled with useful information and concluded simply that it contained 10,424 files. If all you do is capture RFID events, then most of the value is lost because the advantage of RFID is real time knowledge, not data collection.

Instant decision-making on high-volume, high-velocity data streams is not a new challenge. It is the same problem found in the program trading systems of large investment banks, the command and control applications used by the military, and the network management applications in telecommunications. Based on experience in those industries, as well as RFID system development, I pose seven principles to help you effectively manage your RFID data.

Principle #1: Digest the Data Close to the Source

You could water your lawn pretty quickly if you used the fire hydrant on your street. But hooking your garden hose directly to the hydrant will only serve to get you a hose blown apart, a big pool of water and a very angry fire chief. Try to deploy an RFID system by directly connecting RFID readers to your central IT systems and the results will be similarly disastrous.

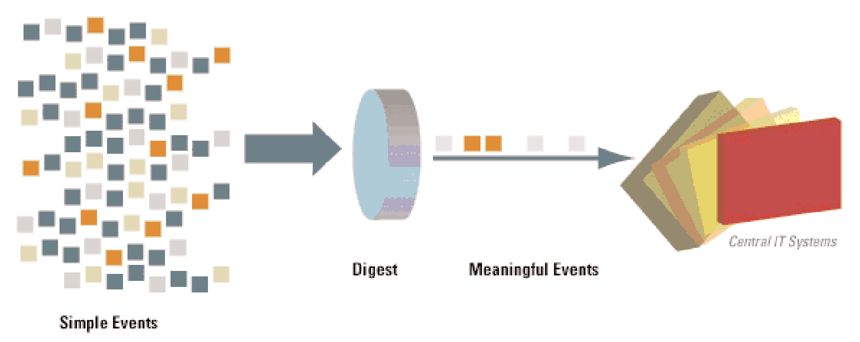

Figure 1: Digest your RFID event traffic at the "edge" of the enterprise.

A better approach is to digest your RFID event traffic close to the source — at the "edge" of the enterprise — and forward only the meaningful events to central IT systems. This digestion process is more than basic filtering — it's data cleansing, consolidation, and summarization; it's exception handling of many types — automated and human-made; it's compensation for unreliable technical infrastructure — application, hardware, network failure; it's adjustment for unreliable RFID tag-reader environments. Digestion of data close to the source allows this complex processing to occur, and exceptions to be handled locally thus protecting central IT systems from the flood of data.

Digest RFID event data close to the source of RFID activity to ensure greater reliability and protect your IT infrastructure.

Principle #2: Turns Simple Events into Meaningful Events

Complex Event Processing (CEP) is a new field that deals with the task of processing multiple streams of simple events with the goal of identifying the meaningful events within those streams. Examples of simple events include a church bell ringing, the appearance of a man in a tuxedo, and rice flying through the air. A complex event is what one infers from the simple events: a wedding is happening. CEP helps discover complex, inferred events by analyzing other events: the bells, the man in a tux, and the rice flying through the air.

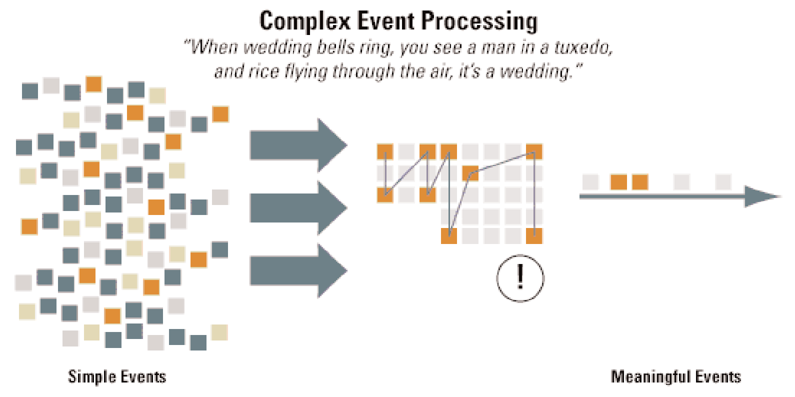
Complex Event Processing
"When wedding bells ring, you see a man in a tuxedo, and rice flying through the air, it's a wedding."
!
Simple Events
Meaningful Events

Figure 2: A simple example of how CEP works. It gathers many simple events and infers a complex event is occurring.

CEP has been pioneered by David Luckham of Stanford University, the author of *The Power of Events,* an Introduction to Complex Event Processing in Distributed Enterprise Systems. In his book, Professor Luckham defines a complex event query language that treats event time and order, in addition to event data, as the basic elements of data processing. According to Gartner, CEP will become a common computing model within five to ten years. But developers aren't sitting still — you can build CEP systems today in languages like Java or C++.

Whether you use a CEP tool or build your own, principle #2 is: turn simple events into meaningful events to derive actionable knowledge from discreet events.

Principle #3: Buffer Event Streams

You could water your lawn using water from a hydrant if you could step down the pressure and flow. One way to deal with the gush of RFID data is to develop RFID data "concentrators" that help buffer the flow of RFID event streams.

An RFID data concentrator is software that collects and processes raw RFID event flows close to the source of the data (principle 1). There are three primary elements: RFID middleware, event processing, and an in-memory data cache.

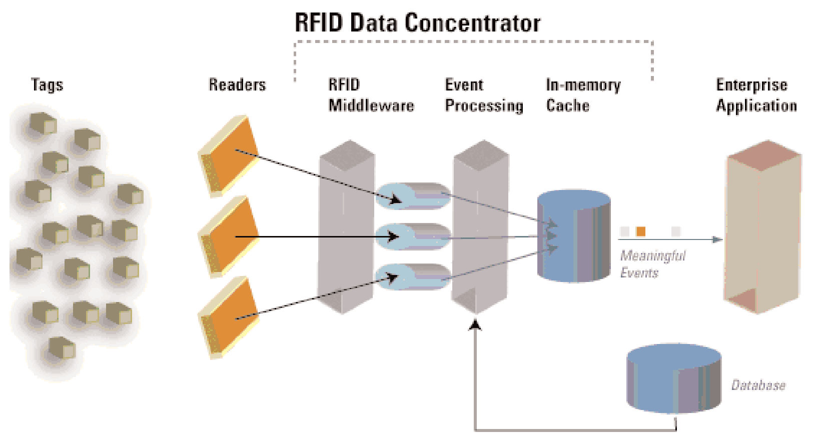

Figure 3: An RFID data concentrator is software that collects and processes raw RFID event flows close to the source of the data.

RFID middleware provides the interface for applications to receive events from RFID readers. There are many commercial implementations of RFID middleware available today. Additionally, the EPCglobal standards organization (EPCglobal is organized by EAN International and the Uniform Code Council (UCC) to drive the global adoption and implementation of the EPCglobal Network) is in the process of defining a standard for RFID middleware, the ALE (Application Level Event) standard, which defines a reader-neutral interface for receiving events from RFID readers.

Event processing handles high-volume, high-performance flows of events by organizing raw events into pipelines. Pipelining is a concept found in hardware and software systems of many types, including the central processing units (CPUs) in your computer, in software designed for handling stock market feeds in real time, and transaction processing systems that your credit card company uses. Pipelines allow the events to be categorized, then processes those categories with a set of simple tasks, as each thread gets a slice of processing time from the CPU. By performing a large number of operations on data in small "bursts," overall throughput is increased, and the average speed that any individual event can be processed is increased, as well.

Finally, an in-memory database, or data cache, makes the concentrator work in real time. In-memory data management techniques are crucial to accommodate the real-time nature of RFID. It's a basic law of physics: memory is 1,000 times faster than disk. Physics is the reason why MIT's Auto-ID center included a "Real Time In-Memory Event Database," or REID, as part of its first RFID reference architecture. Similarly, Stanford's CEP research has employed an "in-core-memory" database. While this cache provides much the same reliability, availability, and fault tolerance of a database, the distinction of calling this element a "cache" is to distinguish it from the enterprise application database, shown at right in the data concentrator diagram.

Employ data concentrators that buffer event stream flows by combining RFID middleware, event processing, and an in-memory data cache to achieve the reliable speed you need.

Principle #4: Cache Context

Most RFID data is simple. Unless you're using sophisticated, expensive tags, all you get is an identification number for the item, a time, and a location. Determining complex events from simple RFID event data requires context. And context typically comes from "reference" data, in a variety of forms.

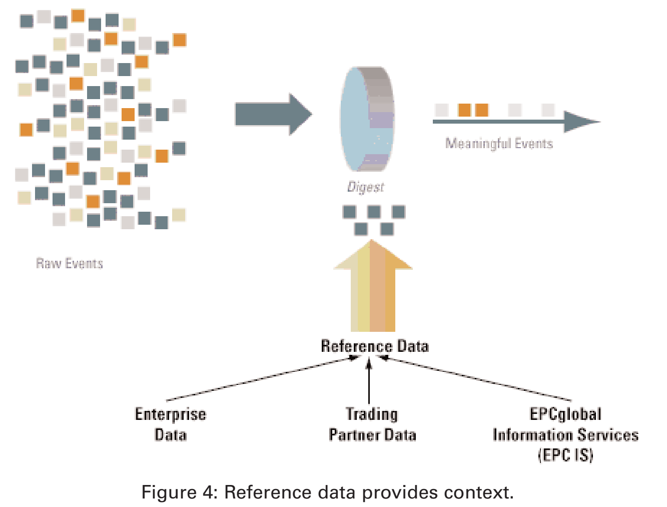

Figure 4: Reference data provides context.

For example, context can come from information in an advanced shipping notice (ASN). As provided by a manufacturing plant, the ASN can be used to confirm that tagged items sent by the manufacturer were actually received. Context may also come from an EPC Information Services directory (EPC IS). EPCglobal is defining standards to allow the EPC IS to serve as the framework for trading partners to exchange detailed product information. The EPC IS allows anyone with proper authorization in Kellogg's supply chain to learn whether the EPC tag 01.0000A89.00016F.000169DC0 is a 24 ounce box of Kellogg's Corn Flakes or a Gillette Mach3 Turbo razor. In addition the EPC IS will allow you to determine where it came from, where it's going, when it was produced, and so on. Reference data may also come from internal enterprise systems. For example, RFID-enabled baggage handling systems use data from passenger information systems to ensure that RFID-tagged bags get on the same plane you do.

Just as principle 3 leverages in-memory data caching for event data, context data needs the same approach. EPCGlobal's next generation (Gen2) standard of RFID readers specifies read rates of 1,800 RFID tags a second, which means a distribution center with 20 readers could generate 36,000 events a second at peak rates. Adding 36,000 SQL requests to your existing warehouse database probably isn't feasible, so it's best to replicate, and cache this data in your data concentrator such that is available in real-time.

By caching reference data, your data concentrator can process RFID event data, in context.

Principle #5: Federate Data Distribution in Near Real Time

Principle #1 advocates processing data close to the source to manage complexity and protect central IT systems. But the items you're tracking won't stay put for long. For example, an RFID baggage tracking system must distribute data about your bags to your destination airport far in advance of your arrival. And since most operations are distributed, you must plan to distribute RFID data, in near real time.

The definition of federated, from Merriam-Webster, is: "united in an alliance." Federating data is hardly new — trading systems federate data daily among different financial centers and do it in near real-time. For RFID, federated data distribution unites the RFID data concentrators in an alliance — whereby meaningful RFID events and context data is shared among members. A reliable, distributed middleware fabric facilitates data federation, integrated with the data caches at each concentrator, thus enabling meaningful events to be distributed among concentrators.

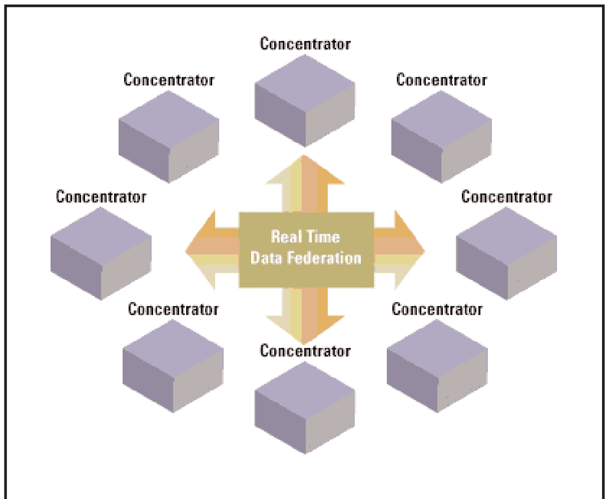

Figure 5: Federation of Real-Time Data to RFID Concentrator

The implementation details of this near real time distribution fabric is beyond the scope of this paper, but the principles are the same as we've described: process data close to the source, utilize data concentrators, cache data in concentrators, filter and handle exceptions. Ultimately, one must forward meaningful events to members of a distributed "alliance" to enable as close to a real-time view of the system as possible.

Federate data distribution so your RFID system can scale globally and as close to real time as possible.

Principle #6: Age RFID Data Gracefully

Even if you're Wal-Mart, you probably won't be adding 7 terabytes of disk capacity daily to accommodate your RFID data. By continuously aging RFID event data, you can reduce your working set, augment event data appropriately, and reduce load on down-stream systems, all at the same time.

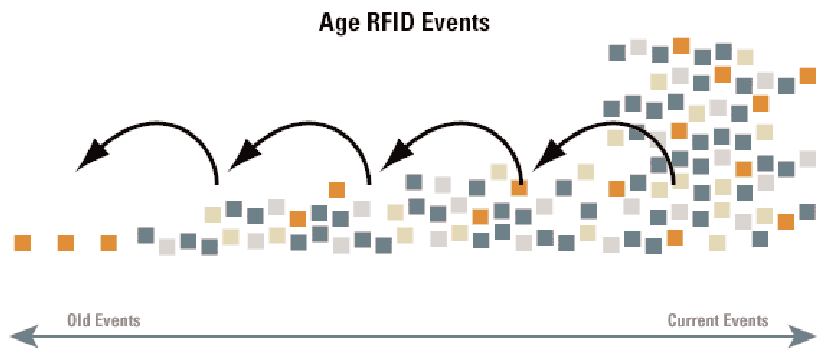

Figure 6: Age Data to Keep Working Set Manageable.

An airline baggage handling system must track events from the gate to the plane, but they don't all have to be stored forever. Yet RFID-enabled systems must permit operators to see the end-to-end baggage flow. The value is in understanding the baggage origin, how long it was on the conveyor belt, where it was handled, and how long it was handled. Furthermore, the basic flow of information will need augmentation — for example, at what time was the bag loaded into the aircraft and which cargo bay was it placed in? Finally, performance and scalability typically require that the storage of these events be optimized as the system runs. The RFID concentrator can delete events that are superfluous (e.g., redundant reads of baggage), supplement events that require context (e.g., associate a cargo bay number to a "bag loaded" event), siphon data off to other systems (e.g., save all baggage events for security subsequent security audits), and prepare event data to be distributed to other airports. At every step of the aging process, the data concentrator's cache (principle 3) is kept transactionally consistent to provide system recoverability.

Age RFID data to keep your working set of data manageable, enrich raw data with required context, and reduce the load on down-stream systems.

Principle #7: Automate Exception Handling

Exception handling is the primary job of any RFID system. Exceptions include our bride at our wedding event who gets cold feet and doesn't show up at the church, or when an airline baggage handling system needs to take a bag off the plane that's already been loaded. In the world of today, people handle these exceptions. The maid of honor searches for the bride; the baggage handlers search for and retrieve bags by hand. In RFID-enabled systems of the future, your software must both detect exceptions and then automate the handling of those exceptions.

Though detection of events is done by complex event processing, exception handling requires knowledge of what happened leading up to this event. That operation requires an algorithmic approach to corrective action. The combination of CEP and event replay is what makes exception handling possible.

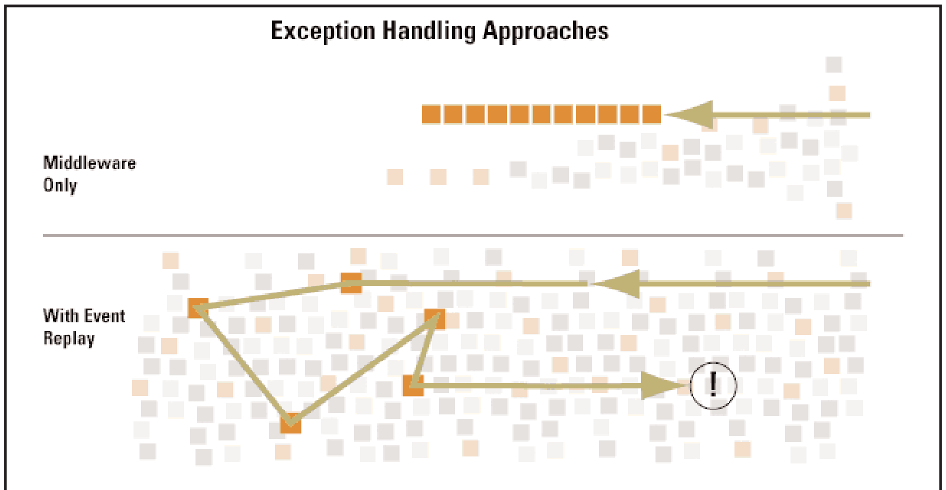

Figure 6: With CEP and event replay, exception handling is possible.

Event replay requires the storage of RFID events in a data cache that operates much like a TiVo for your RFID data. Just as TiVo has revolutionized the way we watch television, with flexible archival and random playback capability, RFID event replay can help automate exception handling. When an exceptional event occurs (e.g., bag must be reloaded), the data cache can replay the events for a particular item to figure out where it was last seen, which operator is physically closest to the item now, and then send a message, for example, to that person's pager.

Without event replay, the RFID system has no ability to go back in time to discover when a bag was loaded, which section of the plane it was loaded into, and what to do to correct the exception.

Automate exception handling to improve overall business efficiency — our final principle of effective RFID data management.

Pulling it Together

Mature RFID deployments will change the face of enterprise IT by making real time data processing commonplace. This shift in computing style requires data management near the source of the data, turning simple events into meaningful events, buffering of event streams, caching reference data to discover context, data federation in near real time, graceful data aging, and automated exception handling. These principles ensure reliable, real time, accurate decisions so that you're ready to tackle the challenges, and exploit the opportunities, that RFID presents. RFID may be relatively new to the "masses," but there are current day examples to illustrate how to be successful — and principles that can guide the way.

Related Resources

Progress Real Time Division RFID Resource Center

http://www.progress.com/realtime/rfid_resources/index.ssp

Look here for recorded webcasts, articles, white papers, and more on RFID.

Progress RFID Accelerator

http://www.progress.com/realtime/products/rfid/index.ssp

The Progress RFID solution delivers an in-memory data caching architecture and persistent data store that capture RFID event streams, thus enabling event-based queries and other processing operations to transform RFID data into business information that can be leveraged by supply chain, retail, manufacturing, and other enterprise applications.

Complex Event Processing Web Site

http://www.complexevents.com/

A CEP web site devoted to documenting applications and products that employ any form of CEP, and the latest research and developments in event processing.

About Progress Real Time Division

The Progress Real Time Division provides event stream processing, data management, data access, and synchronization products to enable the real-time enterprise. Our products monitor and analyze real-time event stream data for applications such as algorithmic trading and RFID; accelerate the performance of existing databases through sophisticated caching; manage and process complex data in the industry's leading object database; and support occasionally connected or mobile users requiring real-time access to enterprise applications. Progress Real Time is an operating unit of Progress Software Corporation (Nasdaq: PRGS), a global software industry leader. Headquartered in Bedford, Mass., Progress Real Time can be reached at http://www.progress.com/realtime or 781-280-4000.

TECHNICAL PRIMER

Mark Palmer is vice president of Event Stream Processing (ESP) for the Progress Software Real Time Division, with responsibility for product development, product management, product marketing, and business development for Progress Software's ESP business. Mark has over 15 years of experience in technology leadership as a CIO and CTO, is published widely, and speaks on technology topics that include Event Stream Processing, radio frequency identification (RFID), and large-scale distributed systems.

mark.palmer@progress.com

TECHNICAL PRIMER

Real Time Division

Worldwide and North American Headquarters
Progress Real Time Division, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280 4000

UK and Northern Ireland
Progress Real Time Division, 210 Bath Road, Slough, Berkshire, SL1 3XE England Tel: +44 1753 216 300

Central Europe
Progress Real Time Division, Konrad-Adenauer-Str. 13, 50996 Köln, Germany Tel: +49 6171 981 127

France
Progress Real Time Division, 3 Place de Saverne, Les Renardières B, 92901 Paris la Défense Tel: +33 1 41 16 16 56

www.progress.com