ICL Partner

ISS45 Internal Architecture

MB-ISS45-1022 Issue 4
February 19, 1997

As we have stressed again and again, **ISS*45*** was designed as a ***system***. All the key foundations—data integrity, security, performance, how tools access the system, and those three ***R***s, Resiliency, Replication and Recovery—were all worked out and elegantly designed before coding began on the feature set. This is why the features are integrated so well, why their operation is so consistent and flexible, and why the system capacity and performance is so impressive.

This bulletin discusses some of the internal workings of **ISS*45***. The information is offered to provide a deeper understanding of this powerful software system.

INTRODUCTION

The core software technology and architecture in **ISS*45*** has been developed and enhanced over a period of twelve years. It has been internationally successful on several continents over a wide range of retail systems ranging from small stores with few POS terminals to those with 100 lanes and more. Within this range of store sizes there is also a variety of applications covering high volume scanning supermarkets to small food and convenience stores — even coffee shops and home improvement centers. To date the core system has been installed in over 6,500 stores with a total terminal population (early 1997) of over 50,000.

The architecture has allowed a wide range of POS terminal types to be accommodated, from early proprietary systems with no open interfaces to the today's modern high performance Pentium systems with touch screens. The system architecture, and therefore **ISS*45***, is continually evolving with new features being incorporated as the retail environment changes and as retailers move toward their visions of the future. The 4GL tool sets described later in this paper make this evolution much easier than earlier systems—where the only option may be to write new software code.

The architecture is designed for this kind of flexibility, but the design also incorporates maximum resilience: no single point of failure can cause loss of any of the four primary system functions — POS, Controller Service, Applications, Communications — and certainly not a general store-down situation. Of course there are scenarios in which an individual terminal, board, cable, or PC may fail, but such problems in no way affect the remainder of the system. This constant operation is critical, but transaction data is equally important: central point data is always available, up-to-the-instant from at least two fully independent points.

Other designers have claimed that performance trade-offs are necessary when designing a system for maximum resilience. No such compromises have been made with **ISS*45:***

- Response performance at POS is about 100-150ms from scan to print. This is fast enough that the printer has often finished the receipt line before the clerk has set down the item being scanned.

- Throughput at each lane is over 3 ½ items per second, per terminal — and that's in a store of over 200 terminals.
- Throughput at each lane is over 3 ½ items per second, per terminal.
- The **ISS*45*** Controller Service Software has been stress tested at a rate of 350,000 items per hour. Even at this rate the consolidated transaction history updates in the back office are never more than 1-2 seconds behind the actual activity at the POS terminals.

OVERVIEW

While parts of the core architecture have driven "proprietary" terminals in the past, the **ISS*45*** system is based upon standard industry platforms, end-to-end.

The POS terminal operates under the MS-DOS system with a wide range of peripherals and interfaces for the connection of printers, scanners, readers, etc. With such a growing plethora of retail devices the software must be easily changed for new devices to make use of their facilities. **ISS*45*** has device-dependent drivers for each type of peripheral: this isolates the application software from the technical differences of each individual device. This strategy also provides a consistent interface to the application, relieving it of the problems of retries, data conversions, validation, etc.

The 4GL tool set allows the application to be tailored for the "look and feel" retailers require without forcing them to adopt any particular convention. This is achieved with a tool known as **ISS*45 Presentation Manager***—IPM. While IPM provides the user interface, the core application software, which can be further extended and customized with "C" language programming, provides all the facilities needed by today's sophisticated retailers.

No POS terminal in a modern store operates in isolation, so inter-operation with other computers is necessary, either locally or at a central site. Again a standard set of facilities has been provided to make this interworking consistent to the **ISS*45*** application: the services layer. For each type of service this layer knows where to obtain the information required.

For example, an electronic payments authorization service will connect to a task on another system that might have a WAN gateway. Therefore there are always two parts to the services layer—the service requester and the service provider. In some cases the service provider is resident on a PC workstation and in others it may be on a POS terminal. The services layer also provides the resilience features that are so important: in this example, if the main service provider was not able to respond, the service request is automatically routed to an alternate.

Transaction data is such a critical commodity that it is always held in at least three places, these being the POS-terminal where the sales transactions are generated, plus the Main and Alternate (Main) Servers. These two main server systems—together known as the **ISS*45 Controller Service Software*** (CSS)—can co-reside on POS terminals or operate as PC workstations. They ensure that data is synchronized in all elements of the **ISS*45*** system, including both physical server locations and each POS terminal. They also provide an update mechanism for remote databases or mainframes. In addition, the POS terminals typically hold a week of transaction data as a further back up.

Unlike standard "fail and back up" redundancy schemes, **ISS*45*** employs the concept of "Balanced Servers". That is, both Controller Services are always operating and sharing the workload for maximum efficiency. If there is a hardware failure on a device supporting one of the Controller Services, there is no takeover process required — since the other server is already running, it simply takes on the full load.

File updates are also the responsibility of the CSS and these will ensure correct synchronization of files, particularly the PLU file. If one server is not available then the other will be available to update the files resident on the PoS. Updates can be received from a remote mainframe or host, from another store system or updated locally.

TRANSACTION DATA HANDLING

Each POS terminal has its own disk drive. With the technology trending toward ever increasing sizes at lower and lower costs, 520MB and larger disks are now seen in terminals. This gives increased flexibility to the operation of the terminals since they are able to hold large amounts of transaction data locally.

Each POS terminal maintains its own transaction log to which each item is written as a 64-byte record. Each record is uniquely identified by transaction number, type code and date/time. As new features are added, new record types can be created by the application.

All activity at the POS is logged, even down to the level of printer jams or changes in the audit roll following a paper out. This information can be used to generate non time-critical alerts either locally or to central systems such as Netview or SMS-based systems.

Overall sales transactions are identified by header and trailer records. These exist to help the retailer perform sophisticated modeling from the data such as "basket analysis" to identify the type of customer, their spending, items purchased together, etc.

As each record is written to the local spool file then an element of the services layer will send this record to the nominated Controller Service — either MFS-1 or MFS-2 — in the CSS. This occurs during the idle loop of the POS application; the RSM (Retail System Manager) function will send up to 1K of data (16 records) each instance through the loop.

When this record is received at the nominated controller service it will be assigned a unique (for that server) sequence number, and be written in the **ISS*45*** transaction database.

The server that received the record will then negotiate with the other physical server and request to transmit the record. The other will check for proper sequencing: it may go back and request earlier records too in case it notices that they are missing. Each server in the CSS maintains a table of sequence numbers for ***both*** services. This way, each server is fully aware of the status of its companion server.

Each record is therefore uniquely identified by the POS transaction number, the POS terminal number, the sequence number and the server at which the sequence number was assigned. In addition, each **ISS*45*** transaction database is uniquely tagged with a day version number. This ensures that there is no duplication of transaction data.

In the event that a server requests a record which the POS does not have available then the POS will flag this back and the server will then synchronize to the earliest available record.

At the end of day the POS will increment its transaction day version number and the original file will be backed up. A utility is available to recover this data in the event of a disaster situation.

On-Line Transaction Processing (OLTP) systems need to be able to handle extreme volumes of individual lookups and transactions. **ISS*45***'s OLTP system provides a special utility known as "Quick-DEX" to process the transaction files. Q-DEX gives better performance on file I/O than standard ISAM file systems which would not be able to cope with, for example, the 350,000 items per hour at which the system has been benchmarked.

The Suspend/Recall operation works from the transaction file held on the Main Servers. Therefore a recall operation is possible at any POS terminal. That is, if the PLU is for any reason absent from the POS terminal, it will automatically check to the master PLU file on the Controller Service before issuing a "not-on-file".

PLU OPERATION

The PLU is also part of the services layer and has a copy of the PLU file normally held on each terminal.

The PLU file uses the Q-DEX file system to give improved PLU performance. This pre-loads the index information into memory so that the hashing algorithms can access the PLU data record in milliseconds. The PLU data record at the POS is 64 bytes and is a subset of the record held on the servers.

Updates to the PLU file may either be created locally or received from another system. When the data is received it is immediately updated in the database on that Main Server and then marked in the activity maintenance file. There are two cyclical files, one on each server, and named differently on each.

When a record has been written to the file, it is transmitted via a background process to the other server. This will execute the record—for example a PLU price change—on its own database.

Changes for files that are also resident on the POS terminals are logged into another maintenance log which is periodically read by all POS terminals. The POS then updates its local files. A PLU can be modified and added to a terminal at any time by the transaction processing subsystem, including during transactions, via a trickle method. Note, however, that **ISS*45*** will not permit the use of an updated price in the middle of a transaction, ensuring that prices of an item are consistent throughout that transaction.

The POS is continually checking that it is using the correct version of the file and if not then will download the correct version, in its entirety. If the POS is using the correct version but the pointer to the maintenance log is not at the logical end of file it will process all pending entries from the server.

The file versions are updated at each end of day at which point all the pointers in the log files are reset.

OTHER FILES

The POS system requires more than just the PLU file and other files are distributed or held at both the main and alternate servers.

The synchronization mechanism is identical to that described for the PLU data but in addition it is possible to mark a file as being non-redundant. In this case changes logged to the database maintenance logs are not propagated across to the other server.

Certain files do not reside on the POS—for example, the cashier file. When the cashier logs on to the system, connection is made to one of the servers and the records pertaining to the cashier are sent to the affected POS terminal. At cashier log-off, the reverse process takes place.

To ensure that a cashier cannot log on simultaneously at two terminals, the RSM looks at the transaction records being sent from the POS. When it identifies a cashier log-on, it marks the cashier file. This occurs at both locations of the Controller Service..

SERVICES LAYER

The services layer, “RSM”, is key to the whole **ISS*45*** system. It is implemented using the standard TCP/IP sockets interface making it as portable as possible across a range of hardware platforms and operating environments.

The services layer uses the UDP datagram facility of TCP/IP and in addition has its own application level protocol to ensure message delivery. By using UDP rather than NetBIOS the load on routers and bridges is reduced. Also, the amount of network traffic is limited to just what is required by the IP stack—such as ARP—and the application.

Each service is configured to communicate with a service provider and, where appropriate, an alternative address for the service. Since each service addresses its provider by name which

RSM then maps to an IP address, it follows that these can be located anywhere. This makes the provision of such items as electronic payments system authorizations much easier and reduces the amount of duplication within any one store. In this example, while the primary electronic payments system authorization facility may be in one store, if the LAN in that store is bridged across a backbone network (as is becoming more and more common) then a "hot standby" system may be located elsewhere to provide a backup. Of course this can be applied to any of the services required by the POS system.

As new service facilities are required, each requires some additional software to present an interface specific to the types of requests and responses for that application. For example, an electronic payments system request and response is different from a Cashier monitoring transmission.

The application interfaces to RSM through a defined set of application program interfaces (APIs). The services layer extends beyond that of the communications architecture to include the peripherals and interfaces to DOS and an RPC mechanism which allows execution—for example, of a search task—remotely. All of these are provided to allow a common API to the application.

TOOLS

One of the principal tools is IPM, discussed earlier. This provides the 4GL techniques which allow the user / developer to configure how the system looks (on screens and reports) through a set of menus and forms. Each form or menu consists of the fields which perform different tasks to make up a complete POS feature or application.

Up to 10 levels of menus are possible with up to 20 selection entries per menu. The size and format of each menu is programmable as is its location on the screen. A screen-builder facility allows the specification of inputs and outputs to the operator without the need to compile code.

Links to user-written C ("CustomLink") code is possible through specific hooks on each field, just prior to the application processing (if any) and just after the application processing. In addition pre- and post-processing hooks exist for the form functions, similar to Visual Basic load functions.

Help text items may also be linked to each field to give full context sensitive help facilities.

A query facility, IQL, allows entry forms to be defined, for example to query the PLU file. Also, the designer may specify which of these should be displayed or scrolled through.

The menu system, screen builder and query facility are all based around a common data dictionary unique to the application.

A version control system is built into IPM to ensure that the correct version is being used and to control automated updates to the POS.

RESILIENCE AND RECOVERY

The recovery mechanisms are detailed below for each of the possible failure scenarios.

POS Failure: If a POS fails then the sale in progress is abandoned. When the POS returns it will check the version number of the file it holds against the version number of the files on the primary server. If they are different then the file is retrieved from the server. Once all files have been synchronized then the POS is available for use again. (Note that the TeamPoS 5000 terminal has an internal battery backup. Power failures, for example, of up to 10 minutes are tolerated without this failure mechanism being triggered. When power is restored, the clerk can continue the transaction without loss of process in the order as if no power failure had occurred.)

LAN Failure: If the LAN fails, such that no device can talk to any other, then many services are lost—EPS (if connected via the LAN), cashier central logon / off, alerts. All other POS activity may continue. An offline cashier mode comes into operation to allow the continuance of the POS activity.

When the LAN is restored the POS will send the complete detailed transaction log data to the nominated server which will perform the synchronization mechanism discussed earlier.

The POS will also check the file version control numbers for changes and download them as required.

Server Failure: In the event that this unit fails then any services unique to the device are lost. All terminals which previously sent transaction data to this particular server will now, after unsuccessfully attempting a send, work with the other server instead.

When the server returns then it will synchronize the transaction log by means of the application protocol handshake detailed earlier.

Any changes to the database of redundant files will be automatically synchronized by means of the background process, sending records from the maintenance log of the other server.

Note that while the failed server is waiting to be returned to service, another device in the store can be newly designated as a server. The system is therefore returned to a fully redundant server system status even before the repair is completed.

Other Main Server failure: The failure modes and recovery mechanism are identical on both servers.

WINDOWS AND ISS45

ISS*45* application currently runs under DOS, and a full 32-bit Window NT application, **ISS*45 V8*** will shortly be released. Early "drafts" of this product have been shown at the 1996 and 1997 FMI MarketTechnics exhibitions. The Windows NT operating system will run the Controller Services, using the Microsoft SQL Server (Version 6.5 or the upcoming Version 7) database with full ODBC. All hands-on and system management will be performed from the NT servers, while the terminals may remain configured with DOS. This solution has been elegantly engineered to provide the NT services without the requirement for the expensive NT Advanced Server software — and all the hardware you need to operate it successfully — only the low cost NT Workstation is required under SQL Server Version 7. To further reduce costs, replication of the NT office can be provided either by a second NT PC or via a DOS workstation depending upon configuration choices.

A separate release, also in current development, will provide the option to run terminals with either NT Workstation or Win95 / Win97. Users will be able to independently select a DOS or Windows front end with either the DOS or WIN NT back office. Note that both these applications will be operating using full Windows capabilities; these are ***not*** the same DOS application merely running in a Windows DOS "box". The system will also be modified to make it compatible with DDE to allow inter-operation with other software; a customer ordering system, for example. The system completes its migration to full object oriented code, (in which it is already internally designed), and compiled to give full 32 bit application and OLE2 compliance. This version will make most use of the features available under Windows 95 and Windows 97 and will also run as a native Windows NT 32 bit application.

All this development closely mirrors the current architecture of **ISS*45*** but gives greater flexibility.

To Your Success,

Tony

Tony van Seventer
Director: Supermarket Systems