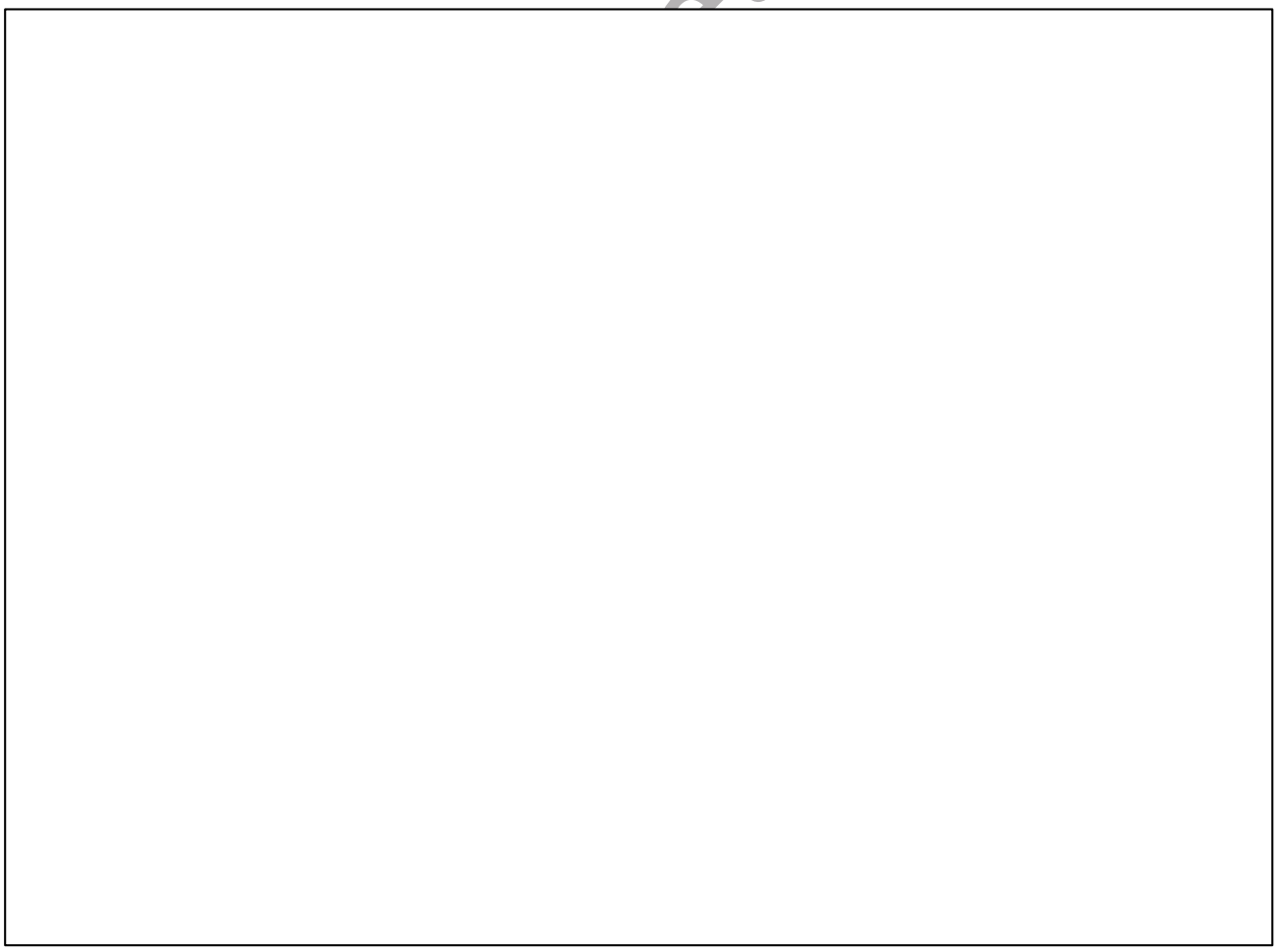
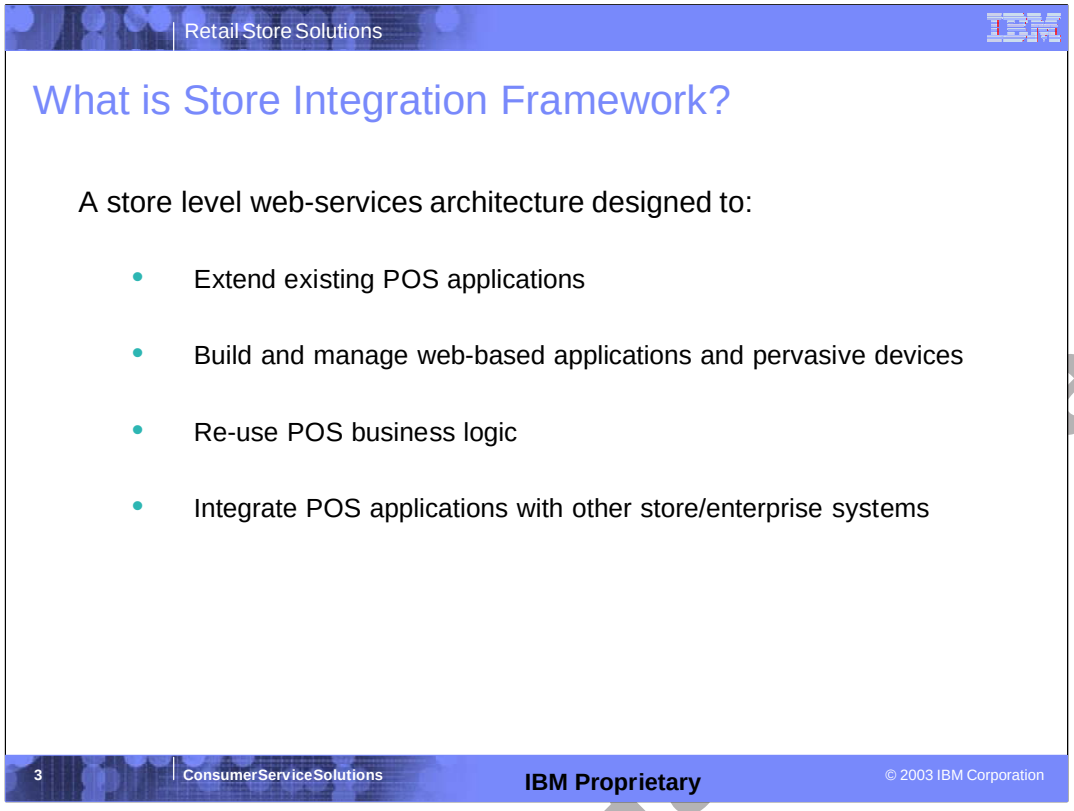
Discussion agenda

- Store Integration Framework - Overview
- Data Integration Facility - Overview
- Scenarios
- IBM's Enterprise Solution



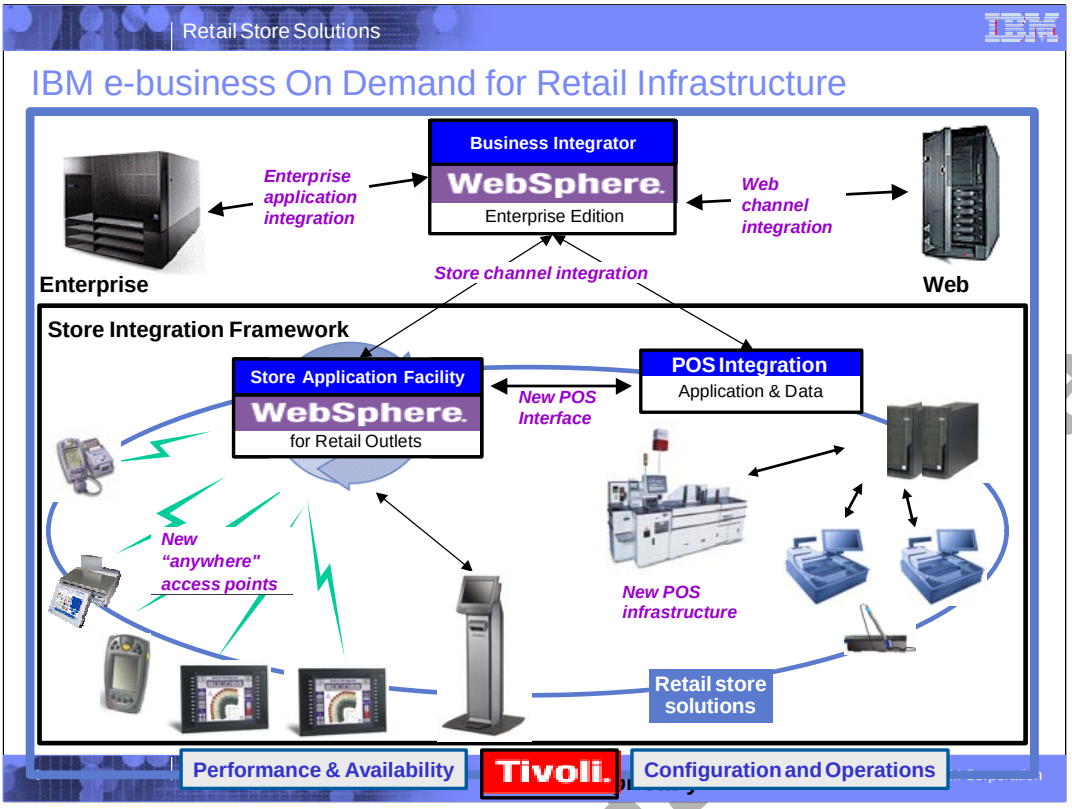


The Store Integration Framework is designed on the J2EE programming model. The SIF provides an open API that allows data and functionality that has been traditionally locked in the POS terminal sales application to be leveraged across the enterprise. This API also provides a standard interface to extend the POS application functionality using the java programming standard. Both the extension and use of the POS business logic is implemented using java programs or as web services. Use of the Java programming environment allows the solution to be platform independent.

Possible applications of the SIF:

- Through the use of a virtual terminal, a browser-based application can use the POS application business logic for pricing and promotion.

- Store transaction data can be shared immediately across the enterprise, allowing the consolidation of multiple sales channels, including the internet.

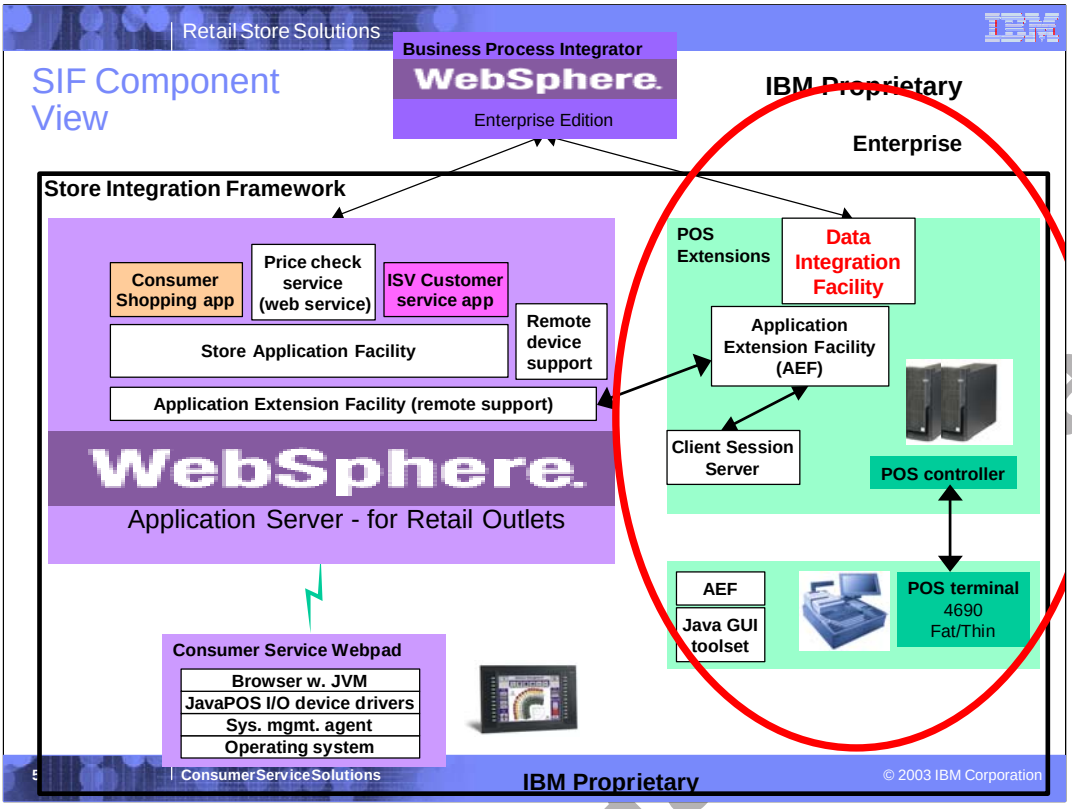


This is an overview of the new On-demand for Retail Architecture, which includes the Store Integration Framework inside the store. **In slide show mode, the new features of this architecture are displayed in 4 clicks as follows:**

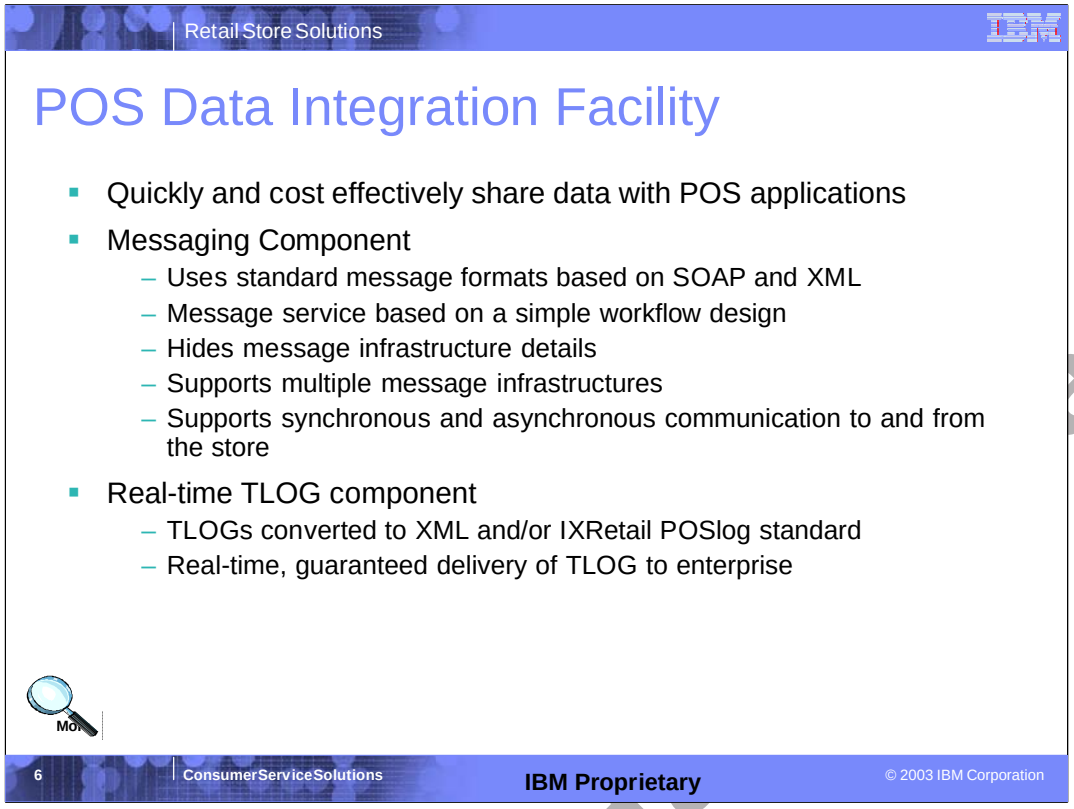
Click 1: Store Integration Framework provides new POS Integration capabilities, allowing for extensions to existing POS applications and data. Four key POS enhancements include: 1) Data Integration for easier access to critical POS data such as transaction logs; 2) Application Extensions for adding function to POS and extending function to the Web; 3) Client Session Server for running POS applications on the POS controller on behalf of remote devices attached to the Store Application Facility; 4) Enhanced POS terminal support including enhanced GUI tools. Also, we see new devices such as self-checkout and Consumer Transaction Terminals integrated to the POS.

Click 2: Store Integration Framework adds a new platform called the Store Application Facility in the store for web-based applications. This platform is based on a new WebSphere Application Server for Retail Outlets offering, providing a scaled down WAS for the store. This platform provides a standard "container" for web-based consumer service applications. This platform supports integration of the web apps to the POS, to the enterprise, and to each other.

Click 3: Now we see various new wireless devices, such as wireless handhelds, tablets, kiosks, and even a produce scale driven by the Store Application Facility. A key feature of the Store Integration Framework is supporting the same application delivered to different types of devices. (Note: Produce scale picture courtesy of Mettler-Toledo).



The Data Integration Facility component of the Store Integration Framework is installed on the POS controller. The DIF includes extensions for WebSphere MQ/MQe messaging for communication with the store, but is designed to be extended to support additional messaging systems. The DIF supports communication with the POS terminal sales applications using pipes and TCP/IP. Minor extensions to the POS terminal sales applications are required to communicate with the DIF. These extensions are supplied with IBM 4690 applications SA, GSA, and ACE.



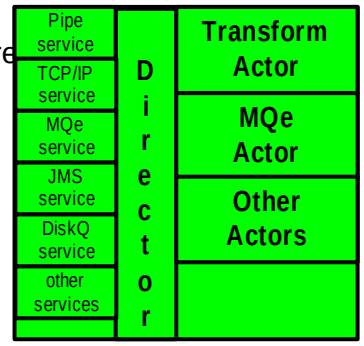
SOAP – Simple Object Access Protocol

The messaging component of the DIF simplifies the details of transmitting messages between the POS controller and other servers across the enterprise. Included messaging components support WebSphere MQ/MQe, and HTTP transport protocols. The API is exposed to extend the framework to support additional transport protocols. WebSphere MQ/MQe is preferred due to it's support for guaranteed messaging.

The real-time TLOG component is a fully designed solution that utilizes the DIF. As TLOG data is generated by the POS application, it is converted to the industry standard XML/IXRetail POSLog format and transmitted to a server using WebSphere MQe. There are multiple customizations for data filtering, and controlling bandwidth.

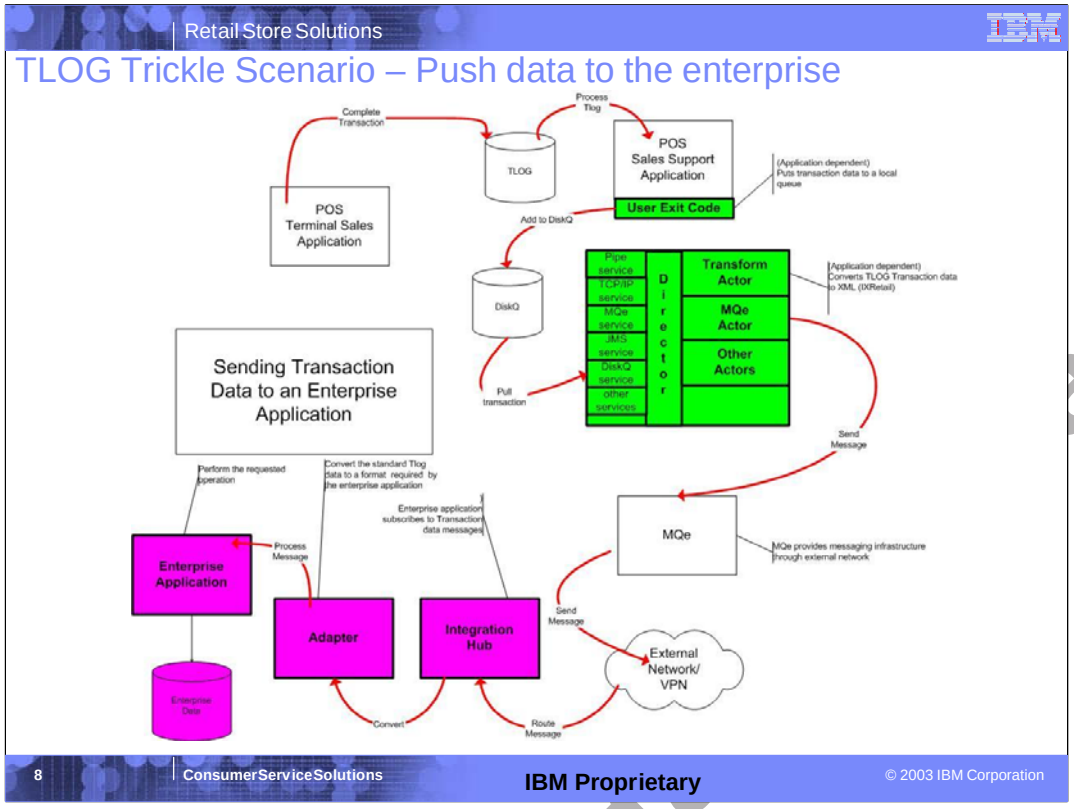
Message Processor

- Provides a flexible interface to the messaging middleware
- Based on a simple workflow architecture
- Configurable
- Extensible



The framework provides connections and messaging components to interface to both the POS application and enterprise messaging systems. The DIF connects to the POS application through a Pipe or TCP/IP service and listens for TLOG data to be generated by the application. The data is then passed to an Actor, which is simply a java component that implements the Actor interface defined in the framework. Several generic actors are provided as part of the framework, and are easily extended. The Director is a special actor, defined in the framework, that coordinates several actors to be used in sequence to perform a function. The Director passes the results of an actor as the input to the next actor. Actors are designed to transmit information using the SOAP message structure, but may also support binary stream data.

Example: A transaction is transmitted over a pipe to the DIF director. The director passes it to an actor to filter the transaction so only totals information remains, and returns the result. This result is passed to the next actor that translates the native POS application data into IXRetail POSLog format, and returns the result. This result is passed to a the next actor that posts the data on an MQe queue to be consumed by other enterprise applications.



This scenario is required for the Returns Management and several other use cases. Part of the Returns Management use case involves saving sales transaction data at a central site. In this use case transactions are sent to a central site as they complete and the POS controller uses a “send and forget” type of message.

The green in the diagram identifies the components that are included in DIF. The purple identifies the components that could be implemented using products from Software Group such as WebSphere MQ Integration Broker.

Complete Transaction
At the ***Terminal Sales Application*** once sufficient payment is processed to satisfy the balance due, the sales transaction is completed and the resulting transaction is written to the transaction log (***TLOG***). Prior to this point, the transaction is pending and could be changed or canceled. The ***TLOG*** contains the details of completed transactions for a given accounting period.

Process TLOG Entry
The ***POS Sales Support Application*** is the process responsible for processing completed transactions from the TLOG. For store purposes it keeps track of the money and sales, and provides the store manager the information to manage the store. It knows which TLOG to process and it keeps meticulous checkpoints to insure that a transaction is processed only once. This TLOG processing infrastructure is leveraged by using an ***Exit*** which gets control when the base processing is completed but before transaction processing is marked “completed”, to add the transaction to the DiskQ. The solution should have minimal performance impact on the existing sales support application.

Add to DiskQ
This queue isolates the sales support processing from the messaging processing while maintaining the reliability characteristics of existing ***POS Sales Support Application.*** By using a highly optimized local queuing mechanism, minimal impact on the ***POS Sales Support Processing*** is achieved.

Pull Transaction
For 4690 applications the transaction pulled by the DiskQ service is a string in “carriage control, line feed delimited”, ASCII, packed decimal format. The DIF RUNTIME is responsible for sending the data to the enterprise and optionally parsing and transforming the POS proprietary format into the IXRetail POSlog format.

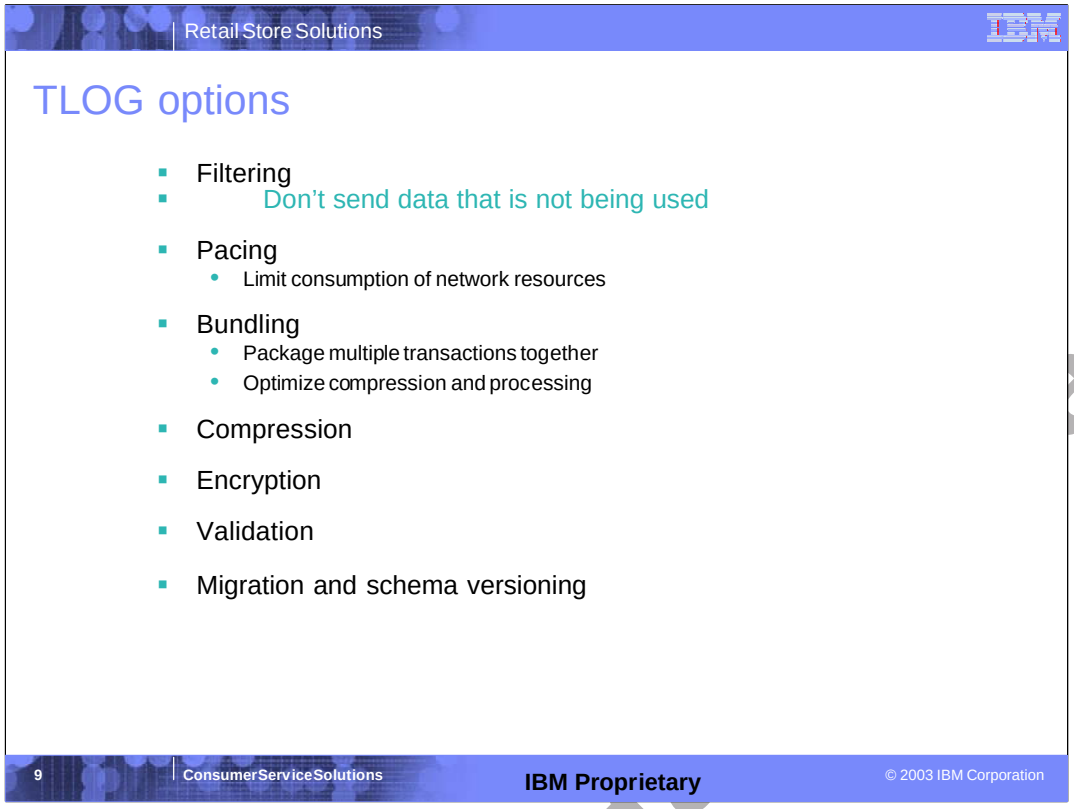
Once the transaction data is converted to the IXRetail POSLog format, it needs to be sent to the enterprise. This is accomplished through the ***MQe Messaging Actor.*** The MQe Messaging Actor provides access to a messaging infrastructure. The workflow design allows multiple communication infrastructures to be used, however the initial plan is to use MQSeries Everyplace or a JMS Actor. If a customer’s messaging infrastructure is different, the messaging actor is designed to be replaceable via services

Send Message (MQE) and Send Message (MQ internals)
This component represents the “out of the box” messaging infrastructure provided with the framework. Since the DiskQ function provides the queuing function, either synchronous or asynchronous messaging can be used. The default is to use synchronous.

Route Message
The integration hub is responsible for routing the message to the appropriate applications. It isolates the store solutions from the details of the enterprise infrastructure and simplifies the network infrastructure. For this scenario, it needs to be able to route the information or parts of the information to multiple applications.

Adapter
An adapter is an optional component that is required when the input and output data formats are not the same.

Enterprise Application
The enterprise application processes the message. For this scenario, it is a consumer of data. The transaction data from the stores includes data for inventory management, sales and demand analysis, promotion analysis, finance, credit cards usage, etc.



Filtering

Via configuration, this allows retailers to exclude selected optional data. For example if the retailer is not using the "Loan" and "Pickup" data, this information can be excluded in the IBM POS specific IXRetail POSLog. The granularity of the filter function is limited to groups of data (e.g., Data Maintenance string, Close String, GSA extra data).

Pacing

The pacing function allows the retailer to do the following:

1. Limit the amount of network resources that are consumed, including when the network becomes available after a period of being disconnected. This allows the retailer to allocate network resources to higher priority tasks such as supporting sales transactions and high priority messages related to sales transactions. Through configuration it is possible to define this threshold for multiple time periods throughout the day. With this method, it is possible to defer all transaction log (TLOG) processing until the store is closed. Once the defined threshold is exceeded, message processing will be delayed until the defined threshold is reestablished.
2. Bundle together transactions before they are processed, to improve the efficiency of the transaction log (TLOG) processing. Based on experiments, we know that the effectiveness of the compression techniques improves dramatically as the size of the XML document grows. Through configuration it is possible to define the number of transactions to accumulate before processing and includes a wait time limit. The DiskQ function will receive messages from sales support and accumulate them until the defined threshold has been met. These messages are then bundled into one message with multiple attachments and sent via an MQe queue. This allows the retailer to specify the following: "Process the transaction data after 10 transactions have accumulated or process the transaction data after waiting 10 minutes from the time the first transaction is received".

Compression

The ability to optionally compress the converted transaction log (TLOG) or other message. For the first release, the WebSphere MQe compression function is used. MQe supports standard compression algorithms such as ZIP, GZip, LZW. Please consult WebSphere MQe documentation for more information.

Encryption

The ability to optionally encrypt the converted transaction log (TLOG) or other messages. For the first release, the WebSphere MQe encryption function is used. MQe supports standard encryption algorithms such as DES, MARS, RIe. Please consult WebSphere MQe documentation for more information.

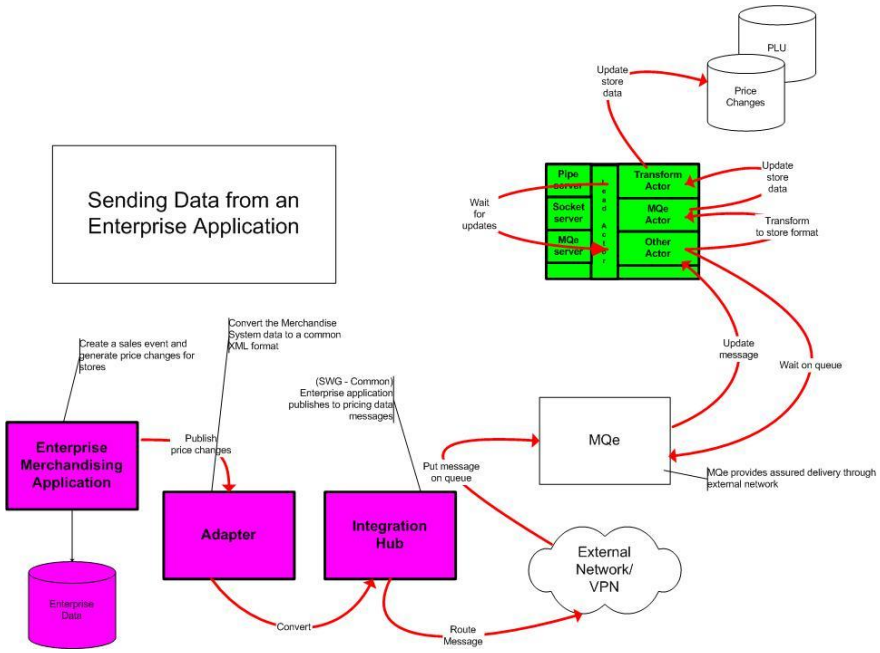
Validation

For retailers who extend the transaction log (TLOG) transformation process or change the configuration using the filtering function, XML validation must be used throughout the test process to insure the changes produce valid XML. This is implemented as an actor which can be configured and run on any Java platform.

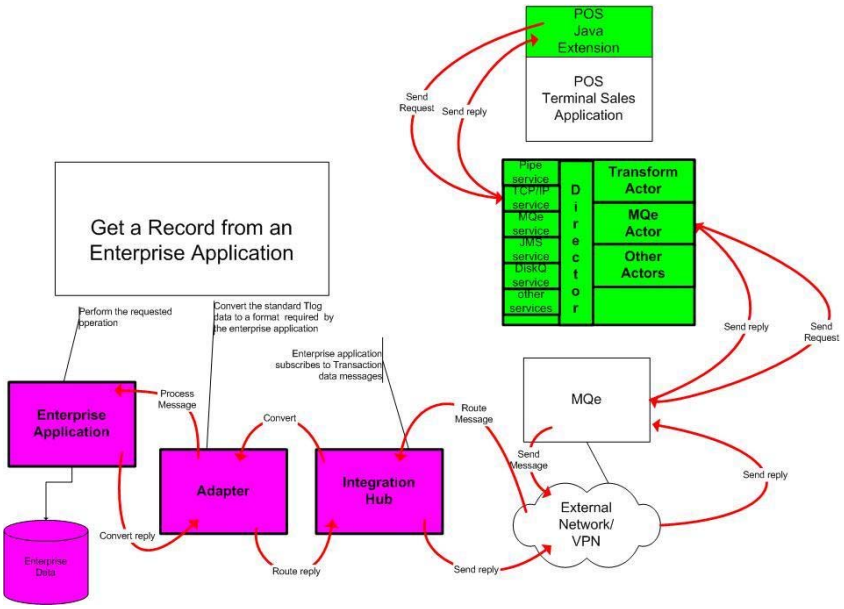
Migration/Schema Versioning

The IXRetail POSLog standard is subject to change. It is not complete and is likely to grow. The TLOGs of existing POS applications will change, especially the more recent applications like ACE, resulting in changes to the IBM POS specific IXRetail POSLog. And last but not least, the customers are likely to add extensions to the existing POS TLOGs as they add capabilities to their systems. Throughout this evolution of POSlog schemas, customers must be able to stay at a particular schema version until they are ready to move.

Send Data to the Store Scenario



Get Data from the Enterprise Scenario



This describes the returns management scenario, but applies to any scenario that requires data stored in a central database to be accessed by a POS terminal sales application.

Send Request
At the POS terminal sales application, a barcoded receipt is scanned. The number is a unique identifier for the transaction and is used as a parameter for the data request (for the returns management scenario, this would be a getTransaction *request).* Using TCP/IP the message is sent to the DIF RUNTIME.

Send Request (via MQe)
For the case where MQe is used for messaging, the DIF RUNTIME is configured to use the *MQe Messaging Actor* which sends the request as a blocking MQe message.

Send Message (MQE) and Send Message (MQ internals), Route Message, Adapter, Enterprise Application
These components perform the same role described in the previous diagrams, however for this scenario the conversational request/reply message model (i.e., the application waits for the response) is supported.

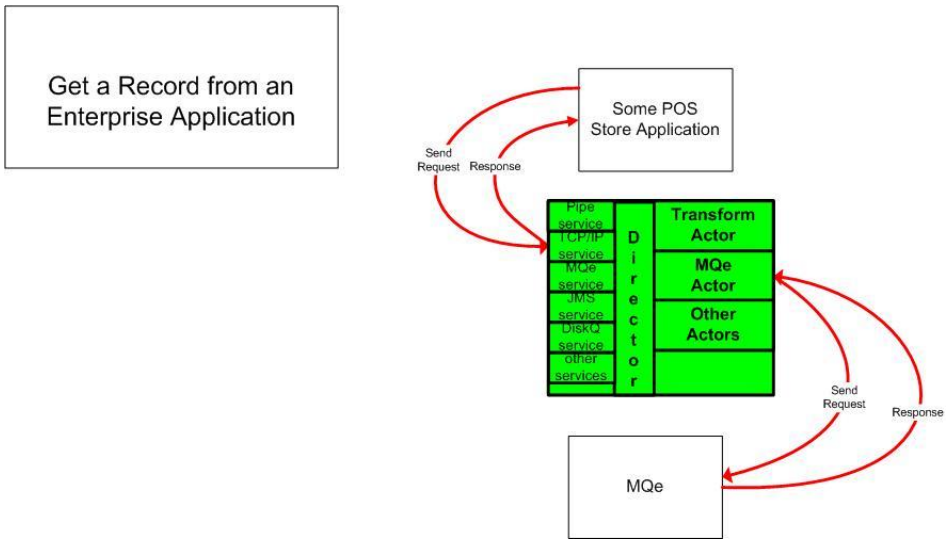
Enterprise Application
The enterprise application processes the request and sends a reply message to the originator

Adapter
The format of the data may need to be converted.

Route Message
For a reply message the routing information was provided with the request. No routing processing should be required for reply messages.

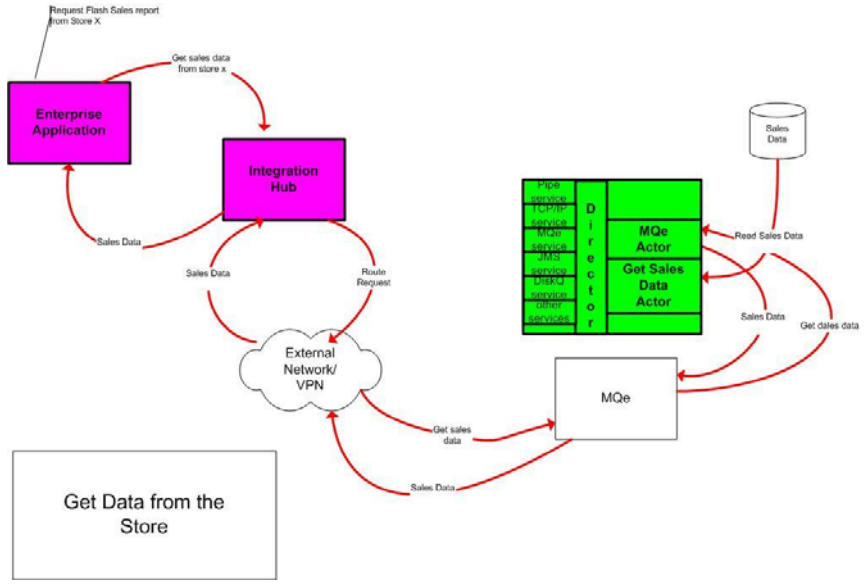
Send Reply...
The reply message is sent through the MQe Messaging Actor which is waiting (for a specified time) for the response with a specified message identifier. Once the message is received, it is forwarded to the originator via the DIF RUNTIME and Java Messaging API. If the wait time is exceeded, then it is considered an error that must be handled by the POS application. In the case where the reply message was just very delayed, the messaging system must be able to cleanup old messages by using a “message expiration” parameter.

Or …Get Data from the Enterprise (w/o Java at POS)



The following scenario is a variation of the prior scenario. Everything is the same except that the Pipe API over pipes is used to communicate with the DIF RUNTIME. This capability is required for customers who decided to evolve into the Store Integration Framework rather than an all or nothing approach. This capability would allow customers to gradually upgrade their systems to meet the hardware prerequisites of SIF while benefiting from the capabilities of messaging on all of their POS systems.

Get Data from the Store Scenario



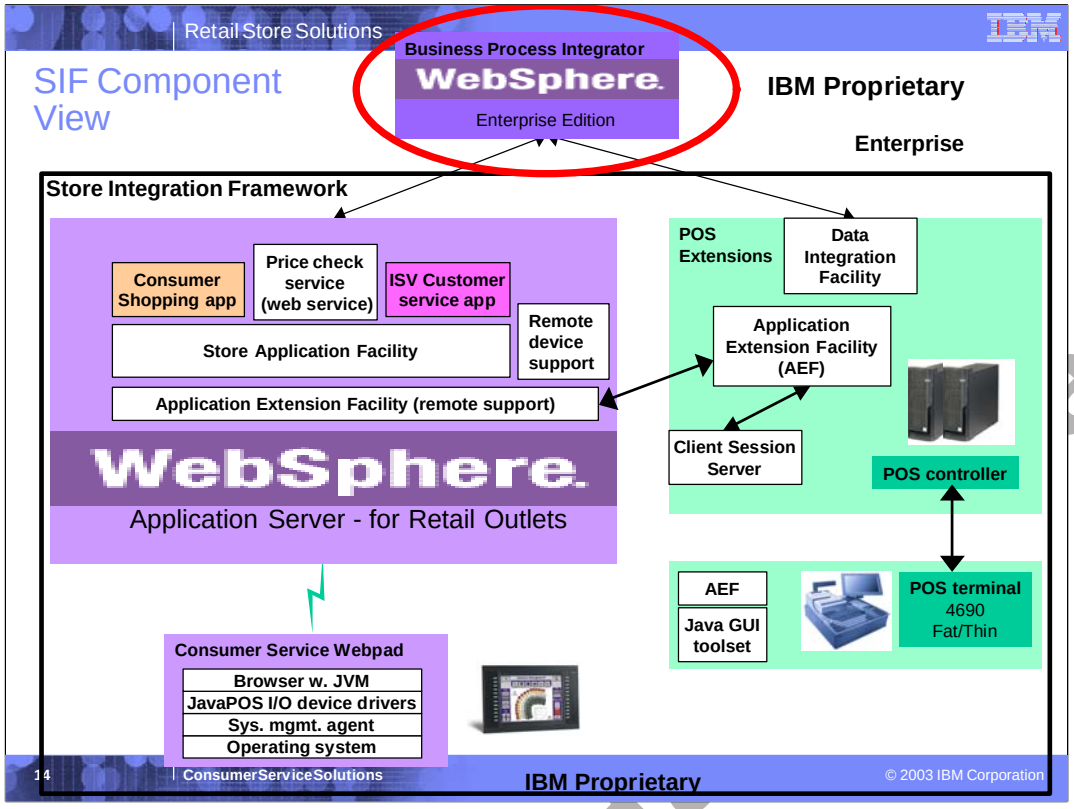
In this scenario an enterprise application needs data from a store or multiple stores and the stores act as a server of local store data. For example, a store contains the most current and accurate sales data during the day. During busy periods a district manager may want to view sales "flash" reporting for stores in his area, and may need access to this service from a remote location. Other examples may include cashier productivity reports or price information where store level competitive pricing is practiced. In each case, the store level POS controller contains valuable data that other applications may need to access. This data is often contained within specialized retail file structures (e.g., keyed files) and is formatted to satisfy POS terminal application requirements for speed and memory constraints (e.g., bit level flags, packed decimal data). In order to service requests for POS data, the Data Integration Facility Component will provide an infrastructure for getting data from the POS system in a standard format.

1) Get sales data from store x

From the enterprise a message is sent to each store requesting specific data. This is routed through the messaging infrastructure until it arrives at the MQe Actor in each store.

2) Get Sales Data Actor

Through configuration options, the "Get Sales Data" request is routed to the Get Sales Data Actor which processes the request and returns the result to the DIF RUNTIME where the results are forwarded to the requestor via the MQe Actor.



WebSphere Business Integrator (WBI) for Retail TLOG Processor

Retail Store Solutions

A standards based product for retail transaction log (TLOG) data brokering, transformation, and storage.

Complements the Data Integration Facility

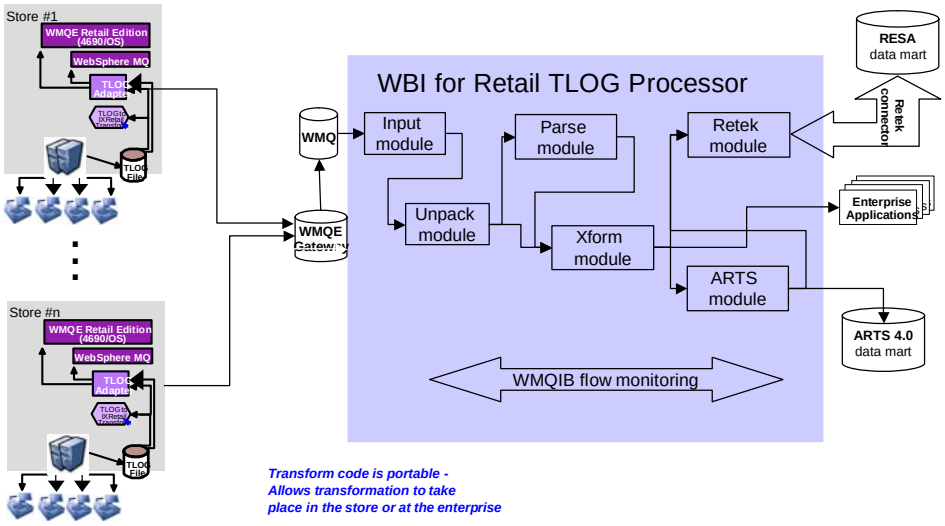
Not yet available as a product

July 10, 2003

WBI for Retail TLOG Processor

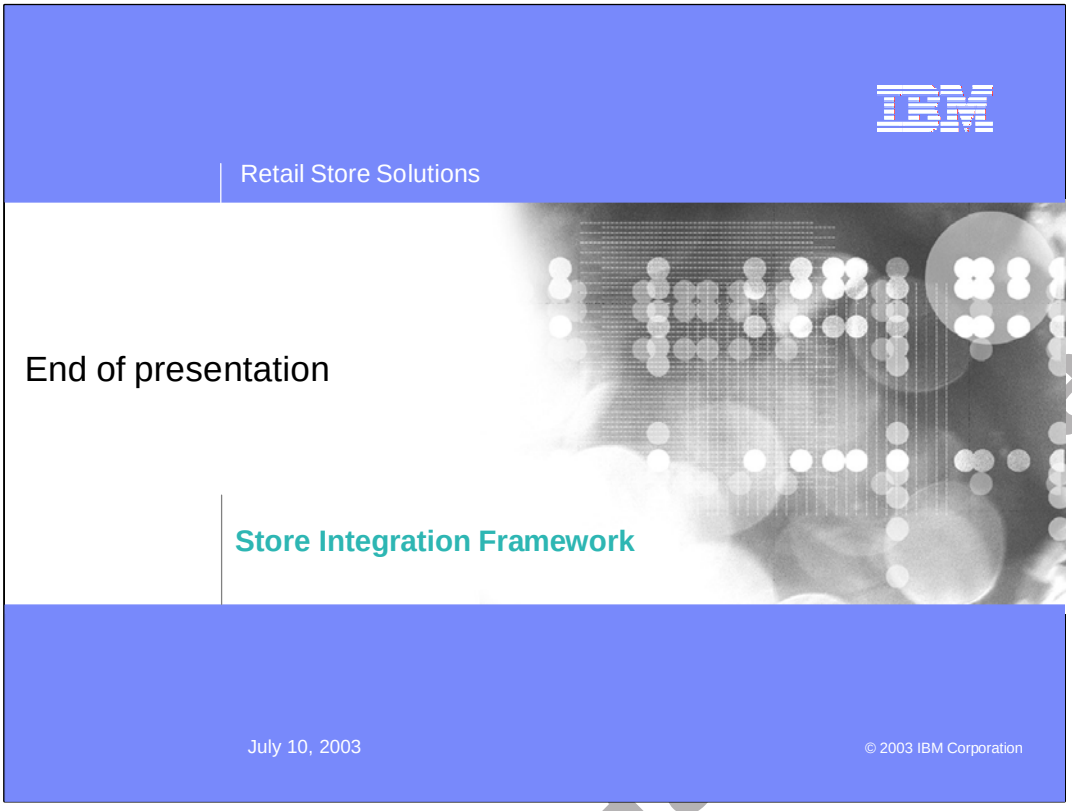
- **Pre –built WMQIB nodes, message sets, and flows:**
  - Optional data transform from native, POS specific, TLOG format to XML standard (IXRetail POSLog Ver. 1) format
    (can be done at the store or at the enterprise)
  - DB2 data mart in ARTS Ver. 4.0 format
  - IBM connector to Retek Sales Analysis (RESA) module
- **Initial support for 4690 POS formats: SA, ACE, GSA**
- **Compliments the RSS Data Integration Facility product in the stores**
  - But loosely coupled; either product can be sold alone
- **Easily configured and extended for additional endpoints**
- **Process flow monitoring and control**

WBI for Retail TLOG Processor completes the end-to-end solution



Transform code is portable - Allows transformation to take place in the store or at the enterprise

100s – 1000s of stores!



Optional pages are not linked to the main deck via navigation links.

Tailor the presentation by inserting these pages into the flow as required.