

::ISUG

techcast
series

Sybase Real Time Data Services

Naveen Puttagunta
Sr. Product Manager
Information Technology & Solutions Group
Sybase, Inc.

Why Real-Time Services

Sybase Real-Time Data Services Solution

Customer Use Case Scenario – Real Time Cash Flow

- Deployment Architecture

Customer Use Case Scenario – Credit Limit Risk Management

- Deployment Architecture

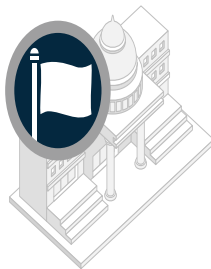
Demo

Companies seek to reduce any delays in moving information from the point of origination to the point of action.



To Improve Revenue

When a new cell-phone subscriber makes the first call, alert customer service to help to setup services like email, messaging, or text messaging



To Reduce Risk

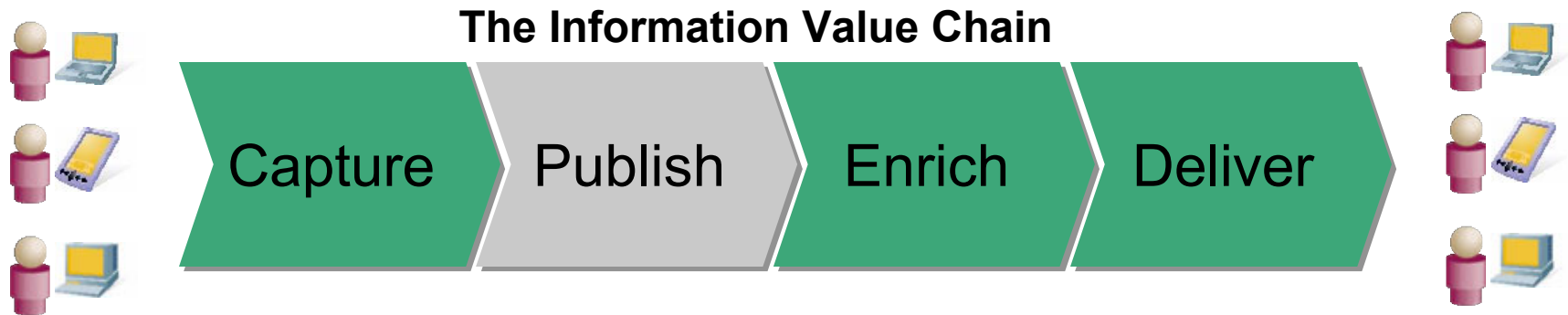
If a passport is stolen or lost, proactively alert border security, airport security, and police



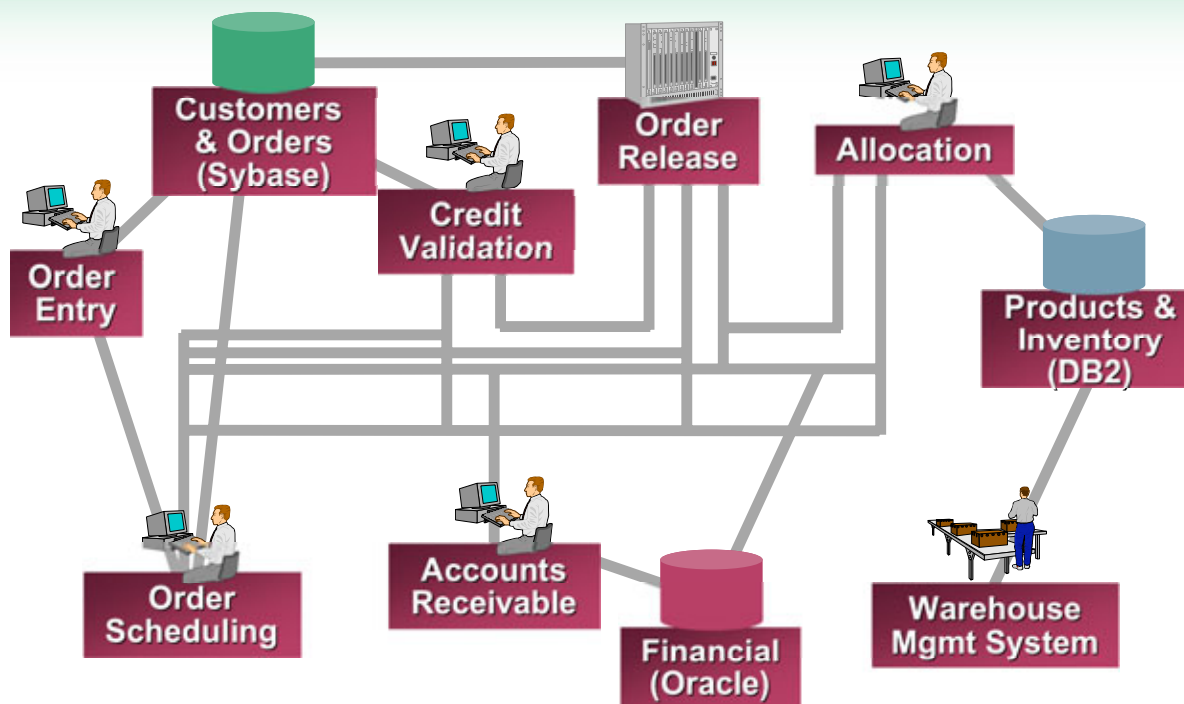
To Improve Customer Satisfaction

Update railway ticket availability as new tickets are issued enabling customers to purchase remote tickets

There are four fundamental building blocks of real-time information flow for the Unwired Enterprise.



Achieving agility requires eliminating the delays in publishing data from where it is captured to the systems and people that need to act upon it.

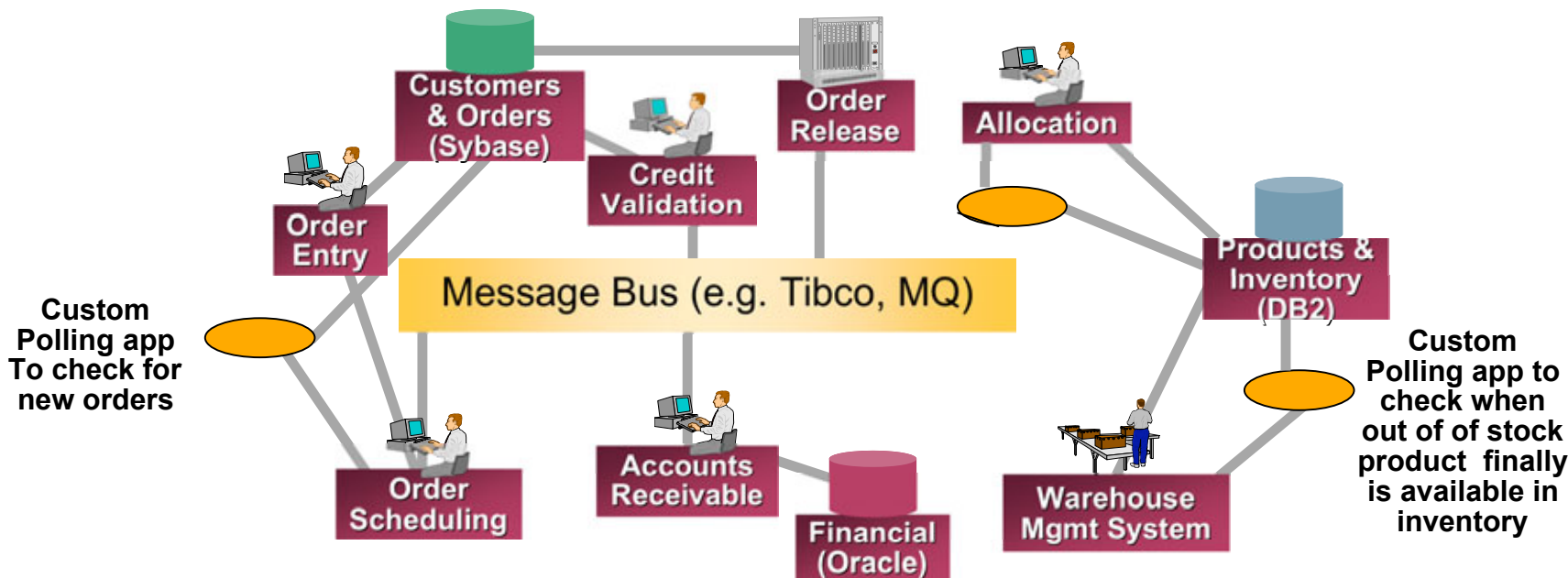


Point-point Applications ➔ *High Maintenance Costs*

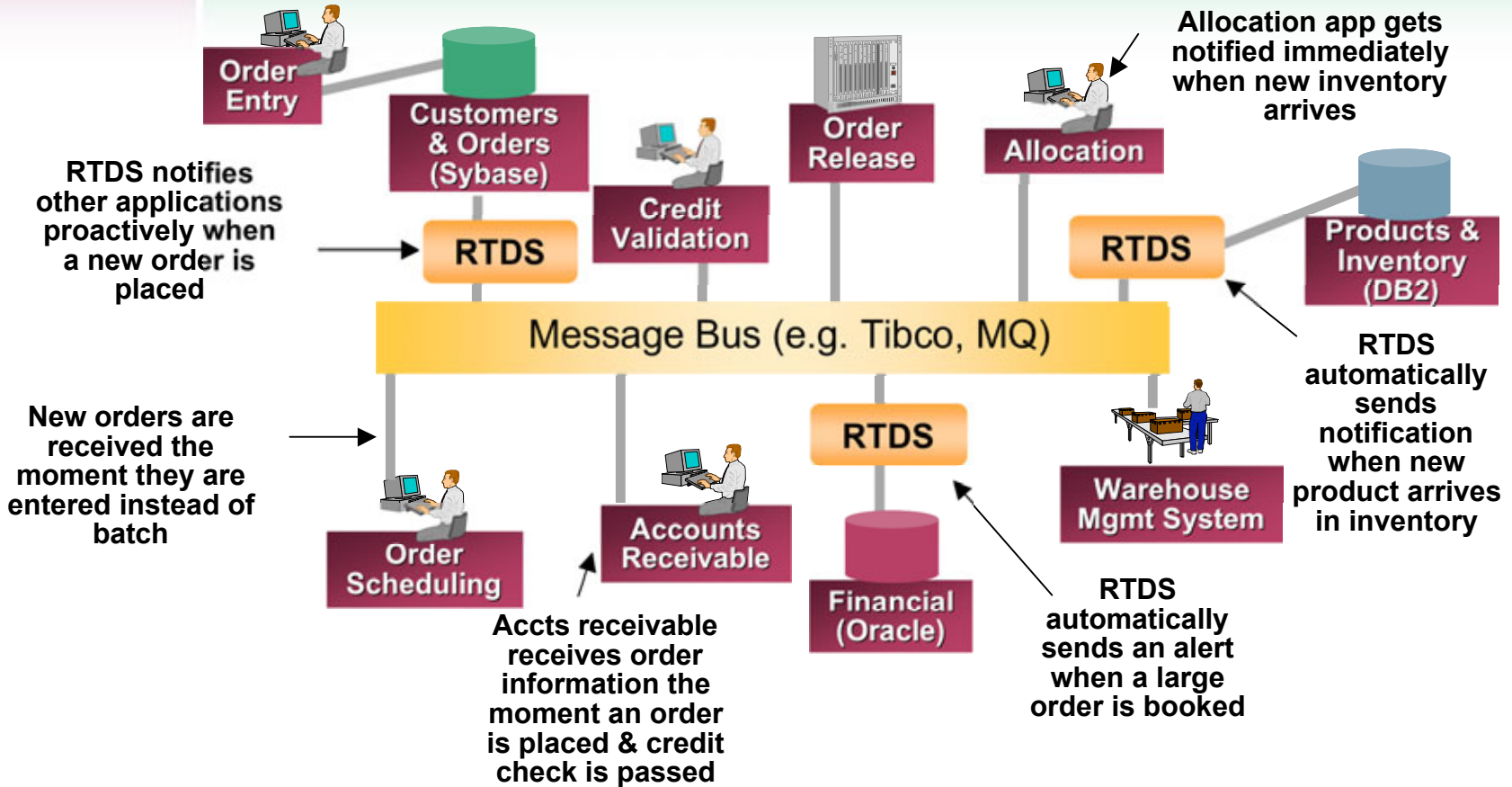
Change Capture Requires Polling/Batch process ➔ *Intrusive; Slows Performance*

Batch Updates for Intraday Changes ➔ *Delays Information*

Message Bus Simplifies Architecture However



- Polling applications still needed to capture changes
- Databases still cannot share information proactively
- Batch updates still being used for intra-day updates

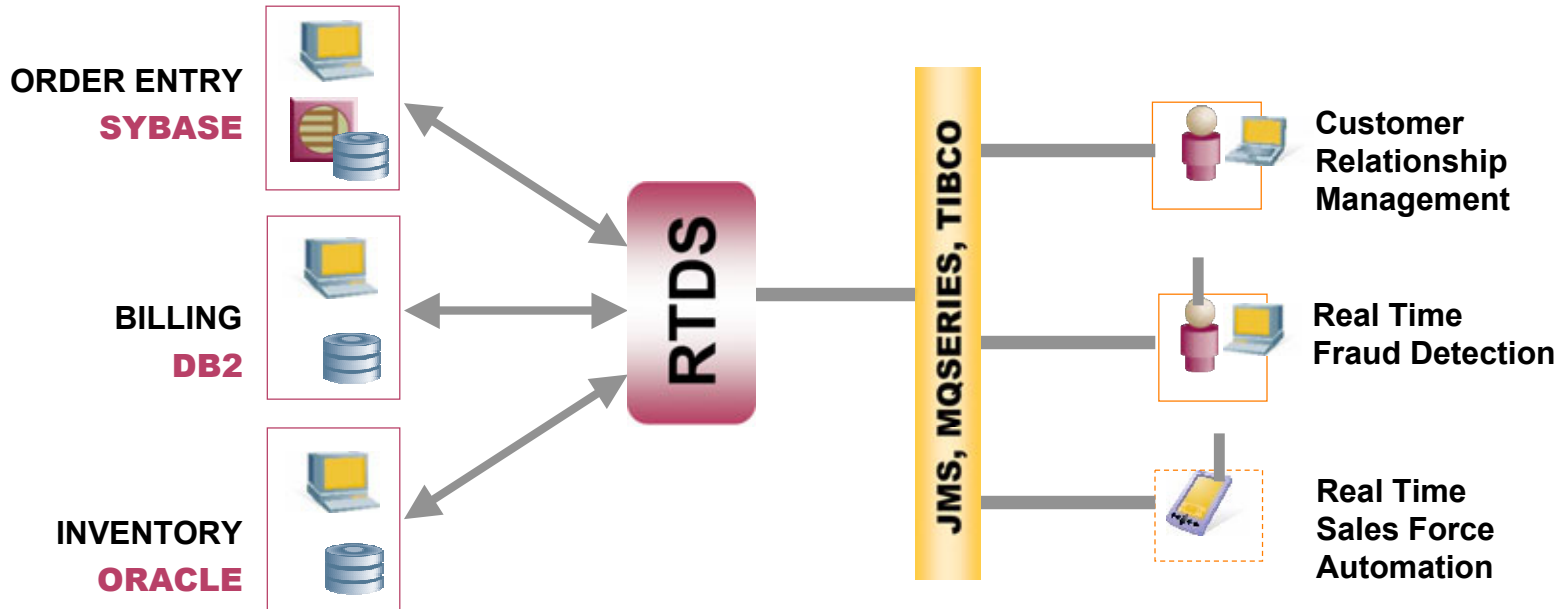


Eliminates need to write *polling applications* for capturing changes

Information is **shared in real-time** instead of batch

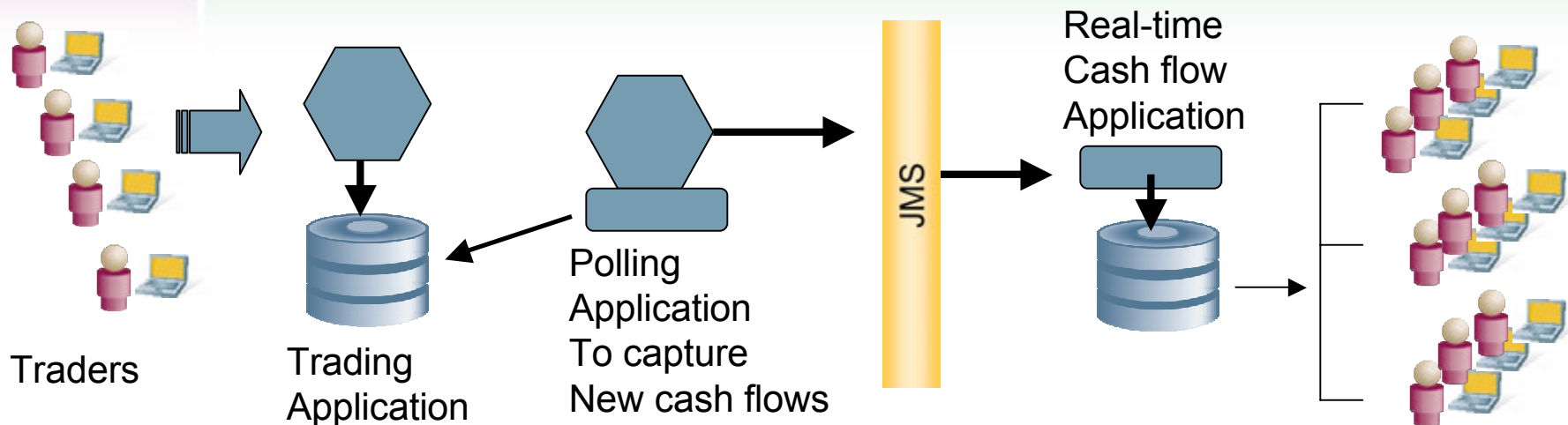
Databases **share event information proactively**, in a **standard format**

Sybase RTDS – Simple, Non-intrusive and Code-Free



Sybase RTDS *proactively* moves time-critical events from *heterogeneous data sources* to business applications, the moment a change occurs, via a messaging infrastructure.

- **Business**
 - Global Equities Group - Offers equity swap services to large hedge funds
 - Cash flow reconciliation is done at end of day
- **Challenge**
 - Customers and the Business wants a real-time view into cash flows as and when they are generated
- **Current Alternative**
 - Poll trading application periodically to capture new cash flows.
 - **Impacts performance of trading application significantly**
- **RTDS Solution**
 - Automatically publish cash-flows as and when they are generated
 - **Non-intrusive and Proactive**



Requirement:

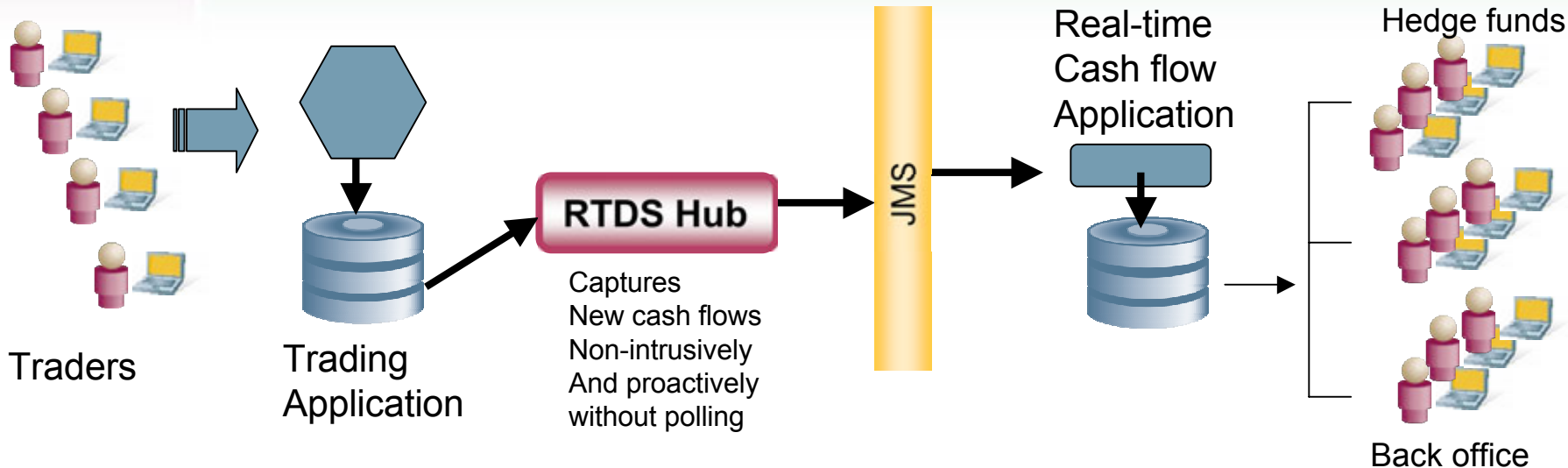
- Each equity swap generates a cash flow that is captured in the trading application
- These cash flows need to be consolidated in real-time to give customers a real-time view of their positions

Current Solution:

- Poll the trading application database periodically for new cash flows generated
- Once a new cash flow is detected, capture that cash flow information and send it to the real time cash flow consolidator application

Risks:

- Each trade involves a basket of securities and each swap has multiple line items of cash flow. Today they get to see only the consolidated cash flow for each trade
- The faster they poll the trading application database, the worse the impact on performance.

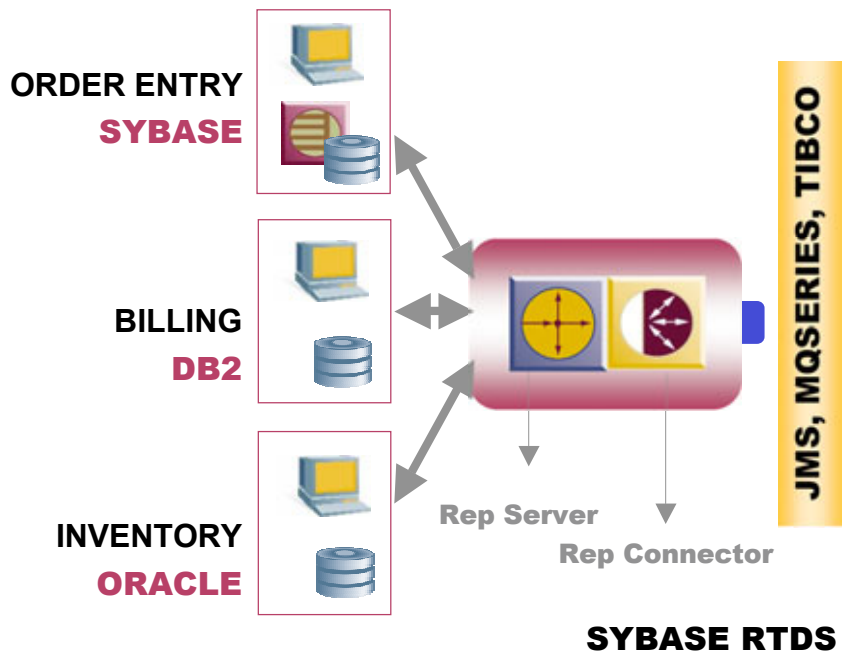


RTDS Solution:

- New cash flows are detected as and when they are created
- Every line item cash flow can be captured and forwarded to the real-time cash flow consolidator application.

Benefits:

- Cash flows are published proactively
- Zero impact on the trading systems, because RTDS eliminates polling
- Granular data on individual cash flows with a basket of securities.
- Simplifies architecture



Capabilities (RepServer → RepConnector)

- Non intrusive solution to propagate database events
- Replicates Full Transactions
- No Custom Applications or Custom Code
- Perfect fit for heterogeneous data sources
- Supports heterogeneous message transports
- DBA oriented.
- Does not lose a single transaction.

Benefits

- Eliminate 'polling' applications, reduce cost
- Be proactively notified when changes happen, without coding
- Zero impact on existing applications
- Enable multiple applications

Any Database to Any Message Bus

RepConnector 2.5

Packaging & Architecture

Bundled in RTDS

- MUST run within an Application Server
 - **SYBASE EAS 4.22 and EAS 5.0**
 - **BEA WebLogic 8.1**
 - **Implemented as a JCA (Java Connector Architecture) connector**
- Transforms DB transactions to XML and sends to Message Bus by default
 - **Messages follow dbeventstream.xsd**

Customization

- **Message Format Customization:** Ability to transform or format the message (*RepTransactionFormatter* interface)
- **Transport Customization:** Ability to send the message anywhere (file, disk, email ...) (*RepraClient* interface)
- **Custom Message Generator for Tibco Active Enterprise Wired Message Format**
- **Allows filtering of events, using JAVA API primitives.**

•DB → Message Bus

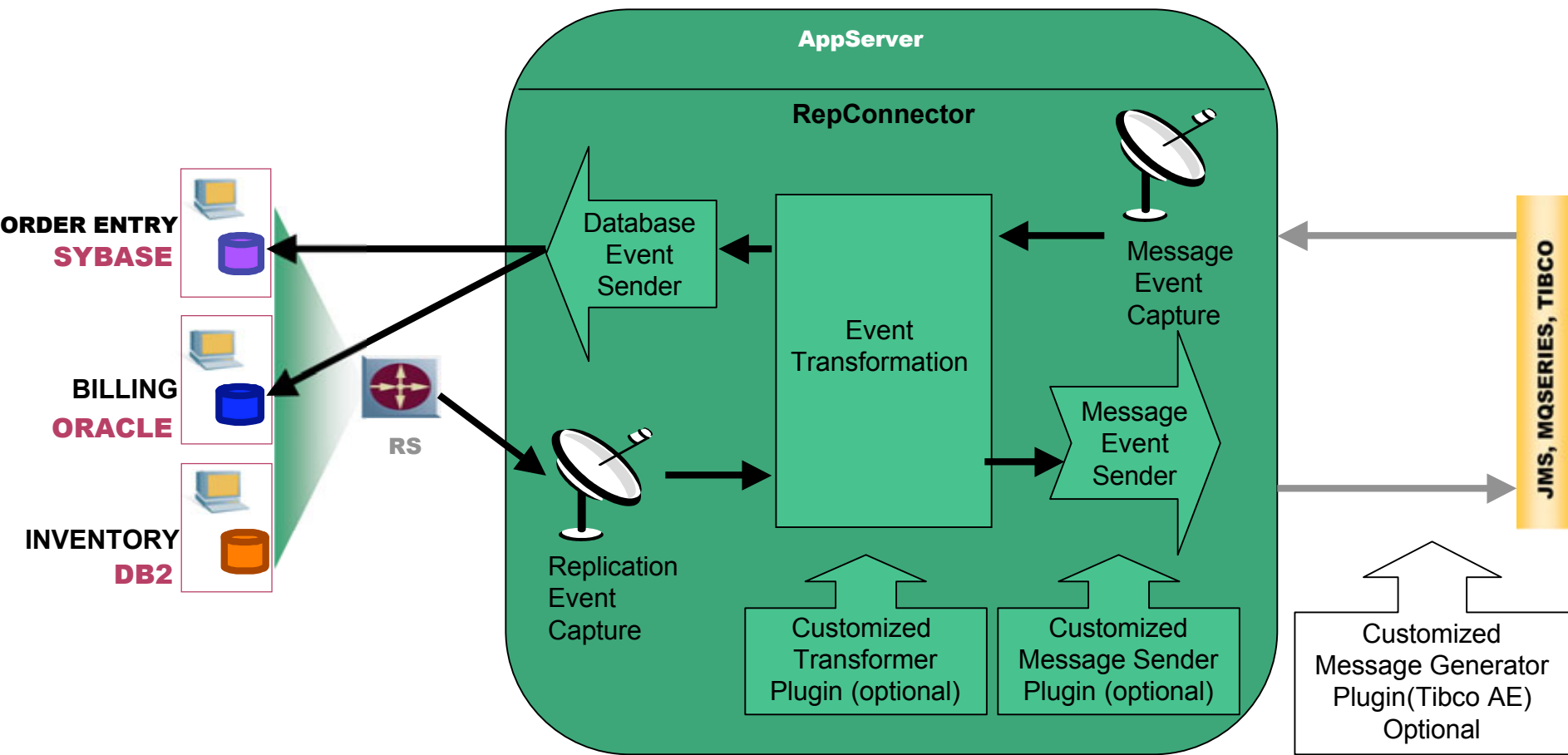
- **Partners with RepServer**
 - **Proven non intrusive event propagator.**
- **Propagates Events occurring in :**
 - **ASE, Oracle, DB2, UDB, Informix, MS-SQL, any Source of Replication Server**
- **To Any Message BUS :**
 - **MQSeries, WSMQ, Tibco RV, Tibco CM, Tibco JMS, MQ-JMS, SonicMQ JMS, EAS JMS, BEA JMS, any JMS 1.1 implementation.**

Message Bus → DB

- **Messages from Any message Bus**
- **Transform messages to SQL commands for ASE or Oracle (8i, 9i) db**
- **Sends SQL to ASE or ORACLE DB using JDBC**
- **Commits message transactions only after committing db transactions**

RTDS for Existing Applications

RepConnector 2.5 Architecture



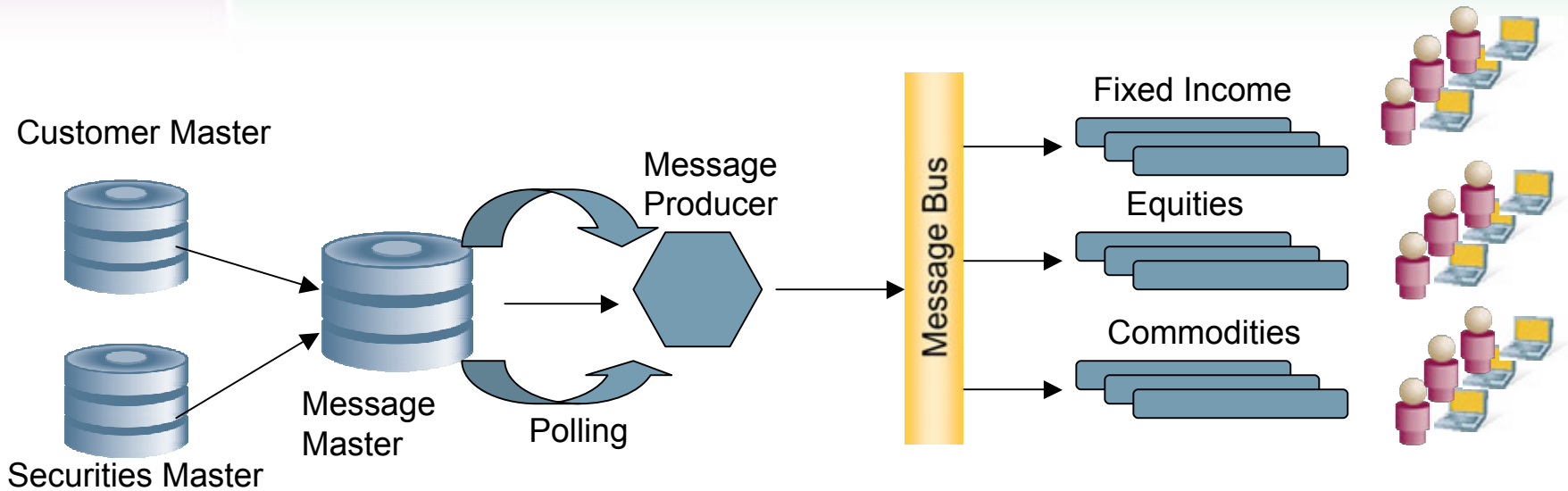
Rep Connector

Output from RTDS

Standardized XML for DB Events

```
<dbStream xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://LISALEWW2K:8080/RepraWebApp/dtd
s/dbeventstream.xsd" environment="sample_repconnector.pubs2">
<tran eventId="104:0000000000001d030004dd0013000004dd00100000">
  <insert schema="authors2">
    <values>
      <cell name="au_id" type="VARCHAR">888-88-888</cell>
      <cell name="au_lname" type="VARCHAR">RepraFirst</cell>
      <cell name="au_fname" type="VARCHAR">RepraLast</cell>
      <cell name="phone" type="CHAR">925-236-7000</cell>
      <cell name="address" type="VARCHAR">1 Sybase Drive</cell>
      <cell name="city" type="VARCHAR">Dublin</cell>
      <cell name="state" type="CHAR">CA</cell>
      <cell name="country" type="VARCHAR">USA</cell>
      <cell name="postalcode" type="CHAR">94568</cell>
    </values>
  </insert>
</tran>
</dbStream>
```

- **Business**
 - Largest trader of fixed income instruments e.g. Treasuries, Govt Bonds
- **Challenge**
 - Manage credit limit risk better across their trading desks.
- **Current Alternative**
 - Poll a consolidated securities and customer master database
 - **User , Account, Credit, and Security changes**
 - Write a custom program to poll for the changes and notify trading desks
 - **Intrusive and requires maintenance of complex custom code**
- **RTDS Solution**
 - Automatically publish changes to Security Master or Customer Master database
 - **Non-intrusive and Proactive**



Requirement:

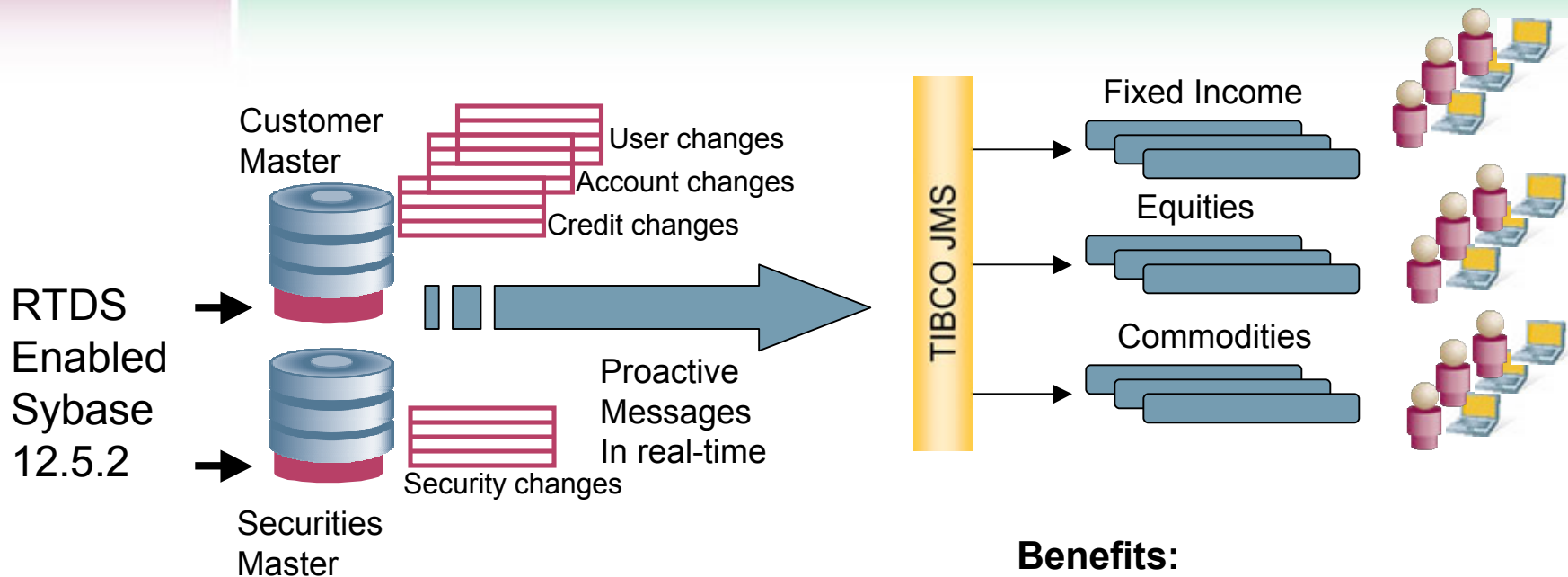
- Automatically prevent trading once a customer credit limit is breached
- Prevent buy side systems from trading when a buy limit is exceeded

Current Solution:

- Poll the message database periodically for credit limits thresholds.
- Poll periodically the message master to check if buy-limit exceeded.

Risks:

- Trading continues even when a limit (credit or buy-limit) is breached because of the non-real time nature of polling
- Message master system heavily impacted due to frequent polling requests for changes.



RTDS Solution:

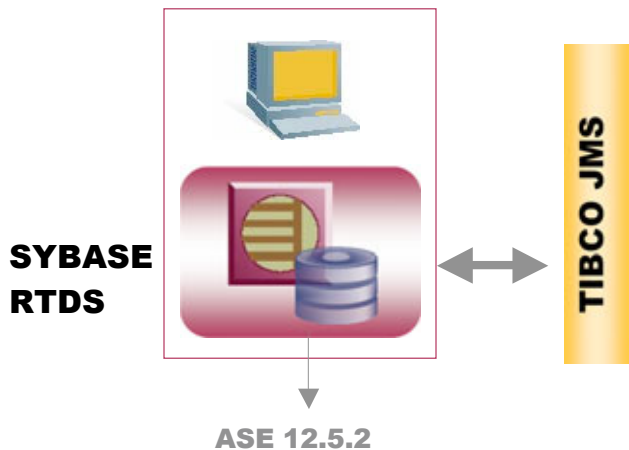
- Account changes are pushed out proactively in real-time
- Security changes are pushed out proactively in real-time

Benefits:

- Trading systems notified in real-time & proactively of limit breaches as and when they happen
- Minimal impact on source systems
- Eliminates custom polling application
- Simplifies architecture

Deployment Architecture

Empowering New Applications



ASE to JMS Natively

Capabilities

- Push events from within client apps
- Bi-directional – write/read from Message Bus
- Can be used anywhere (triggers, stored procedures or ad-hoc SQL)
- Message content can be raw text, binary data or XML
- Content can be created from relational data or application logic

Benefits

- Use SQL to generate events
- Applications developers don't need to learn yet another API
- High Performance message delivery and archiving

RTDS for New Applications

ASE 12.5.2 Real Time Messaging Service

- **Point to Point (Queue) Support in ASE**
 - SQL Built-ins, msgsend() and msgrecv()
 - msgsend() – send a message to a queue
 - msgrecv() – receive a message from a queue
- **Publish and Subscribe (Topic) Support in ASE**
 - SQL Built-ins, msgpublish() and msgconsume()
 - Subscriptions must be registered first with sp_msgadmin command
 - Allows multiple ASE session to publish (and subscribe to) same topic
 - msgpublish() – publish a message to a topic
 - msgconsume() – consume a message from a topic
 - msgsubscribe() – start a subscription
 - msgunsubscribe() – stop a subscription
- **Architecture**
 - Messages received can be stored into database tables or handled by the application logic
 - Allows custom (user defined) properties to be added to the message
 - Property names and property values can be created from user logic as well as from relational data
 - Preserve transactional semantics across the database and message bus
 - if a transaction fails then message won't be published

set transactional messaging full

- Messages sent to and received from the JMS message bus are part of the ASE transaction
- Rollback of ASE transaction will undo work done on the message bus
- Errors encountered on the messaging bus will rollback the ASE transaction

Example #1

```
set transactional messaging full
```

```
begin transaction
```

```
select msgpublish('hello world', 'sub1')
```

```
insert into t values (100)
```

```
select msgconsume('sub1')
```

```
commit
```

Failure of msgpublish, msgconsume, or insert will
rollback the transaction

Example #2

```
set transactional messaging full
```

```
begin transaction
```

```
select msgpublish('hello world', 'sub1')
```

```
insert into t values (100)
```

```
select msgconsume('sub1')
```

```
rollback
```

Rollback will undo msgpublish, insert and
msgconsume

set transactional messaging *simple*

- Messages sent to and received from the JMS message bus are part of the ASE transaction
- Rollback of ASE transaction will undo work done on the message bus
- Errors encountered on the messaging bus will NOT rollback the ASE transaction

Example #1

set transactional messaging *simple*

begin transaction

select msgpublish('hello world', 'sub1')

insert into t values (100)

select msgconsume('sub1')

commit

1. Failure by msgpublish or msgconsume will not rollback the transaction
2. Failure by insert will undo msgpublish

Example #2

set transactional messaging *simple*

begin transaction

select msgpublish('hello world', 'sub1')

insert into t values (100)

select msgconsume('sub1')

rollback

1. Rollback will undo work done by msgpublish and msgconsume

set transactional messaging none

- Messages sent to and received from the JMS bus are NOT part of the ASE transaction
- Rollback of ASE transaction will NOT undo work done on the message bus
- Errors encountered on the messaging bus will NOT rollback the ASE transaction

Example #1

```
set transactional messaging none
```

```
begin transaction
```

```
select msgpublish('hello world', 'sub1')
```

```
insert into t values (100)
```

```
select msgconsume('sub1')
```

```
commit
```

1. Failure by msgpublish or msgconsume will not rollback the transaction
2. Failure by insert will not undo msgpublish

Example #2

```
set transactional messaging none
```

```
begin transaction
```

```
select msgpublish('hello world', 'sub1')
```

```
insert into t values (100)
```

```
select msgconsume('sub1')
```

```
rollback
```

1. Rollback will not undo work done by msgpublish and msgconsume

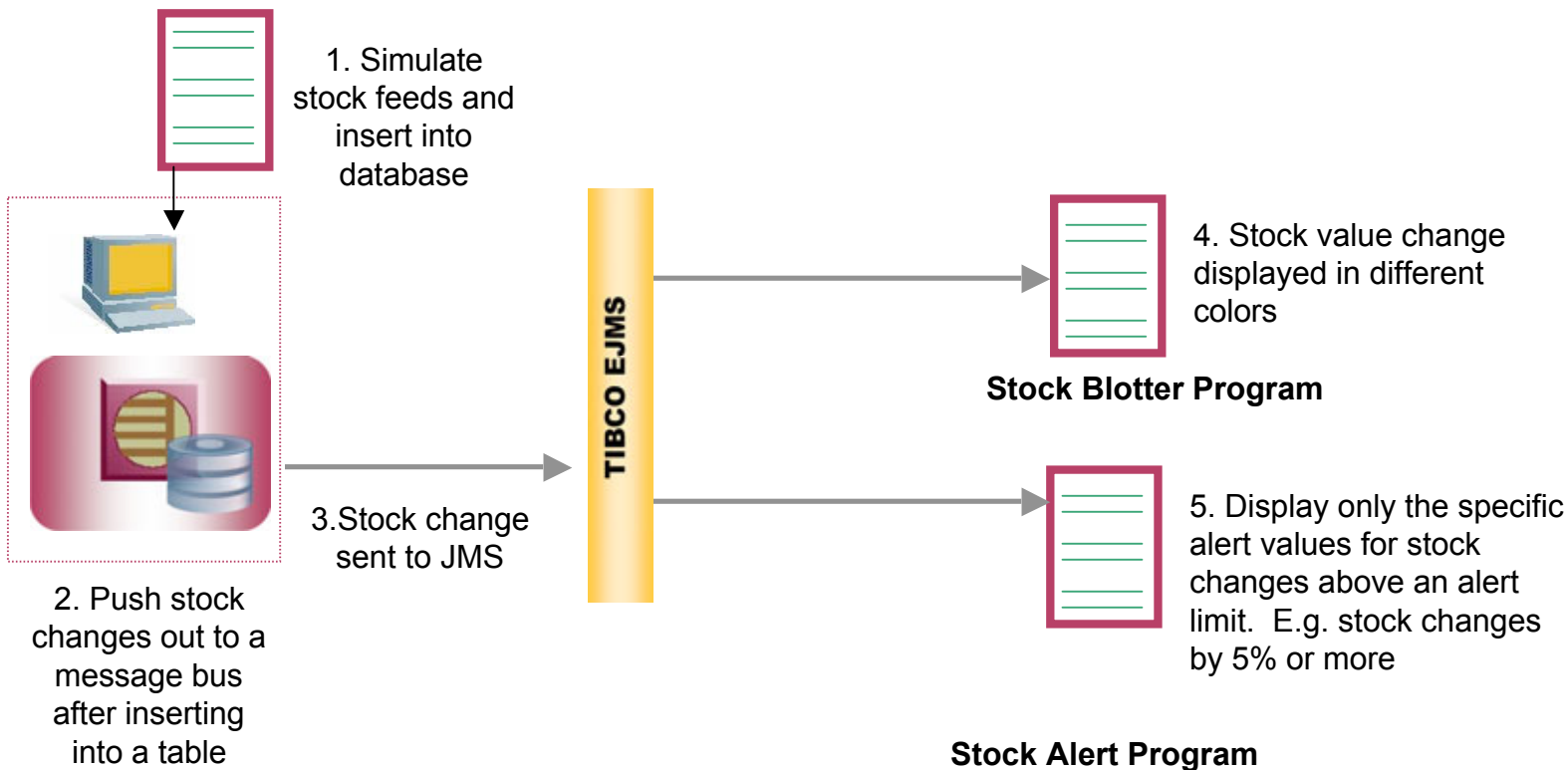
- **Notifications and Alerts without Polling**
 - Notify applications or devices as and when changes happen
 - Deliver enriched event information just when they need to know
 - **Benefits:** Eliminates need to write custom code to capture events

- **Real-time Business Activity Monitoring**
 - Instrument applications without changing application code, or impacting operational systems
 - **Benefits:** Bring legacy systems into the mix for business activity monitoring, for more accurate view of business processes.

- **Improve Real-Time Cash Flow Visibility**
 - Notify trading systems proactively of any change to customer or security attributes
 - Reduce need for end of the day reconciliation's for cash flow view
 - **Benefit:** Get an accurate view into real-time cash flow, enabling better leverage, without impacting existing systems

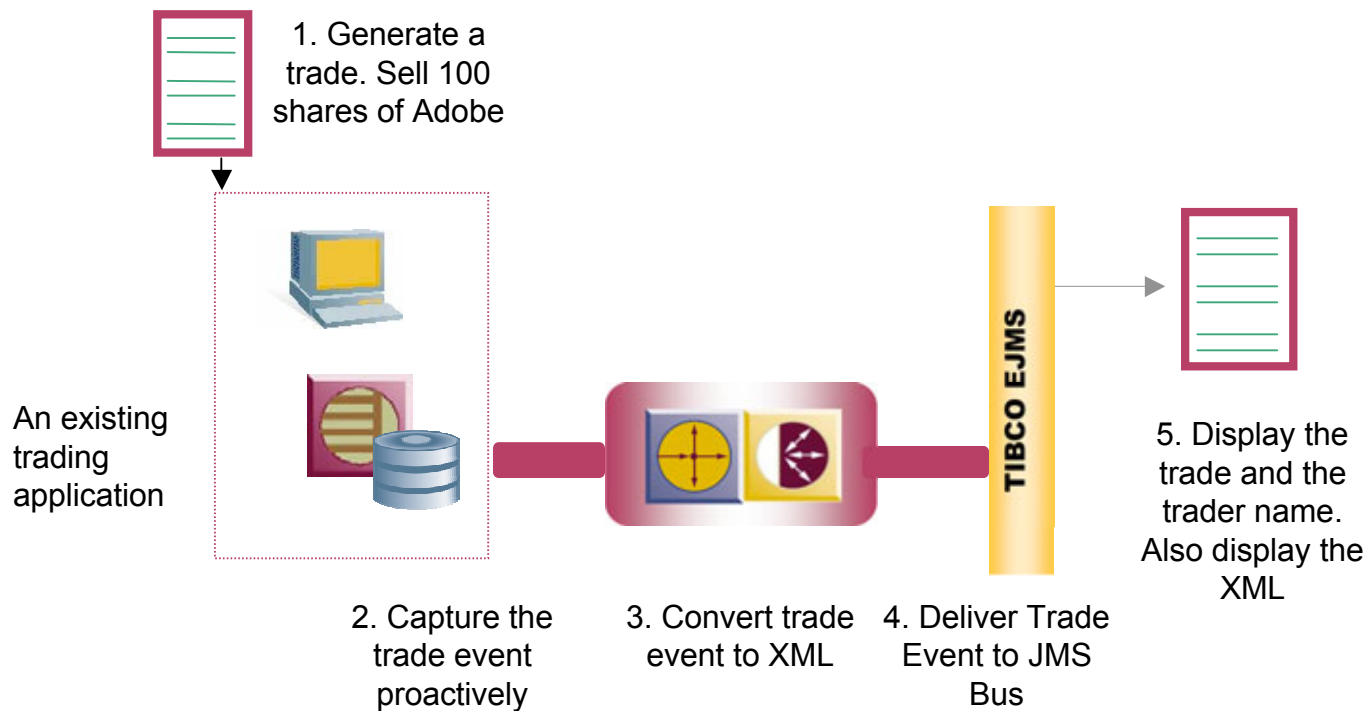
- **Reduce Exposure Risk in Trading Situations**
 - Notify trading systems proactively when a buy order limit is reached for a security or a customer credit limit is reached
 - Reduce time lag between the occurrence of an event and notification of trading systems
 - **Benefit:** Reduce the risk of exposure due to delayed information.

Real Time
Data Services
Enabled ASE
12.5.2



→ PROACTIVELY DELIVER NOTIFICATION OF AN EVENT →

NON INTRUSIVELY CAPTURE AND DELIVER AN EVENT NOTIFICATION



NO CODING, NO APPLICATION CHANGE

Real-Time Data Services

Unique Value & Advantages

Makes Heterogeneous Database Systems Proactive

- Heterogeneous
 - Any database to any message bus
 - Fits in to the existing messaging infrastructure
- Non-Intrusive
 - Capture events non-intrusively
 - Impact free alerts and notifications from existing systems
 - No performance degradation for existing apps
 - No change required to existing systems
- No custom code
 - Code free integration of database to message bus
 - Eliminates need to write custom polling applications
 - Substantially reduced development time/cost

- **Messaging architectures enable sharing information among multiple applications**
- **But databases today store information until applications ask for it**
- **With Sybase Real Time Data Services (RTDS) database events are pushed to applications the moment change occurs**
- **Sybase RTDS enables information flows that are faster, more relevant, and actionable**