EXECUTIVE SUMMARY
• Automated real-time reaction is the key to efficient, data-driven business operations.
• Inference rules enable business analysts to declare how to react to events without concern for when and how those events occur.
• In-memory computing enables high performance stateful rules that can track thousands of events per second and react to situations that evolve over minutes, days, or months.
• Data grid technologies enable horizontal scale out of inference rules to meet big data challenges.

INTRODUCTION
Analytics on historical data give business analysts and decision-makers valuable insights into the operational realities of a business, for example, into optimal staffing levels or into why a power outage was not resolved in a timely manner. However, such knowledge comes only after-the-fact by analyzing data at rest that describes events that have already happened. To be valuable, this knowledge must be fed back into operations. The most competitive organizations know that operationalization comes not just from changing processes and guidelines, but from feeding data directly into automated operational decision-making.

Event processing is the operational counterpart to business analysis of historical data using business intelligence tools. Event processing is a strategic imperative for organizations that aspire to achieve competitive advantage through providing better customer service, reducing costs, and improving visibility into business operations.

Real-time operationalization of business insight means spotting patterns on real-time data (data in motion), so action can be taken. Automated action can stop a customer from becoming an ex-customer, or it can contain a power outage. Automated action can maintain regulatory compliance and provide an audit trail. Automated action can act on an opportunity while it is still available.
Opportunities and threats are not the result of one stray incident. One dropped call is not likely to make a customer switch services. Knowing your customer requires seeing the whole of their experience so you can put them, and what is happening for them, in context. Power equipment would have generated enough symptoms at different points in time that if carefully detected and correlated would lead to timely preventive maintenance. Organizations that can correlate events over time can take preventative action by applying to these events historical knowledge gained from analytics on data at rest.

Inference rules provide an excellent mechanism for organizations to operationalize the knowledge gained from analytics on data at rest, to correlate data in motion, and take action before an opportunity passes. This paper discusses TIBCO’s approach to event processing with inference rules.

THE POWER OF STATEFUL EVENT CORRELATION

Any kind of meaningful correlation of events would typically involve maintaining the state of events over a period of time. This could mean updating a field with every new event, such as a customer’s purchases over a six month window to keep track of loyalty eligibility. Or it could mean storing a history of a certain number of transactions over a period of time, such as a list of items that the customer has purchased over the last six months to identify a purchase profile in order to make relevant offers. Not only is it important to store the state of events, in most situations, it is also important to store the state of events in a way that can be saved as well as retrieved very quickly for timely responses to new events. It is therefore necessary for event processing applications to have:

a Stateful Data Structures - Data structures that can be used to save the state of events, either through summarized information or a detailed history of transactions related to an entity over a period of time. This would also entail storing complex hierarchical relationships between entities such as items ordered by a customer in every order they placed in the last six months.

b Memory Based Storage - In-memory mechanisms for quick access to the stateful data for correlation with incoming events to decide if action is necessary is also needed. Traditional relational database storage is proving to be not fast enough for today’s real-time application needs. Therefore, alternative technologies, such as a distributed in-memory data grid (IMDG), or shared-memory data storage, need to be explored.
DECLARATIVE PROGRAMMING AND FORWARD CHAINING

Event correlation is a very complex art, and it is often not possible to predict a sequence of events. Therefore, the rules for correlation need to be independent of any predictable sequence of execution. It is therefore difficult, if not impossible, to use procedural programming paradigms such as business process management or process orchestration to perform event correlation. Any meaningful event correlation framework needs to support a declarative programming paradigm, where the users are able to define rules without caring about a particular sequence of execution.

Users should be able to declare inference rules in the form “If event A comes in, and the following conditions are satisfied, then perform the following actions.” The conditions could be from the data in the incoming event, from the state of past events, or a combination of the two. For example, an inference rule could be used to specify a pattern such as “If the customer is trying to make a withdrawal using an ATM card and has made three withdrawals in the last three days, but the average ATM card usage for this customer is twice a month in the last three years, the transaction could be fraudulent and therefore the customer’s identity would need to be verified before letting the transaction go through.” Here the incoming event is being correlated with past transactions by the same customer to compare the recent transaction rate with the historical transaction rate stored in the form of stateful information captured from past events.

A single event could lead to multiple independent actions. For example, when a customer is trying to make an online purchase using their credit card, the retailer could:

- Initiate a fraud detection check to ensure the customer is not using a stolen credit card
- Evaluate if the customer is eligible for a loyalty discount based on their category and purchases over a period of time
- Check whether the order causes inventory to fall below threshold levels, and if needed, initiate a workflow for replenishing inventory for the affected items
- Evaluate the best way to source and ship the order based on inventory levels, demand projections for regional warehouses, and shipping costs
- Look at the customer’s purchasing patterns over a period of time and correlate these findings with perishable inventory to generate offers while the customer is still interacting

All these event correlations and resulting actions can be captured using independent rules that could be triggered in no particular sequence. What is important to note is that a particular rule’s actions could result in state changes that could in turn lead to the conditions of other rules triggering actions. This can continue in a domino effect that will stop only when no more rules evaluate to true. It is therefore important for an event correlation engine to support forward-chaining of rules where every time the state changes, rules are reevaluated.

The concepts of declarative programming and forward chaining are very powerful when it comes to event correlation and form the foundation of inference rules.
CONFLICT RESOLUTION
In a system of event processing where events are correlated using declarative and forward-chained inference rules, the sequence of execution of independent rules is determined by the event processing engine at runtime. The engine evaluates all rules where the incoming event is declared a participant. As the sequence of execution is not deterministic, it is possible for the outcome of various rule actions to also become non-deterministic. Therefore, if two rules can lead to conflicting outcomes, a mechanism for conflict resolution is needed. This can be done by letting application developers assign priorities or ranks to rules, so the engine knows which rule to execute first in case the conditions of two rules are satisfied in parallel.

INFERENCE RULES AND TIBCO EVENT PROCESSING
Back in 2004, TIBCO pioneered event correlation using inference rules. It created an event processing platform based on the RETE algorithm and an engine that would execute and forward chain inference rules declaratively.

The TIBCO Event Processing platform supports a stateful inference rules engine that allows capturing the stateful information from events into data structures called “concepts,” which can have simple data types, as well as array data types and history-enabled fields. Concepts can also inherit from other concepts, contain other concepts, or have references to other concepts.

While the TIBCO Event Processing platform does support the notion of maintaining state in a single JVM, in real life scenarios where event correlation is needed on a lot of historical data, it no longer becomes possible to maintain the state in a single JVM. The TIBCO platform therefore offers two different models for maintaining state:
IMDG MODEL
One model is based on a bundled distributed in-memory data grid that allows distribution of the stateful data across a number of processes running on a number of commodity machines. The data stored in the IMDG can be replicated in-memory to achieve fault tolerance and also be persisted centrally on a database or decentralized on local storage media such as SSDs.

SHARED MEMORY MODEL
The other model is one based on running multiple JVMs on a single powerful box using its shared memory to maintain state. This model allows for replicating the shared memory on one box to one or more boxes that could even be spread across geographically dispersed data centers. Replication is based on partitioning of the data set between machines as opposed to the consistent hashing based transparent distribution in the IMDG model. Moreover, as opposed to the IMDG model, the shared memory model supports lockless transactions.

Inference rules in the TIBCO Events Processing platform are fully declarative with the option of forward-chaining. Inference rules are comprised of three sections:

- Declaration Section – Describes the participants in an inference rule.
- Condition Section – Provides one or more conditions that will be evaluated by the engine when an event triggers a rules evaluation cycle, referred to as a “Run To Completion (RTC)” cycle. A rule gets triggered only if all the conditions evaluate to true.
- Action Section – Specifies actions that will be undertaken when the rule triggers. Typical actions include:
  - Sending outbound events to external systems
  - Creating, modifying, or deleting concepts
  - Executing catalog functions that are a rich library of pre-packaged functions that ship with the product, custom functions that are created by the users using the exposed Java API, and re-usable rule functions written using the rules language that can make use of catalog or custom functions

CONCLUSION
TIBCO, the company that pioneered the notion of the Two-Second Advantage®, recognizes the importance of event processing in real time for achieving competitive advantage. It provides inference rules as a powerful tool in the event processing arsenal to correlate events with stateful and historical data. Event correlation operationalizes the knowledge gained from analyzing historical data for repetitive enforcement (patterns) to spot opportunities and threats before the fact. Inference rules are developed using declarative programming and forward chaining and provide conflict resolution mechanisms that supply deterministic behavior. In addition to inference rules, the TIBCO Event Processing platform provides a number of other paradigms.

For an overview of event processing platforms and how businesses are deriving competitive advantage from them, read the whitepaper, Event Processing with TIBCO BusinessEvents, available along with other resources at www.tibco.com/BusinessEvents.