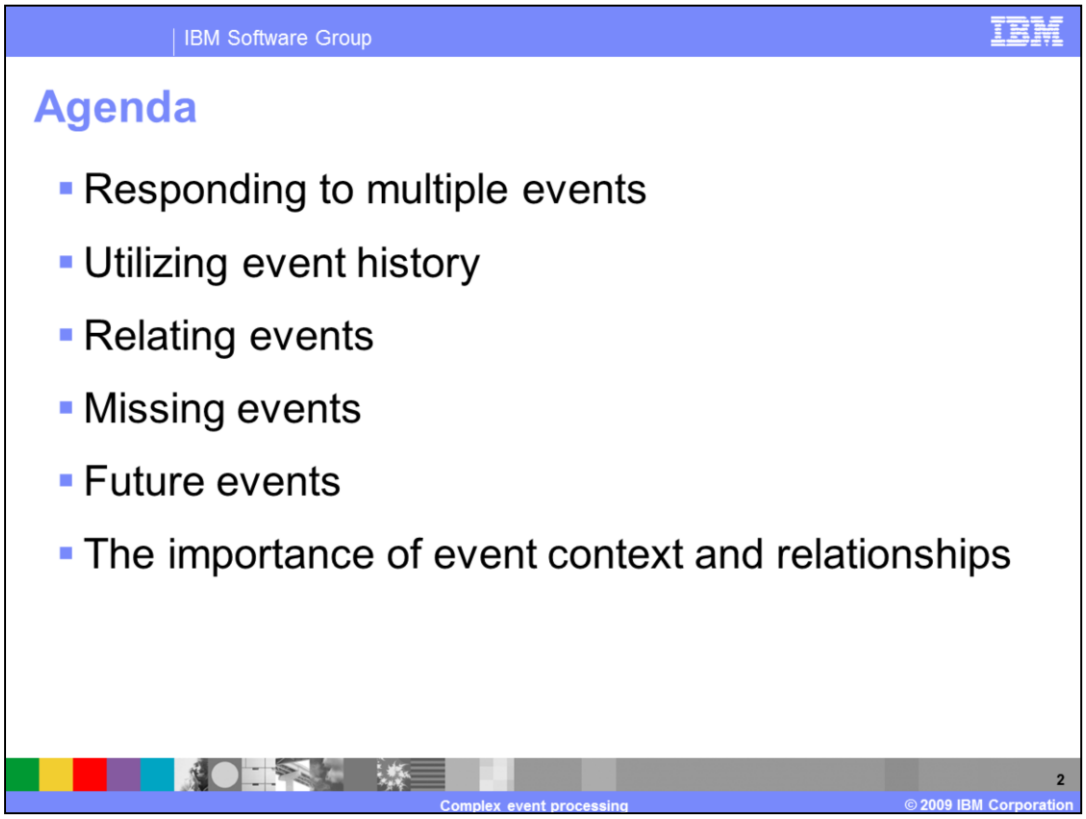
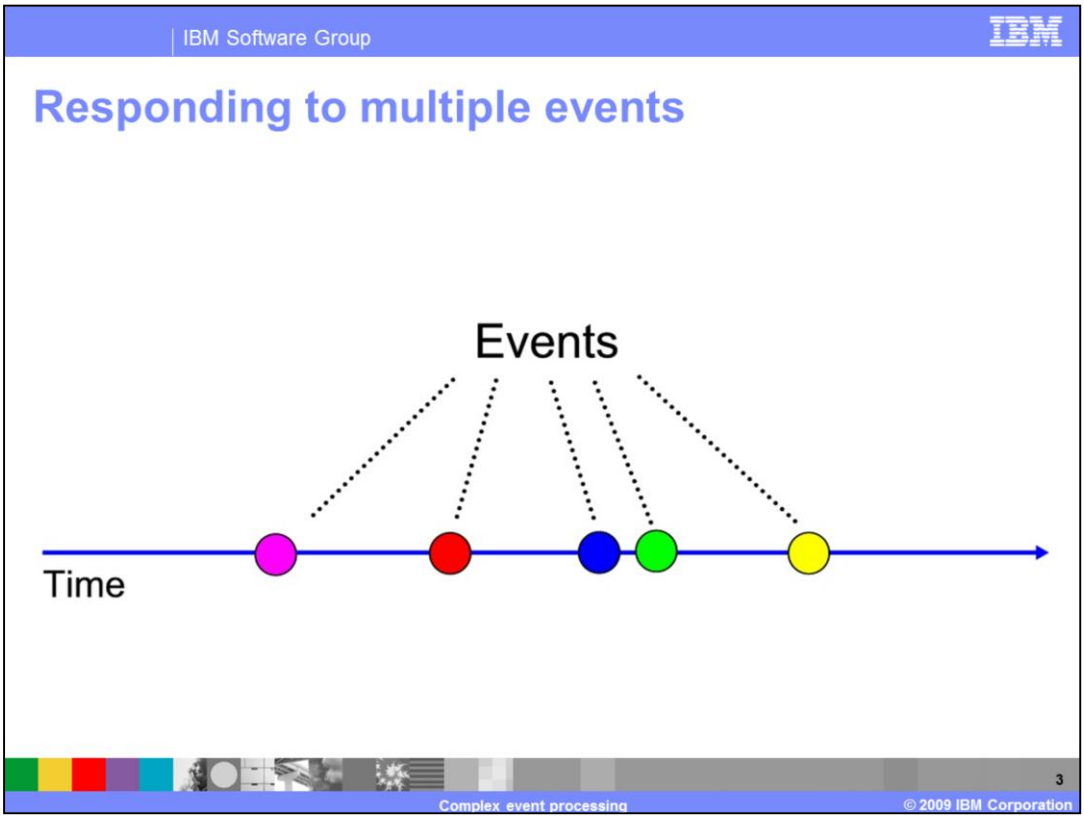
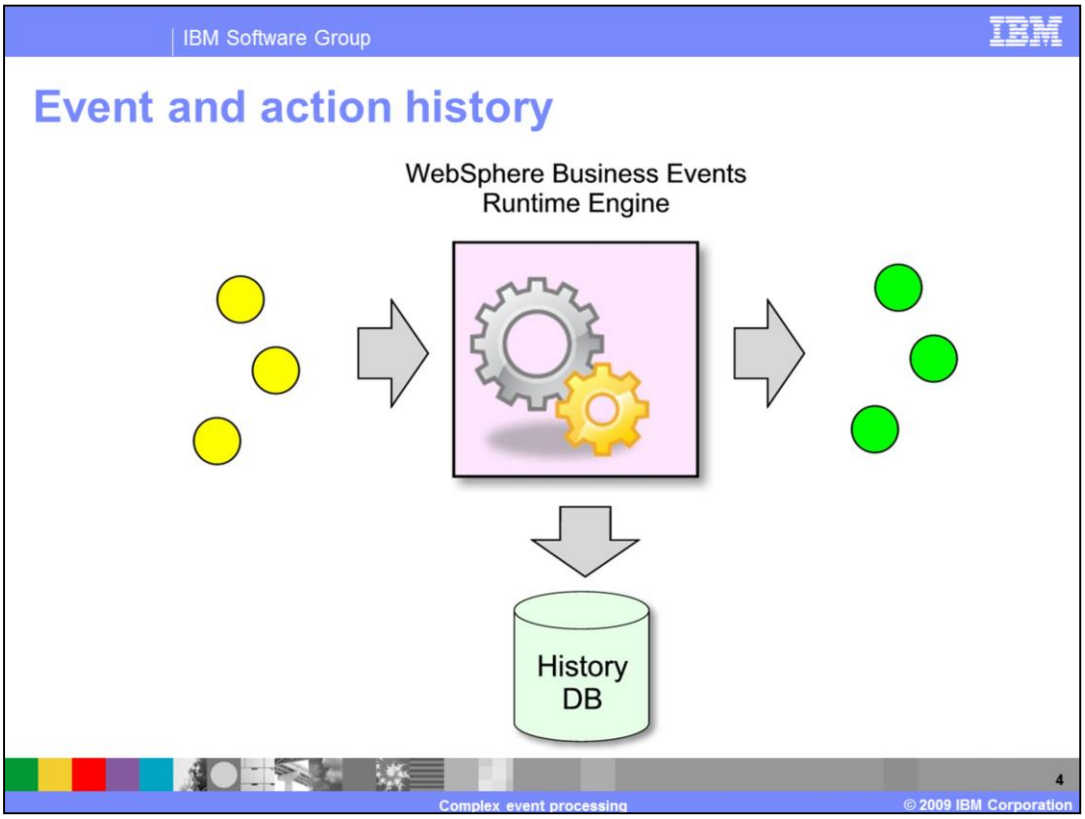
This presentation builds on the knowledge you have gained to allow you to understand how the components of a WebSphere® Business Events solution can be combined to provide a true complex event processing solution.



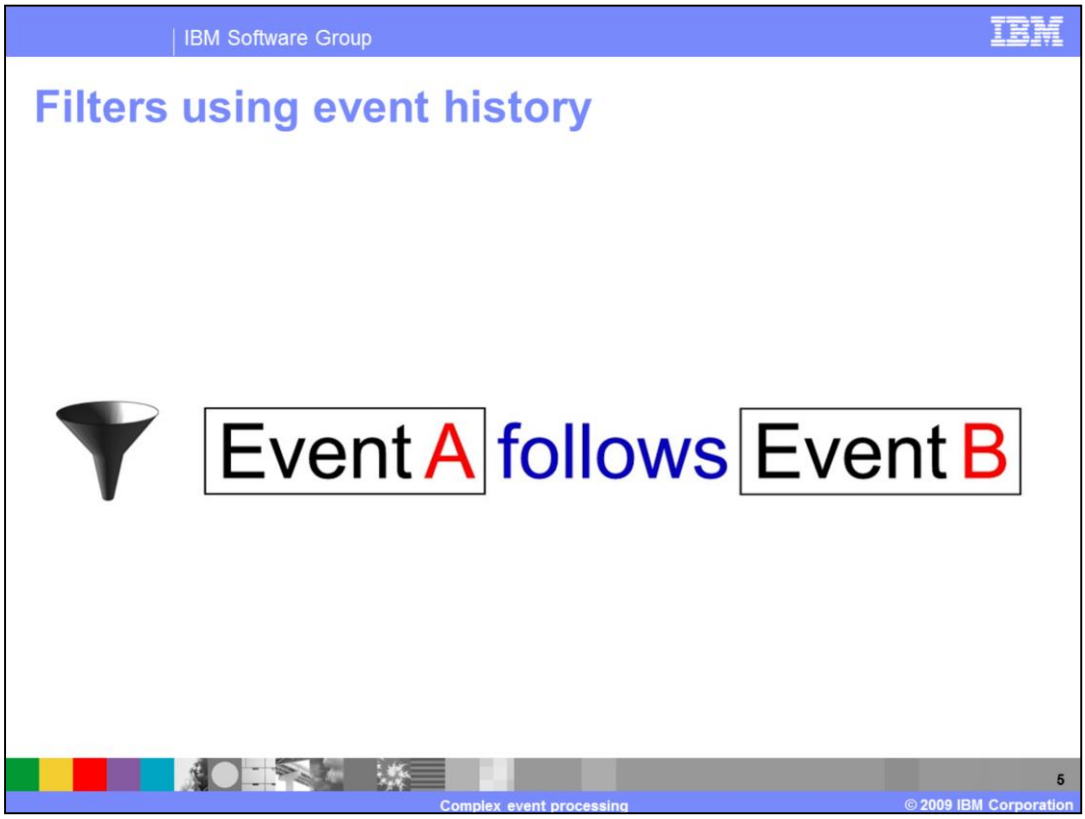
The goal of this presentation is to help you understand complex event processing. Complex event processing is the handling of multiple related events. There are several key points; event history has to be saved, filters relate events together and the solution can look for both omitted events and potential future events.



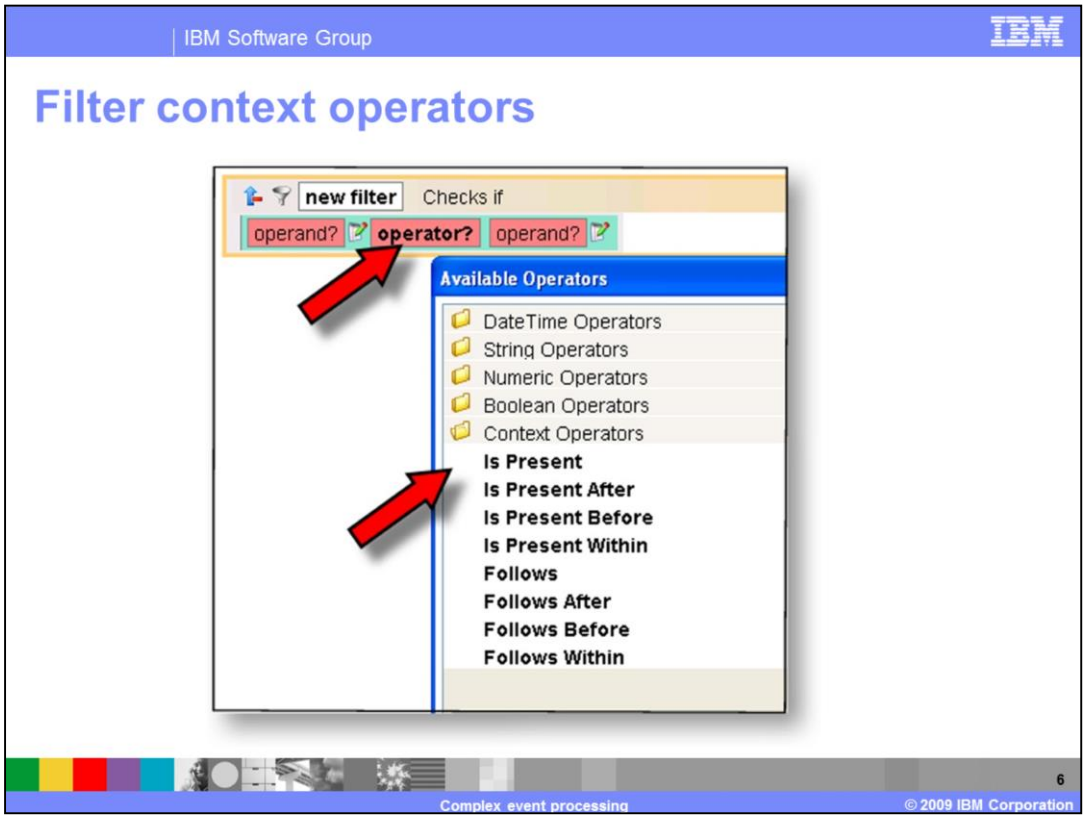
When an event is presented to the runtime, you can issue an action as a result of that event having arrived and being detected. This ‘one event’ to ‘one action’ mapping is useful but by itself limited. The true power of WebSphere Business Events originates from the idea of seeing a sequence of events and executing actions based upon the detection of **multiple** events. When an event is detected, you immediately know two things; **what** kind of event it was and **when** it occurred. The diagram illustrates a sequence of different events arriving over a period of time.



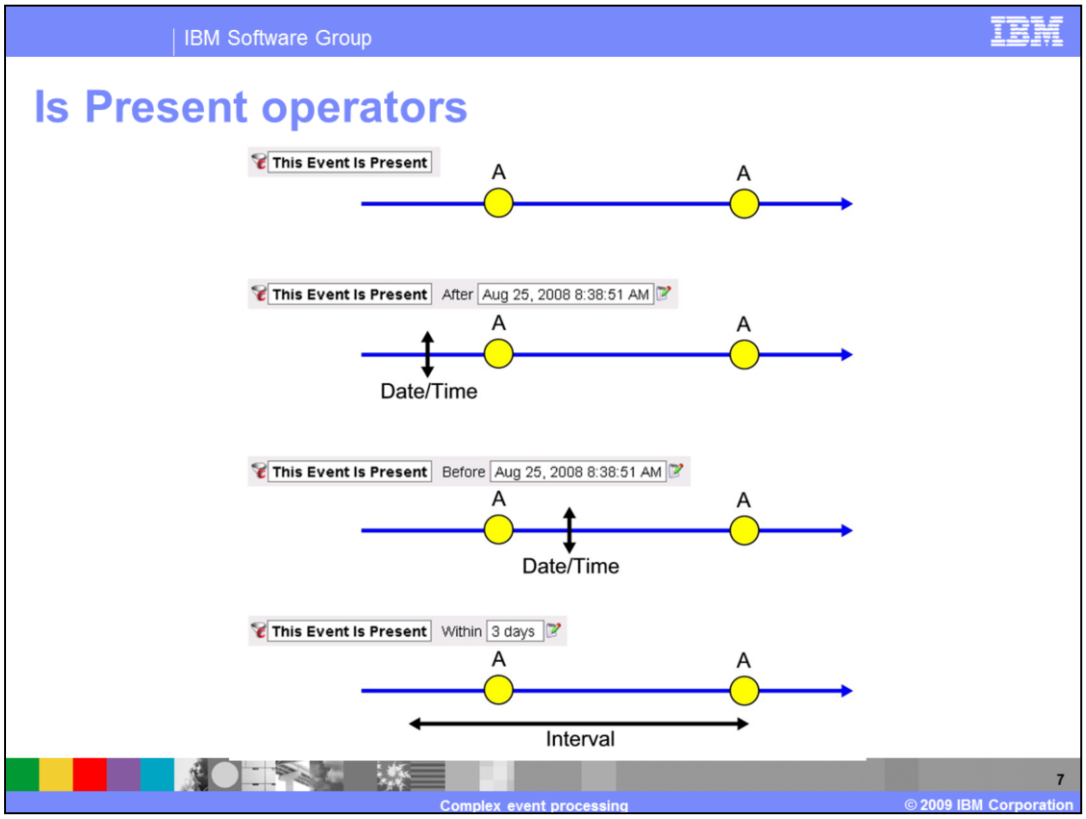
When an event is detected by the runtime or when an action is generated, a record of the event or action is written into the runtime's history data store. This allows subsequent determination of which events have been seen, when these occurred and what actions have been generated.



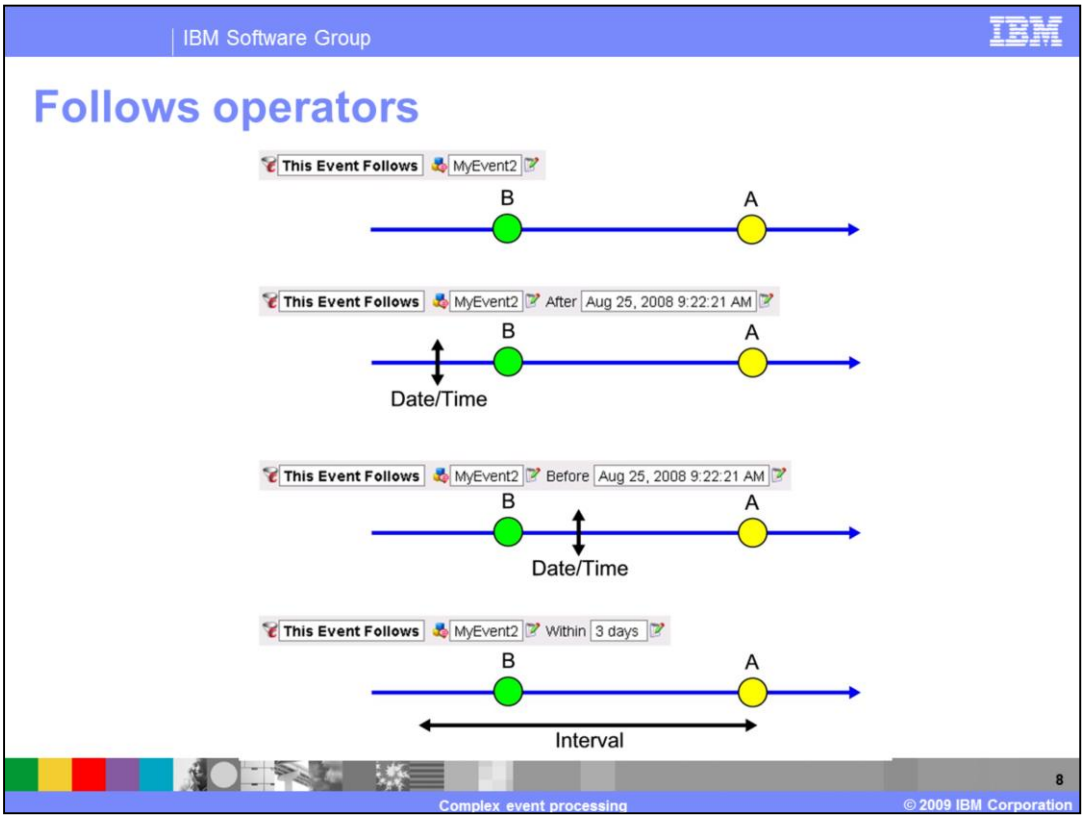
Since WebSphere Business Events is now aware of the preceding history of events and actions, the filter expressions can use and exploit this knowledge to determine filter results based on preceding events. Remember that a filter is an expression that results in a true or false outcome, and if true, can allow an action to be taken. Using this notion, you can now define more interesting filters which are defined to be true if one event follows another.



When a filter is created in the design tool, it asks for an operator that should be used to calculate the result of the filter expression. One of the sets of operators available to the filter designer is called “context operators.” by selecting one of these, the filter is instructed to use the history of preceding events and actions in evaluating the filter expression. To access this set of operators, click the operator box in the filter editor and from the operator selection dialog, expand the context operators folder.



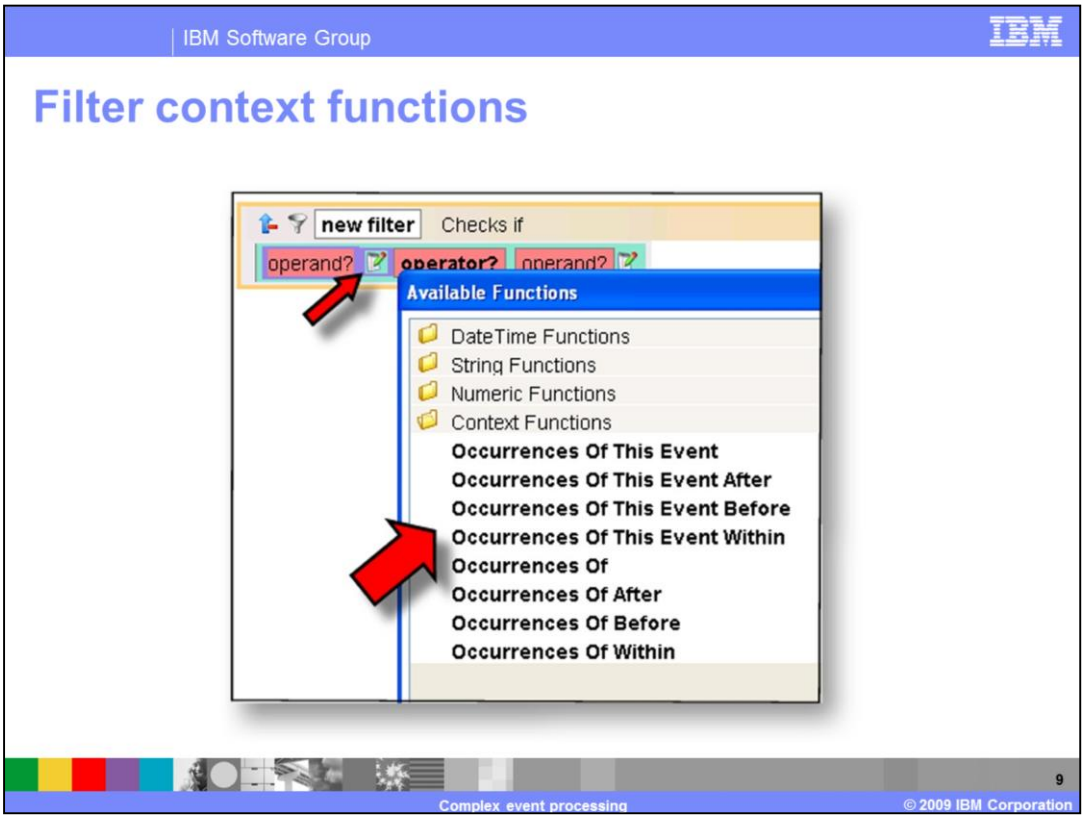
In the filter context operators, two distinct groups are available. One of them begins with the name “is present”. These operators use the event that triggered the interaction set rule and ask questions about previous occurrences of this type of event. The first of these operators is “is present” which asks if any instance of this event has previously been seen. Another operator is called “is present after” which asks if an instance of this event has previously been seen **after** a given date or time. Another operator is called “is present before” which asks if a previous instance of this event has been seen **before** a given date or time. Finally, there is the “Is Present Within” operator which asks if an instance of this event has previously been seen within a supplied time interval before the current event.



Similar to the “is present” operators, a second set of operators are also provided which start with the text “follows”. These operators look for a preceding event which might be **different** from the current event. The previous set of operators looked for a previous event that was the same type as the current event.

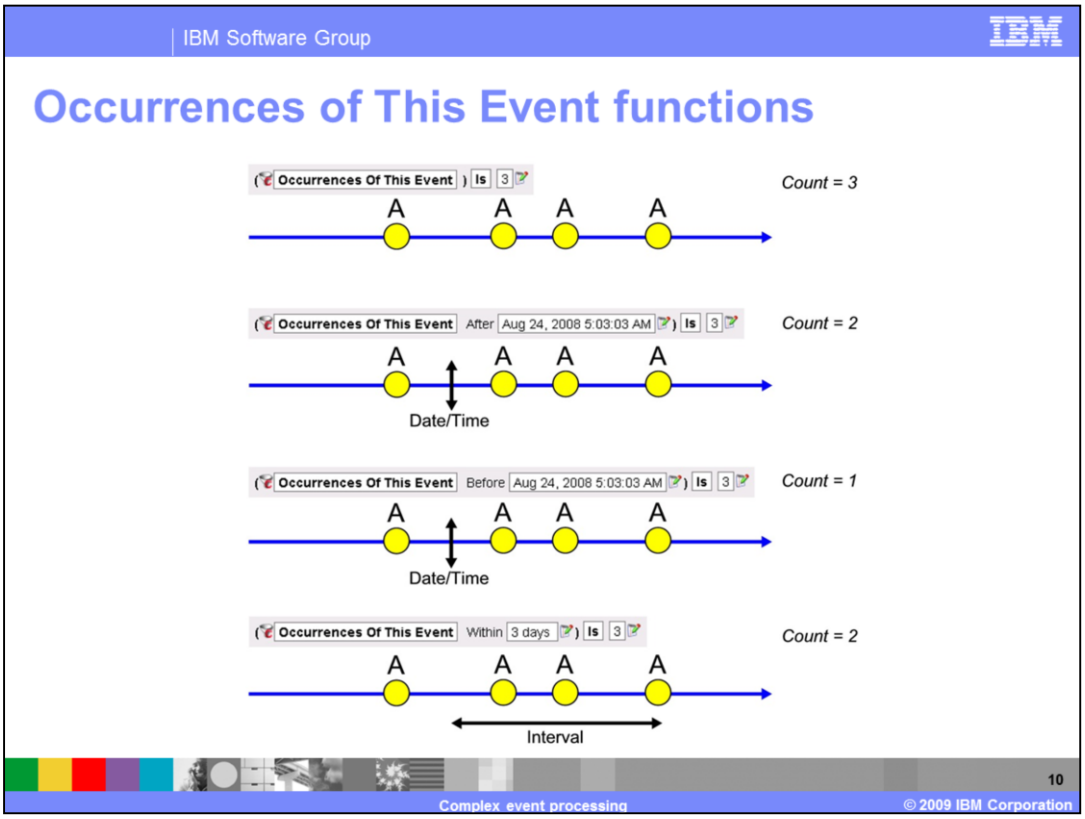
The first operator in the “follows” class of operators is called “follows” and looks for a named event having occurred any time in the past. A second operator is called “follows after” and this looks for a named event having occurred previously **after** a given date or time. The next operator is called “follows before” which looks for a named event having occurred before some time in the past. Finally there is the operator called “follows within” that looks for a named event having occurred within some defined interval before the current event.

The “follows” operator can also be applied to actions.



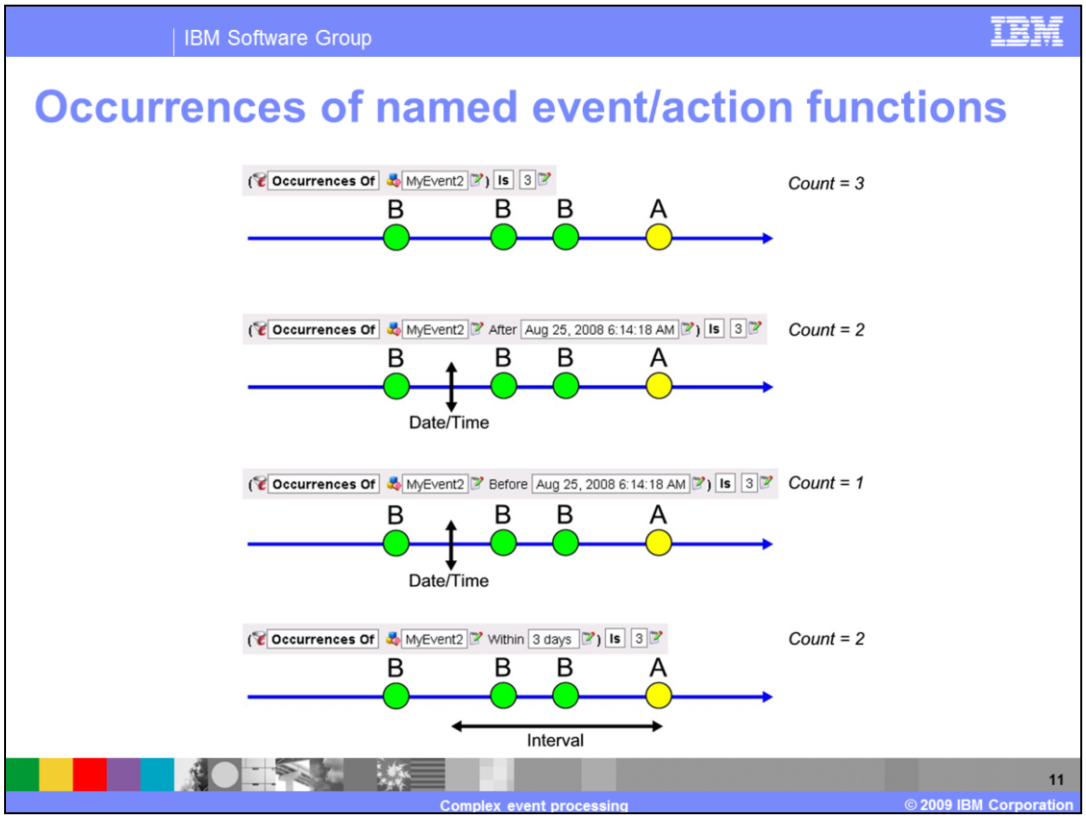
An additional set of context related filter items are available as predefined functions. From within the operand menu there is an item called “select a predefined function”. When selected a dialog containing a list of such functions is presented. Under the context functions folder a set of functions related to context can be selected. Each of these functions returns the number of occurrences of an event or action within different time periods. There are two classes of occurrence functions, one which returns the count of the occurrences of the current event and one which returns the count of occurrences of the named event or action. These are described in more detail shortly.

By counting of the number of occurrences of an event, interesting expressions can be created. For example, the number of failed login attempts or the number of stock purchases over a given period of time (assuming these were both modeled as events).

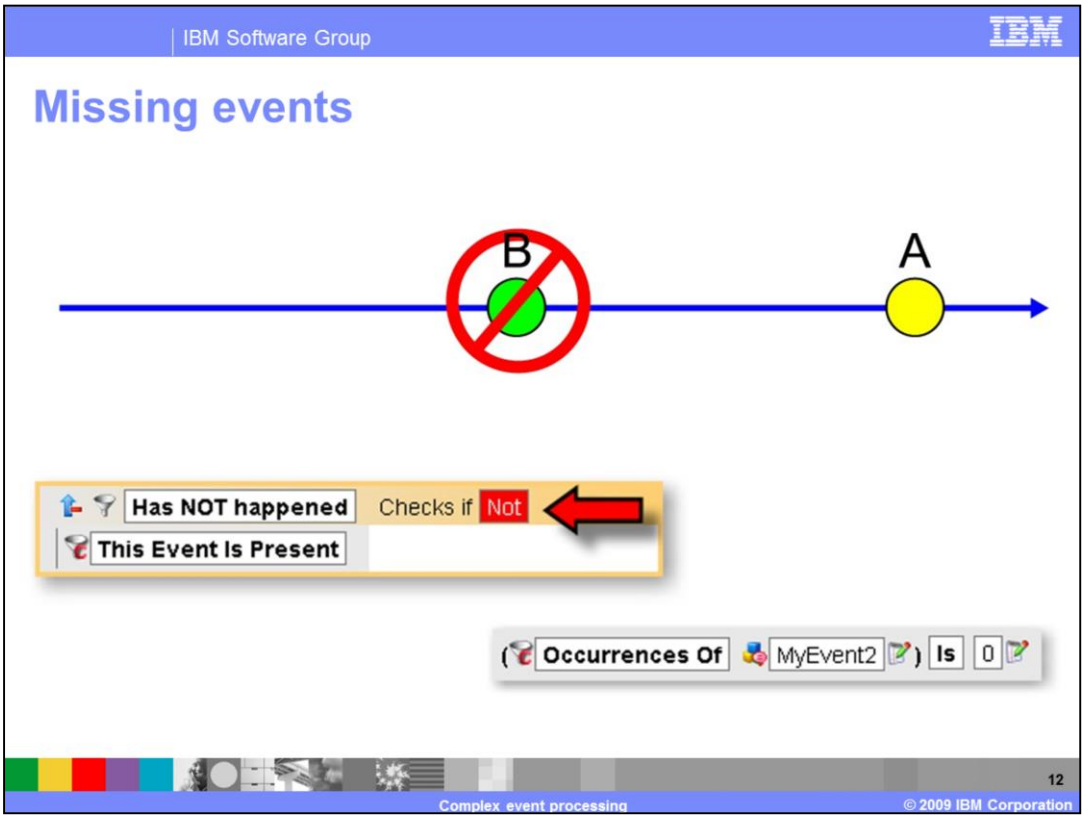


The first set of occurrence functions uses the current event that triggered the interaction set as the type of event to count. There are four functions. One function counts all events of the given type. A second function counts all events of a given type that have been seen after a specific date and time. A third function returns the count of events of a given type before a specific date and time. The last type of function returns the count of a given type within an interval from when the current event was seen.

It is important to note that this set of functions do **not** include the current event in the count. Even though this event is of the searched for type, the functions are **exclusive** of the current event.



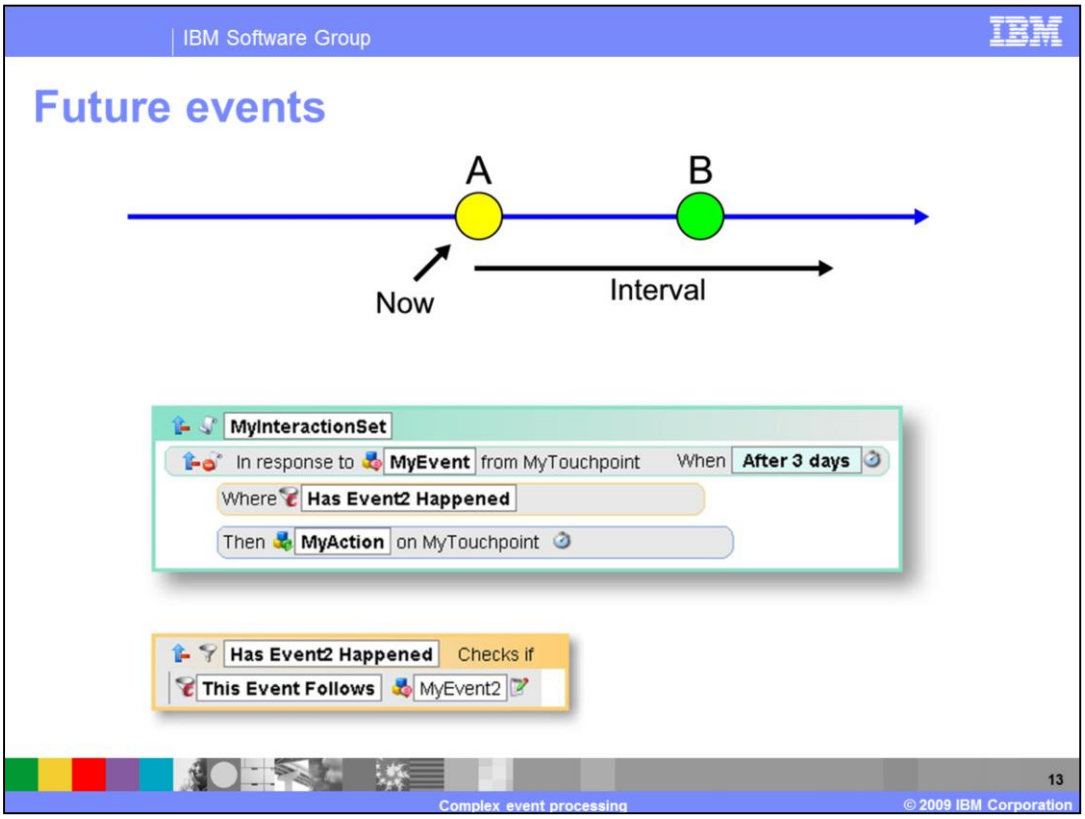
The second set of functions also counts occurrences but these functions can take a named event or named action as a parameter and return the count. There are again four instances of the function. The first returns the total number of the named event or action that has been seen. The second returns the count of the named event or action seen after a given date and time. The third function returns the count of the named event or action seen before a given date and time. The final function returns the number of occurrences of the event or action within a given time interval before the triggering event was seen.



Although the detection of the occurrence of preceding events is highly valuable, detecting the absence of preceding events can be just as valuable. For example, the detection of a cash payment without a preceding authorization can trigger a halt to the payment. In this case, the missing authorization event is crucial.

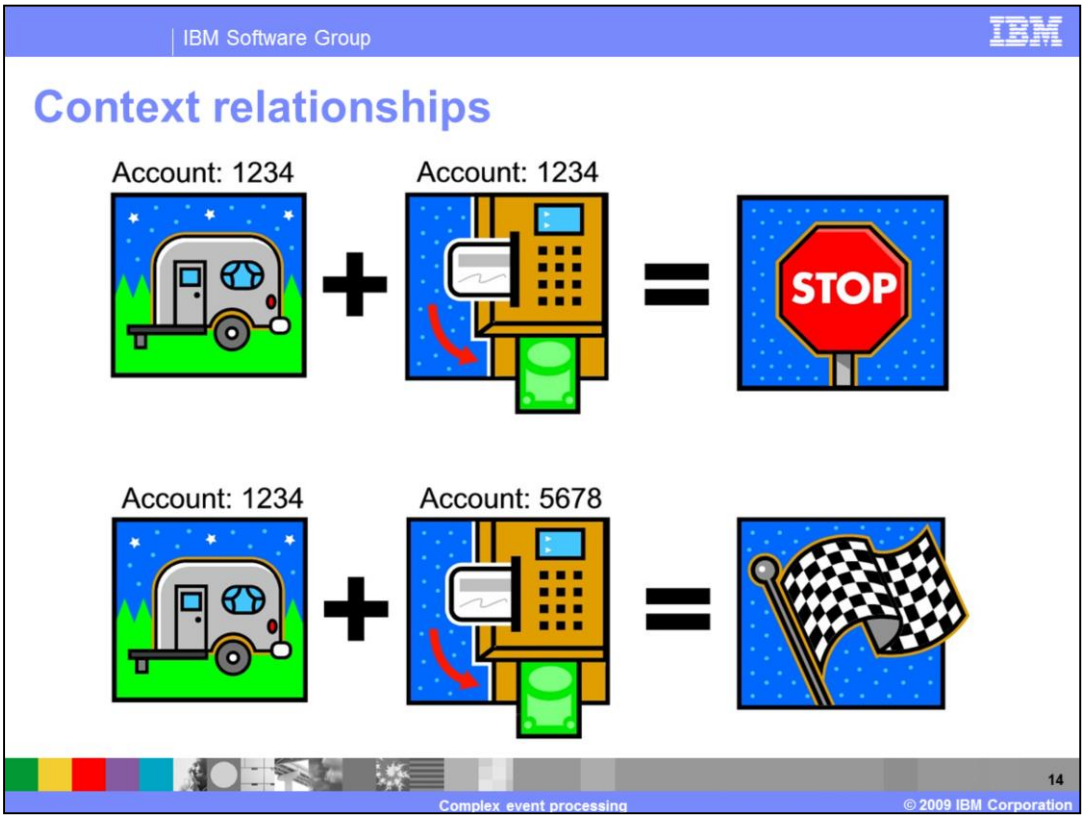
Since a filter expression can be used to detect the occurrence of a preceding event and a filter can be logically inverted with the **NOT** operator, this can easily be used to determine the absence of an event. For the occurrence functions, a count of zero indicates that no events of the given type in the given time range have been seen.

The diagram illustrates both an inverted filter and an occurrence function asking if there were zero occurrences.

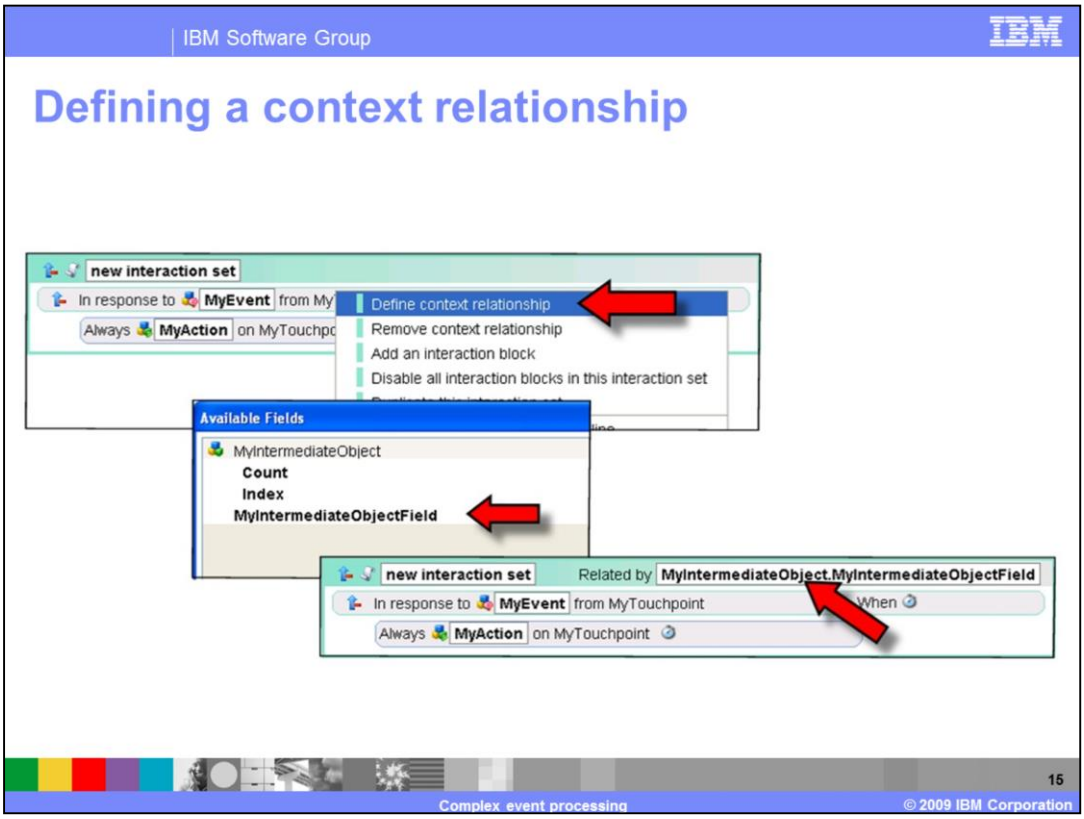


A simple but powerful combination of capabilities can also be used to look for the presence or absence of future events. When defining an interaction block, a delay can be associated with the execution of that block, after the specified event arrives. When the delay has passed, only then are the filter expressions evaluated and possible actions executed. Combining this interaction block capability with the complex event processing, you can start to formulate rules that can be used to match for future event combinations. For example, the arrival of an event, wait for a period and then ask if a second event has now arrived. One example of this might be if an event has been generated saying that goods have been shipped to a customer. In this case you can delay the processing of that event for twenty eight days and then ask if a corresponding payment event has been received. The absence of such an event might cause you to contact the customer for the expected payment.

The diagram illustrates the delay of processing an event for three days after which a check for a subsequent event that is expected to arrive after the first event is performed.

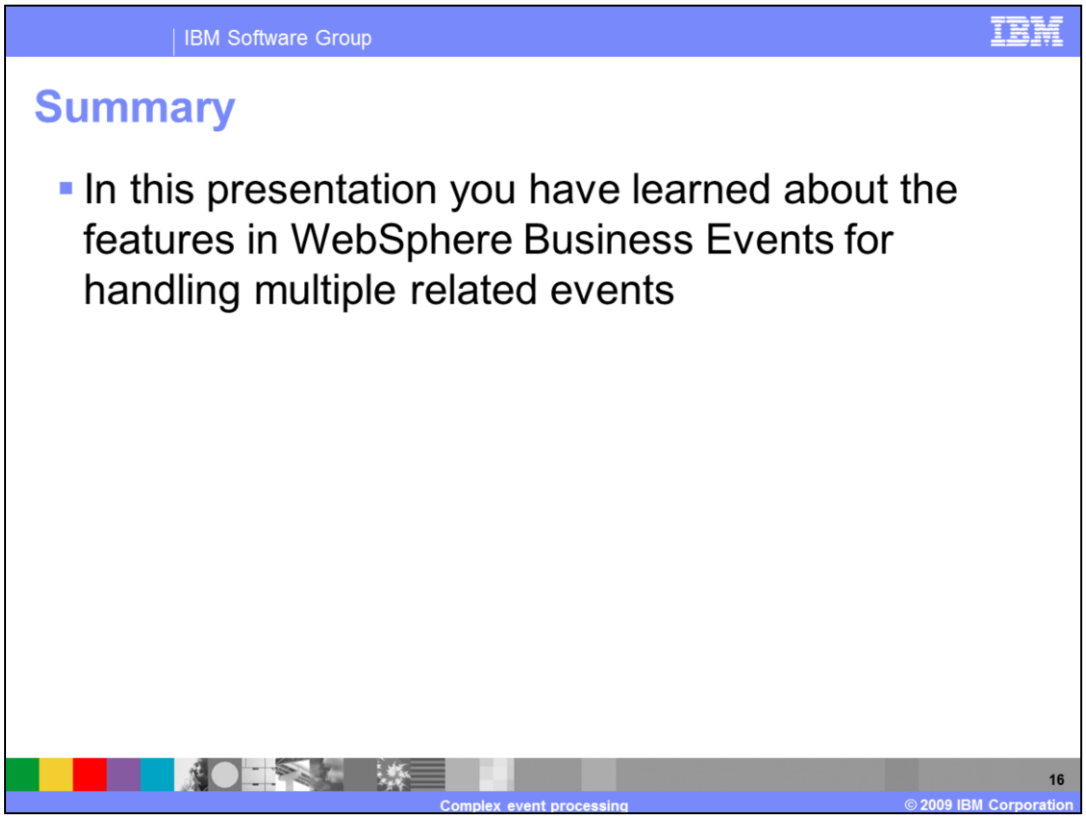


To ask if one event has happened before another event is not that useful. What you really mean to ask is if one event that is **related** to the current event has happened before. For example, think of the scenario where a change of address followed by a large account withdrawal is an indication of a potential banking fraud. In this case you need to also recognize that these events have to happen against the **same** account. Events happening against different accounts are not important. This then tells you that there has to be something in an event that allows you to group or relate events together. This relationship is termed the context of the event. WebSphere Business Events allows you to configure a context relationship.



A context relationship is defined with respect to an interaction set. The context relationship allows you to select a single field from the intermediate object that is populated from the arriving event. The value of this field is then used to relate sets of events together. For example, if a field in an intermediate object is used for a bank account number, then events that supply that intermediate object can be related together through the value of the account number. Events against the same bank account will have the same value for this intermediate object field.

To define a context relationship, right-click in the interaction set for the associated event. From the pop-up menu, select “define context relationship.” A dialog opens which allows you to select a field from the associated intermediate object fields. Upon completion, the relationship is shown as an attribute in the interaction set editor.



In this presentation you have learned about capabilities in WebSphere Business Events for handling multiple related events. These included the notion of event history, how filters relate events together and how to look for both omitted events and events that might happen in the future.

The presentation also covered the details about how to use the WebSphere Business Events: Design tool to define filters that use operators in the filter expression for processing related events. You also learnt how to specify context relationships in an interaction set and how these can be used to relate events together.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.