



C15

Tivoli Enterprise Console State Correlation Engine Rule Building Workshop

Jerry Saulman, Tivoli Software Global Response Team



San Francisco, CA

May 7-10, 2007



| IBM Software Group

Tivoli Enterprise Console

State Correlation Engine Rule Building Workshop

Tivoli software



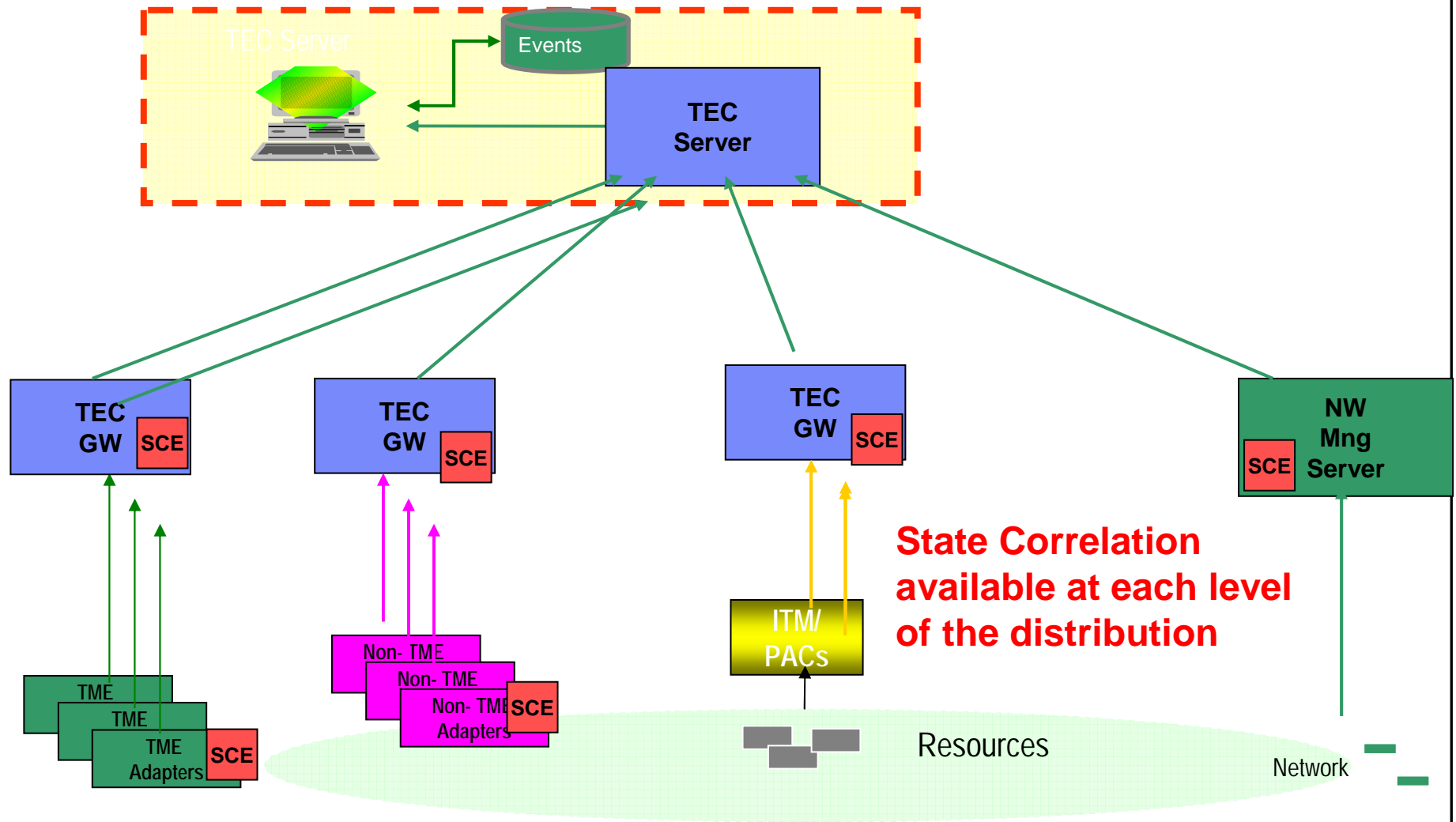
Agenda

- Introduction to the session :02
- SCE Introduction/Details/Logic :05
- Form teams & Group Work :23
- Group Presentations :45

Session Goals

- In this session:
 - ▶ Form workgroups 5-10 people
 - ▶ Receive packets of info for the team
 - ▶ Build your solution in 20 minutes
 - ▶ Present the solution to the group, receive feedback, Q&A

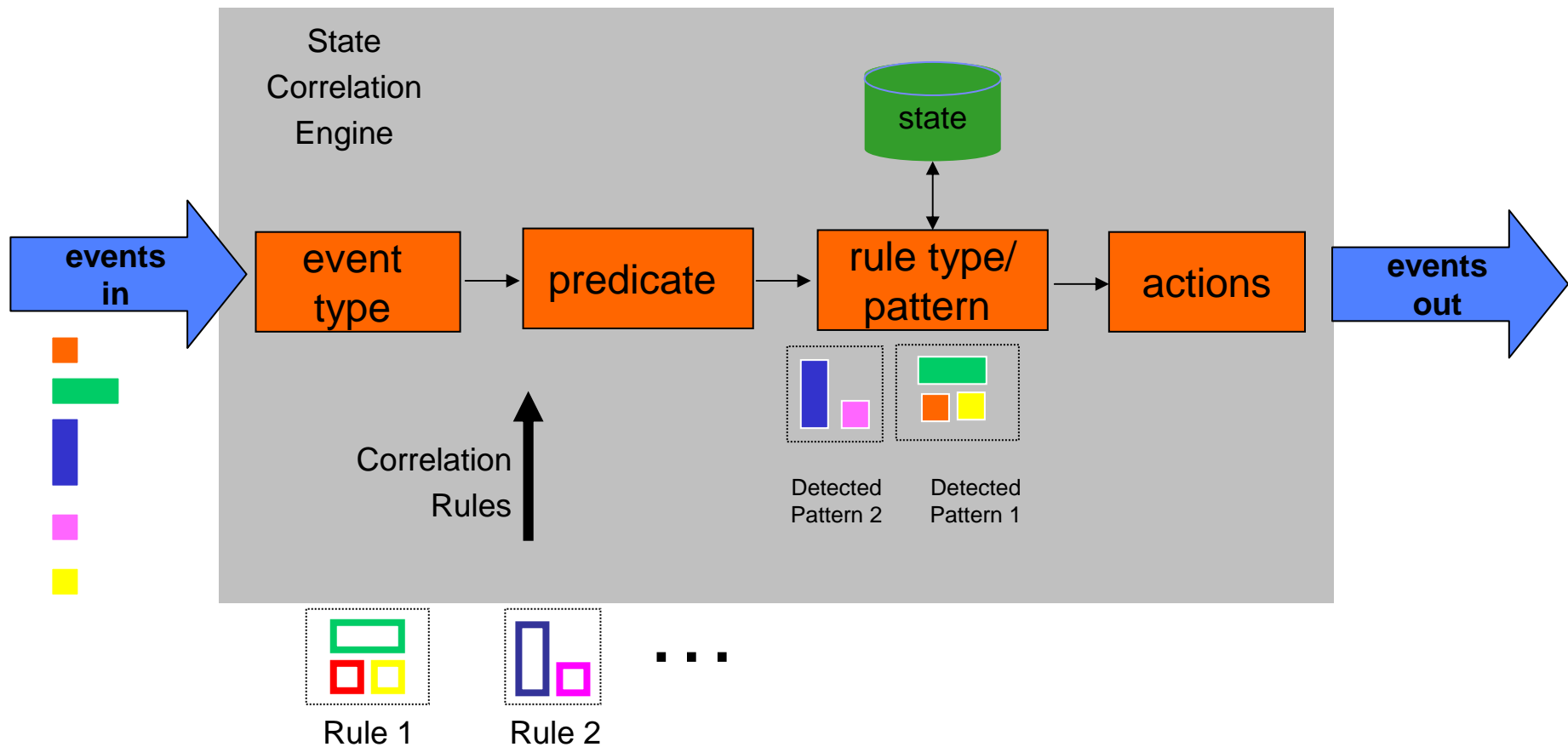
Distributed Event Correlation



State Correlation Capabilities

- Filters events at several hundred events/second
 - Duplicate suppression
 - Event Thresholding
 - Toggling (link-up/link-down)
 - Summary events
- TEC3.9 Enhancements:
 - TEC GW SCE supports TME and non TME (sockets) communication.
 - SCE supported at the adapter level as part of the EIF library
 - Customizable Action support
 - Sequence and Incomplete Sequence rule types.

State Correlation Concepts



Rule Types

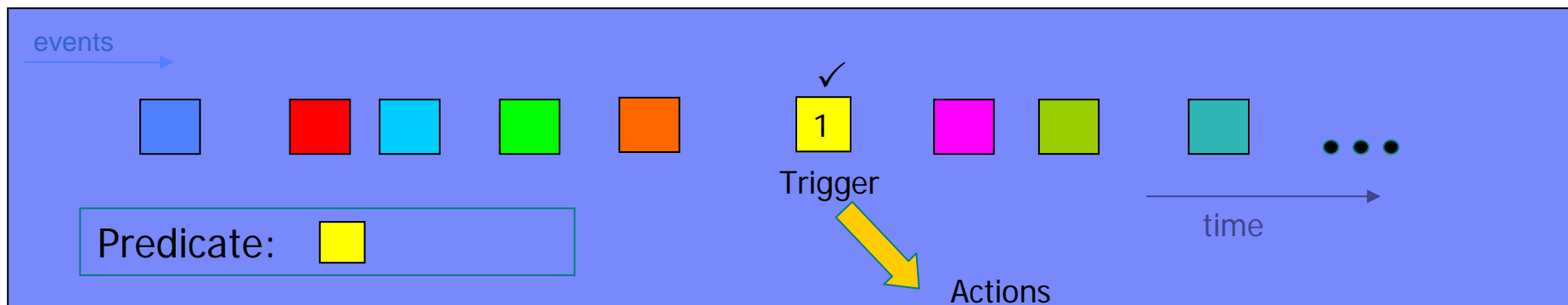
- Match
- Collector
- Duplicate
- Threshold
- Passthrough – Ordered/ unordered sequence
- ResetOnMatch – incomplete sequence

Base patterns can be combined/nested to create more complex patterns.

SCE rules are provided via an XML file conforming to
\$BINDIR/TME/TEC/default_sm/tecsce.dtd.

Match

- Each event is matched against a predicate expression.
- If the expression is true, event matches and actions are executed.
- Stateless correlation – no state or history of previous events is used to detect pattern.
- Useful for filtering events out of the stream, automation.



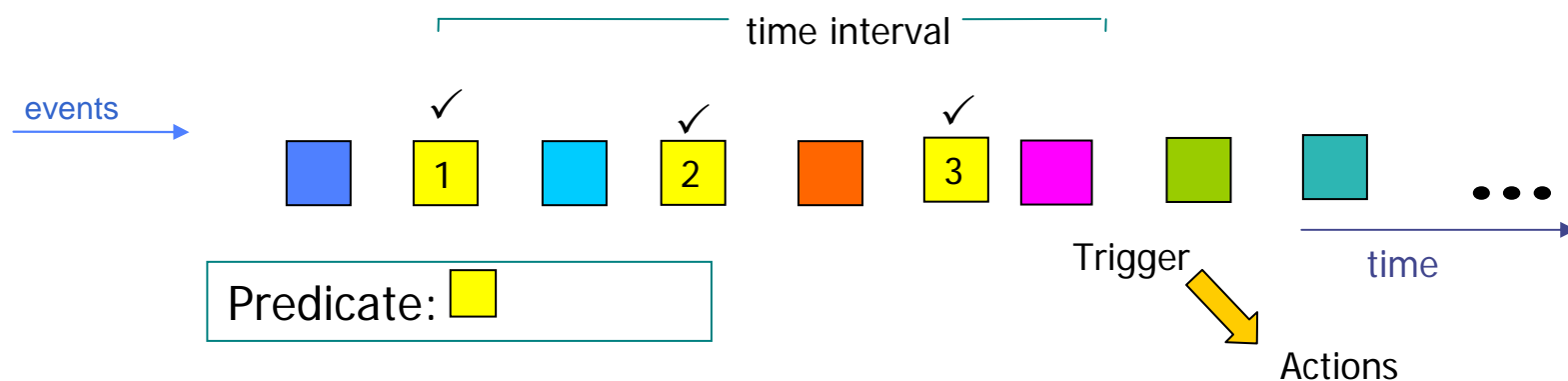
Match Example

If a *serverStatus* event reports *serverLoad > 0.95*,
send an *Alert*

```
<rule id="simple_filter.match" >
  <eventType>serverStatus</eventType>
  <match>
    <predicate>
      <![CDATA[&serverLoad > 0.95]]>
    </predicate>
  </match>
  <action function="Alert" >
    <parameters>
      <![CDATA[
        destinationHost = test.raleigh.ibm.com]]>
    </parameters>
  </action>
</rule>
```

Collector

- Stateful correlation across multiple events
 - ▶ Events are collected during a time window.
 - ▶ At the end of the period, collected events are available for actions.
- A common use is to count number of events and summarize them into just one event.



Collector Example

Collect *RS_PingFlood* events that have the same *source* and *destination IPAddresses* for *30* seconds and forward a *summary* event

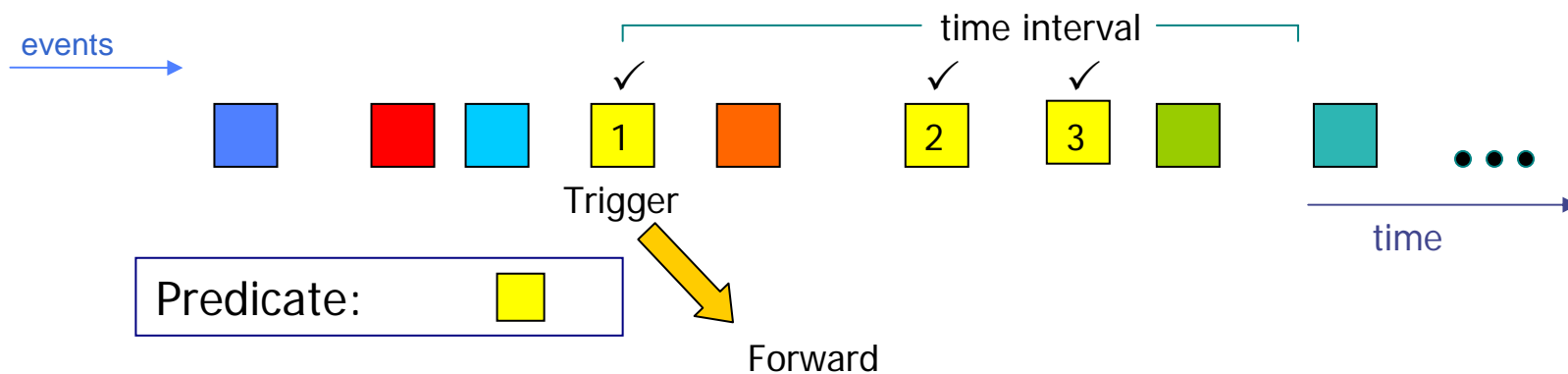
```
<rule id="RM.RS_Summary.RS_PingFlood_Evt">
  <eventType>RS_PingFlood</eventType>
  <collector timeInterval="30000" >
    <cloneable
      attributeSet="rm_SourceIPAddr rm_DestinationIPAddr"
      ignoreMissingAttributes="true"
    />
    <predicate>true</predicate>
  </collector>
  <action function="TECSummary">
    <parameters>
      <![CDATA[SET:msg=SUMMARY_PingFlood]]>
    </parameters>
  </action>
</rule>
```

Cloneable

- You must use cloning (use the clonable tag) if you want events to match the same conditions but be counted differently on some other attribute(s).
- As an example, if you make the hostname clonable, then 10 event exact same events from 10 different hostnames would only be one occurrence per hostname.
- Using more than one attribute means that all attributes specified must be present for the event to match the conditions UNLESS you use the undocumented `ignoreMissingAttributes=true`.

Duplicate

- Blocks the forwarding of duplicate events within a time interval
- The first event is immediately forwarded while its duplicates are discarded.



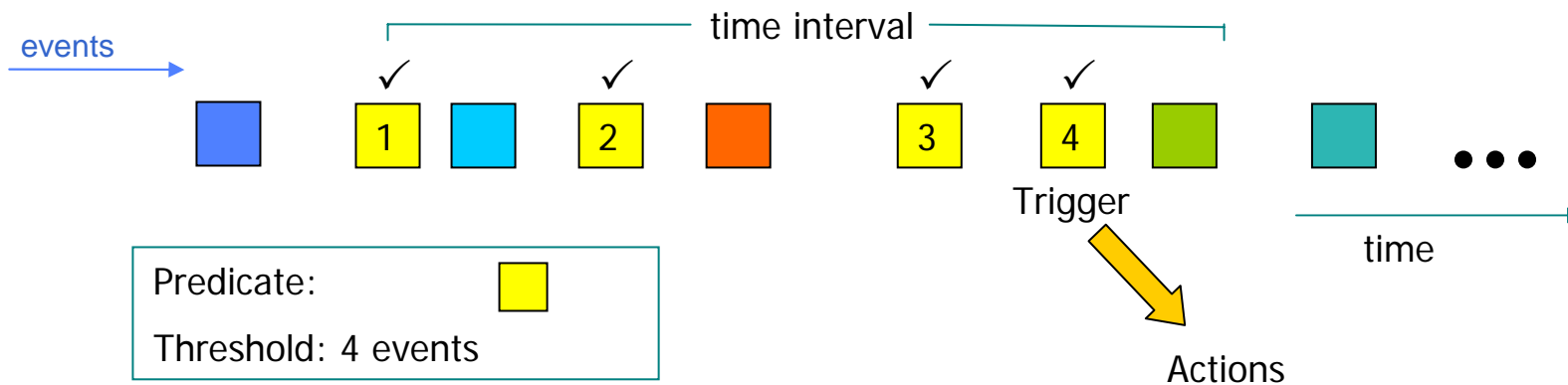
Duplicate Example

Forward the first *serverUnreachable* event and ignore all others from the same *serverName* for *10* seconds.

```
<rule id="block_duplicates" >
  <eventType>serverUnreachable</eventType>
  <duplicate timeInterval="10000" >
    <cloneable ignoreMissingAttributes="false" attributeSet="serverName" />
    <predicate>
      <![CDATA[true]]>
    </predicate>
  </duplicate>
</rule>
```

Threshold

- Evaluate a threshold based on events received within a time window.
- The time window may be fixed or sliding.
- Threshold may be based on event count or on a aggregation across the collected events.



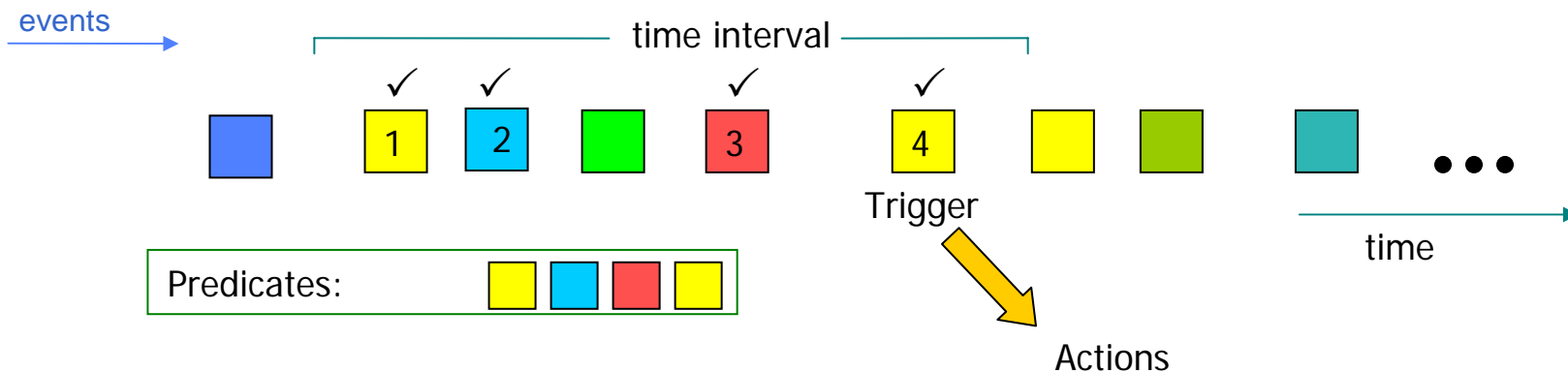
Threshold Example

If we receive more than *100 serverUnreachable* events for the same *serverName* in *1* second then send an *Alert*

```
<rule id="threshold" >
  <eventType>serverUnreachable</eventType>
  <threshold thresholdCount="100" timeInterval="1000"
    timeIntervalMode="fixedWindow" triggerMode="firstEvent" >
    <cloneable ignoreMissingAttributes="false" attributeSet="serverName" />
    <predicate><![CDATA[true]]></predicate>
  </threshold>
  <action function="Alert" >
    <parameters>
      <![CDATA[destinationHost = test.raleigh.ibm.com]]>
    </parameters>
  </action>
</rule>
```

Sequence

- Look for a ordered or unordered sequence of events within a period of time.
- If the sequence occurs within the time limit, events are available to Actions
- Otherwise, events collected by the rule are discarded.



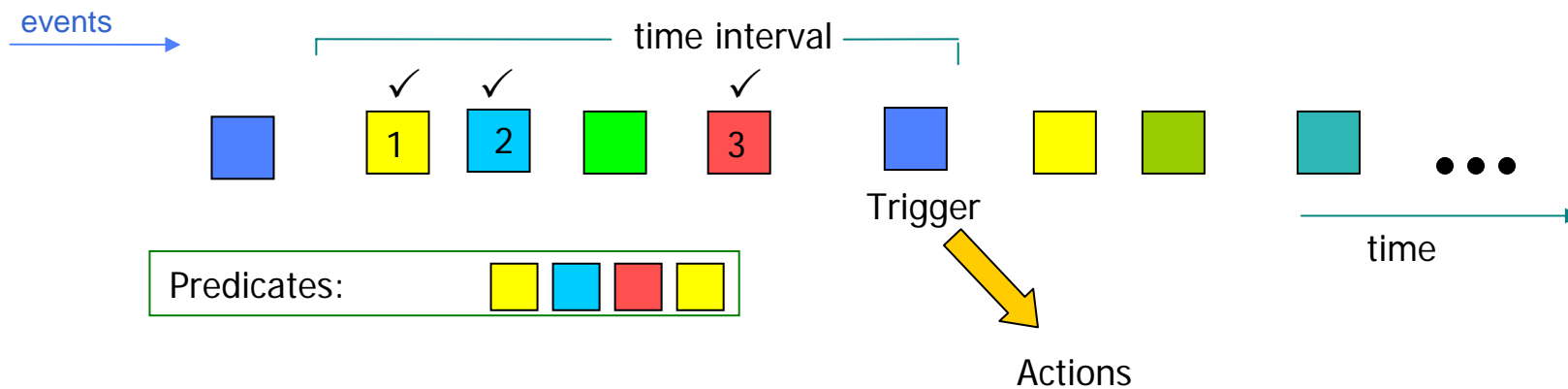
Sequence - Example

If we receive the *serverStatus* events: “*database component is down*” and “*webserver component is down*” in *any order* in a *10* minute interval then send an *Alert*

```
<rule id="detect_sequence.passThrough" >
  <eventType>serverStatus</eventType>
  <passthrough timeInterval="600000" randomOrder="true" >
  <cloneable attributeSet="serverName" />
  <predicate>
    <![CDATA[ &serverType == "webserver" && !&serverOperational ]]>
  </predicate>
  <predicate>
    <![CDATA[ &serverType == "database" && !&serverOperational ]]>
  </predicate>
  </passthrough>
  <action function="Alert" >
  <parameters>
    <![CDATA[destinationHost = test.raleigh.ibm.com]]>
  </parameters>
  </action>
</rule>
```

Incomplete Sequence

- Sequence is started but not completed during the time window.
- If the event sequence does not occur within the time limit, the rule triggers and events that arrived are available to Actions
- If the event sequence is completed then the events are discarded. The rule is “reset”.

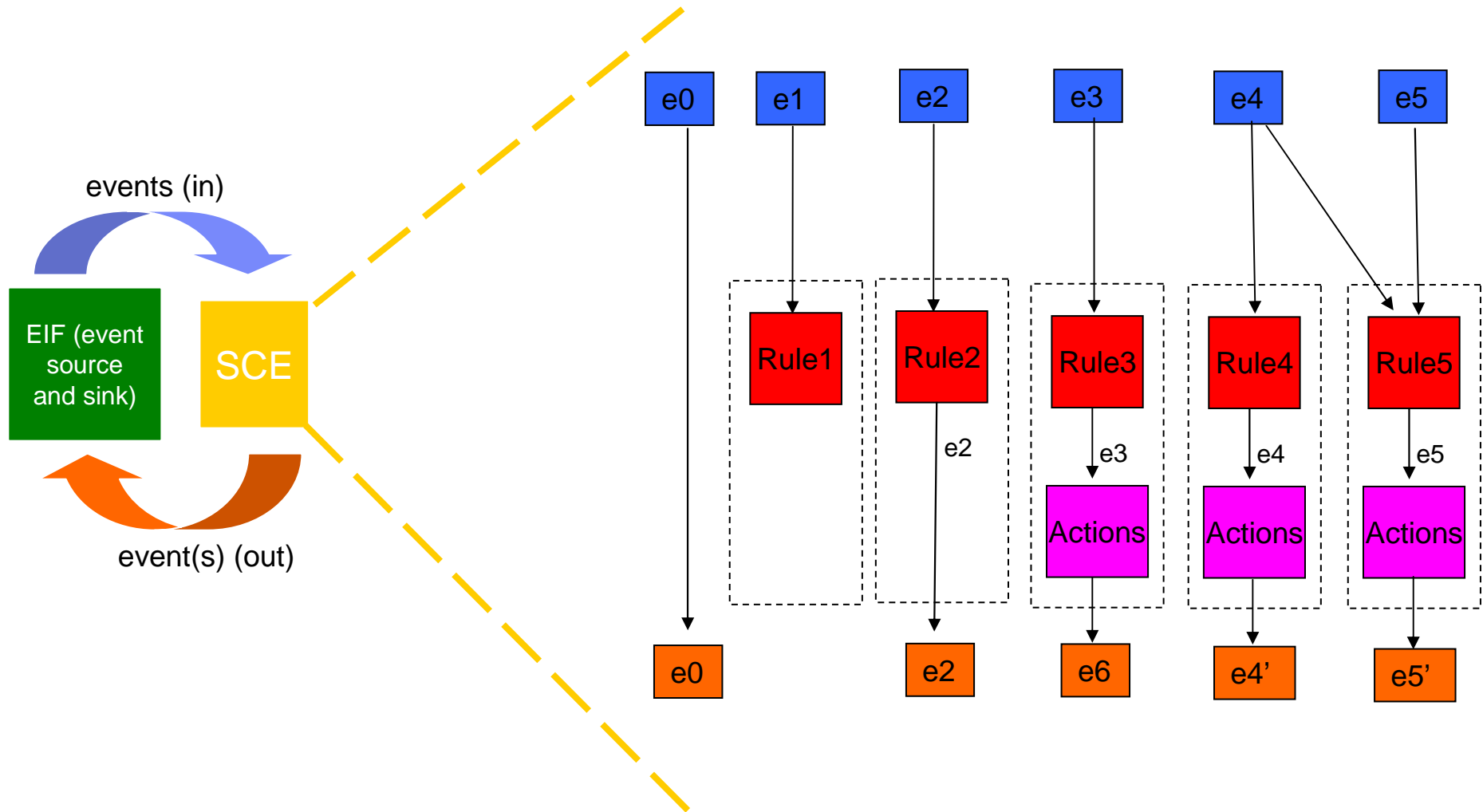


Incomplete Sequence - Example

Toggleing Suppression: Suppress *Fan down* and *Fan up* events that happen within 20s:

```
<rule id="suppressToggleing.fanState">
  <eventType>UPS_Fan_Status</eventType>
  <resetOnMatch timeInterval="20000" >
    <cloneable attributeSet="hostname"/>
    <predicate>
      <![CDATA[
        &status == "down"
      ]]>
    </predicate>
    <predicate>
      <![CDATA[
        &status == "up"
      ]]>
    </predicate>
  </resetOnMatch>
  <triggerActions>
  </triggerActions>
</rule>
```

SCE Event Flow

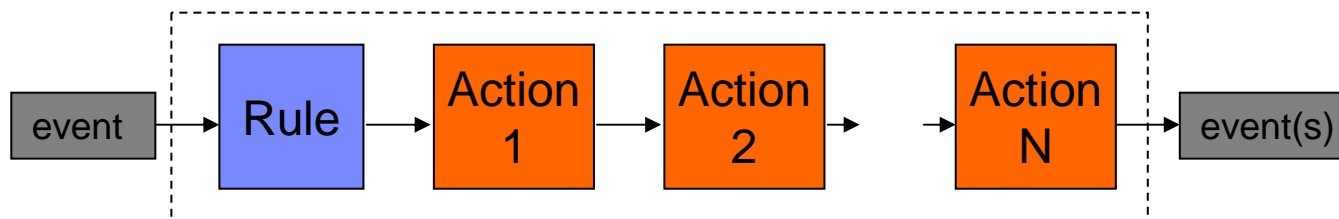


Event Flow Details

1. EIF receives a TEC event, creates a SCE event and calls the SCE
2. A copy of the event is passed to every rule (that subscribe to events of its type) in sequence (XML order)
3. Each rule processes the event possibly adjusting the state of the state machine on which the rule is based.
4. The event may be retained by stateful rules
5. If a rule triggers, the event or set of events held by the rule are passed to the first action defined for the rule. The event that triggered the rule is passed to the next rule only after the actions are finished.
6. The action processes the event(s), modifying, creating new events or dropping existing events and forwards it to the next action.
7. The final action forwards its modified event(s) back to SCE
8. SCE passes the events back to EIF using a callback mechanism.
9. EIF sends the event.

Actions

- Actions are Java classes which methods are invoked when a rule triggers
- TEC predefined actions:
 - ▶ Discard: Drops the event
 - ▶ Forward: Sends event to another rule(s)
 - ▶ TECSummary: Sends an event summary (derived from the last event sent to the action) updating the repeatcount and optionally setting the msg.
 - ▶ SendTECEvent: Helper action for custom Actions that modify events.
 - ▶ DebugRule: prints triggered rule info + PREFIX: parameter to FILE: parameter.
- An action receives an event or a list of events
- A rule can have a sequence of actions. The result of each action is passed to the next action in the sequence
- If no actions are specified, events are send back to EIF for forwarding.



Rule Logic - 1

- Using a “fixed” window of time is quite different than a “sliding” window of time.
- The SCE analyzes the events according to the time they are received at the SCE, NOT by the time in the event slot attribute.
- There is no “exit” statement in the rule language, therefore all rules will be processed (unless you forward).
- If you use the Forward action, you must be careful not to create infinite loops by passing the event from one rule to another and then possibly back up the chain to some rule that it has already seen as this will cause recursion. (`directAccess="false"`)
- The only way to drop events at the SCE is to have the event count not meet the thresholding limits within the specified time interval OR by specifically matching with a match rule and then doing a Discard action.

Rule Logic - 2

- Pay attention to what other rules do... it doesn't make sense to use a match rule with discard as the action if you have a collector rule that impacts the same event!
- Think about the impact that ignoreMissingAttributes=true has on your rule when doing cloning.
- The only slot that can be changed with the out of the box product is the msg slot. If you change it, the previous content is lost.
- (ALREADY SAID BUT IMPORTANT!) The SCE knows NOTHING about TEC itself--no BAROC, no event classes, no event hierarchies, nothing about TEC. Don't forget this.

SCE References

- “Tivoli Field Guide - TEC 3.9 State Correlation Engine: How to Prevent TEC from Becoming Flooded” by Arend Berg and Marc Purnell.
- TEC3.9 Rule Builders Guide, Part 2
- TEC3.9 User’s Guide, Chapter 5, Configuring the Gateway for State Correlation
- TEC3.9 Event Integration Facility Reference, Chapter 5, Filtering with State Correlation
- "IBM Tivoli Enterprise Console TME New Installations v3.9" CD:
 - ▶ Samples: EIFSDK/samples/state_correlation
 - ▶ Javadocs: EIFSDK/javadoc/state_correlation



| IBM Software Group

Group Work

Tivoli software



Problem Statement

- Choose a rule from the existing default rulebase for ITM 6.x and come up with a way to implement it at the SCE.
 - ▶ OR ALTERNATIVELY: Provide your own TEC rule to re-write for the SCE.
- Your group should present:
 - ▶ Explanation of rule logic in TEC rule
 - ▶ Flow chart for SCE conversion
 - ▶ Explanation of flow
 - ▶ What does the solution offer? What is missing?



| IBM Software Group

Group Presentations

Tivoli software



For More Information



Tivoli User Groups

You can get even more out of Tivoli software by participating in independently run Tivoli User Groups around the world; learn about online and in-person opportunities near you at www.tivoli-ug.org



Tivoli Training

IBM offers technical training and education services to help you acquire, maintain and optimize your IT skills. For a complete Tivoli Course Catalog and Certification Exams visit www.ibm.com/software/tivoli/education



Tivoli Services

With IBM Software Services for Tivoli, you get the most knowledgeable experts on Tivoli technology to accelerate your implementation. For a complete list of Services Offerings visit www.ibm.com/software/tivoli/services



Tivoli Support

IBM Software Premium Support provides an extra layer of proactive support, skills sharing and problem management, personalized to your environment. Visit www.ibm.com/software/support/premium/ps_enterprise.html