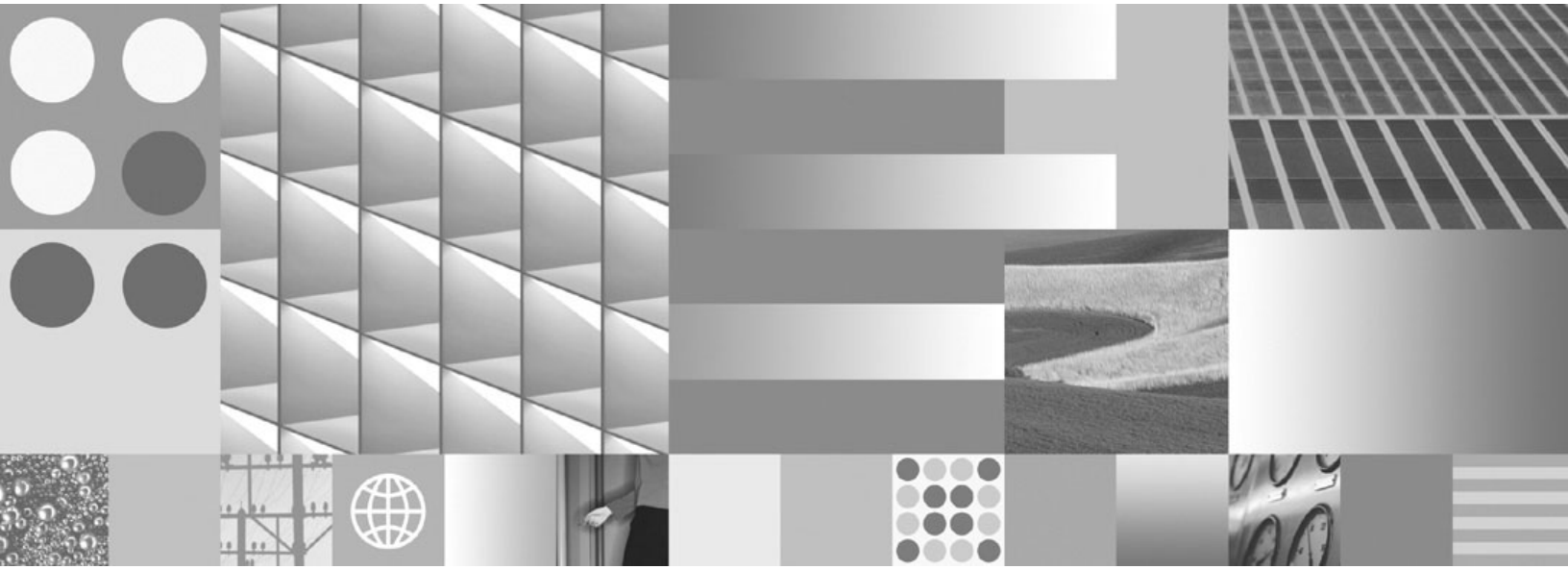
ACT(Active Correlation Technology)

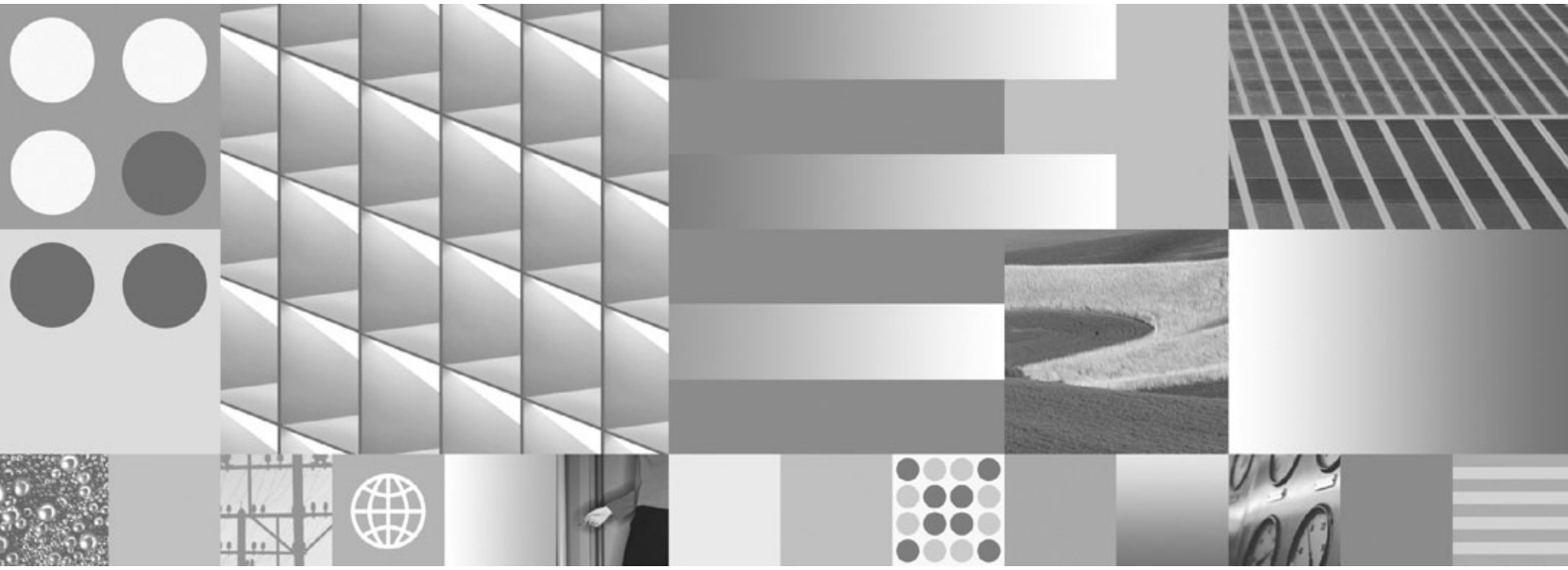
버전 **6.0**



규칙 작성자의 안내서 및 참조서

ACT(Active Correlation Technology)

버전 **6.0**



규칙 작성자의 안내서 및 참조서

참고

이 정보 및 이 정보가 지원하는 제품을 사용하기 전에 135 페이지의 『주의사항』의 정보를 읽으십시오.

제 1 판(2006년 6월)

이 책에 대한 정보

이 책에서는 IBM® ACT(Active Correlation Technology)의 개요 및 복잡한 이벤트를 처리하는 ACT의 역할에 대해 설명합니다. ACT(Active Correlation Technology) 규칙 언어는 이벤트를 상관시킬 규칙을 작성하는 XML 기반 언어입니다. 이 정보는 비즈니스 조직의 이벤트를 상관시킬 규칙을 작성하는 방법을 이해해야 하는 규칙 작성자를 위해 설계되었습니다.

목차

제 1 부 규칙 작성자의 안내서

제 1 장 소개

복잡한 이벤트 처리(CEP, Complex Event Processing)에 대해 간략하게 설명하고 ACT(Active Correlation Technology)의 개요 및 복잡한 이벤트 처리에서 ACT의 역할을 설명합니다.

오늘날의 비즈니스 환경

오늘날 상업 및 정부 조직은 컴퓨터 네트워크, 특히 인터넷을 통한 전자 정보 처리에 의존하고 있습니다. 그리드 컴퓨팅과 같은 추가 기술을 사용하여 조직은 중요한 어플리케이션을 언제든지 세계 어디에서나 실행시키고 있습니다. 비즈니스 프로세스, 활동 및 인프라 그리고 글로벌 사회는 조직의 정보 기술(IT)에 의존하고 있습니다.

조직은 항상 비즈니스에서 어떤 일이 벌어지고 있는지 알아야 합니다. 예를 들어, 조직은 중요한 어플리케이션이 사용 가능한지 여부와 제대로 작동하는지 여부 및 비즈니스 프로세스, 활동 또는 인프라에서 잠재적인 위기를 발견하고 방지하는 방법을 알아야 합니다. 위기 상황이 발생하면 조직은 즉시 문제점을 이해하고 문제의 원인과 이를 해결하는 방법을 찾아내야 합니다.

비즈니스 프로세스, 활동 및 인프라와 관련된 대부분의 이벤트에는 정보량이 많거나 정보가 개별적이고 서로 관련되지 않은 조각으로 존재하므로 요약하기가 어렵기 때문에 중요하게 인식되거나 이해되지 않았습니다. 그러나 이벤트가 축적되고 연관되면 이러한 이벤트 간의 관계를 쉽게 이해할 수 있으므로 양질의 정보를 얻을 수 있습니다.

복잡한 이벤트 처리의 목적은 실시간으로 이벤트에 대한 더 나은 정보를 얻기 위한 것입니다.

복잡한 이벤트 처리

*이벤트*는 발생한 일에 대한 단순한 공고입니다.

*복잡한 이벤트 처리*는 분석에서 파생된 상위 레벨 이벤트, 연관 및 이벤트 구동 시스템에서 하위 레벨 이벤트의 요약입니다. 복잡한 이벤트라고 불리는 상위 레벨 이벤트는 사람들에게 비즈니스 기회나 문제점에 대해 알기 쉬운 용어로 알려주거나 자동 프로세스를 트리거하는 데 적합합니다. 그러면 조직은 잠재적인 기회나 문제점을 초기에 경고하고, 비즈니스 프로세스, 활동 또는 인프라의 상태를 변경하는 근본 원인을 보다 잘 이해하여 보다 효율적으로 운영할 수 있습니다.

*이벤트 연관*은 이벤트 스트림에서 실시간으로 패턴을 정의 및 발견하고 관련된 이벤트에 대한 응답으로 조치를 구현하는 프로세스입니다. 이는 발견한 증상을 기반으로 문제

점을 식별하는 데 사용됩니다. 이벤트는 원인별, 시간별, 멤버쉽별 또는 이러한 요소의 조합별로 연관시킬 수 있습니다. 이벤트 연관은 복잡한 이벤트 처리의 필수 부분입니다.

ACT(Active Correlation Technology)

ACT(Active Correlation Technology)는 규칙을 사용하여 이벤트 스트림에서 실시간으로 패턴을 발견합니다. 이 기술은 대부분의 경우 응답 조치는 하나의 하위 레벨 이벤트에 의해 트리거되어서는 안되며 대신 다른 컨텍스트 내에서 다른 시간에 발생하는 이벤트의 복잡한 작성에 의해 트리거되어야 한다는 이해를 기반으로 합니다. ACT(Active Correlation Technology)는 이벤트 간 관계를 사용하여 비즈니스 기회 및 문제점에 대한 인식을 제공합니다. 예를 들어, 이벤트의 연관을 통해 실시간으로 확보한 비즈니스에 대한 새로운 인식을 기반으로 조직은 다음과 같은 유형의 조치를 수행할 수 있습니다.

- 홀리데이 세일 기간 중에 일부 또는 모든 고객에게 배송비 할인을 제공합니다.
- 이후 30일 동안 배송비를 배송 회사, 주문 금액 및 주문 수량에 따라 계산합니다.
- 2005년 7월 1일부터 2005년 12월 31일일까지 500USD 이상 구매 고객에게 25USD의 상품권을 보냅니다.
- 주문 처리가 36시간 내에 완료되지 않은 경우 관리자에게 알립니다.
- 30초 내에 동일한 컴퓨터에 다섯 번 이상의 로그인 시도가 있는 경우 관리자에게 알립니다.

ACT(Active Correlation Technology)는 다음과 같은 기본 항목으로 구성되어 있습니다.

ACT(Active Correlation Technology) 규칙 언어
이벤트를 연관시키기 위해 규칙을 작성하는 XML 기반 언어입니다. 그런 다음 이러한 규칙을 ACT(Active Correlation Technology) 런타임 환경으로 전개할 수 있습니다.

ACT(Active Correlation Technology) 엔진
ACT(Active Correlation Technology) 컴파일러의 출력에 따라 이벤트를 처리하는 ACT(Active Correlation Technology) 구성요소입니다.

ACT(Active Correlation Technology) 규칙 빌더
ACT(Active Correlation Technology) 규칙 언어로 된 연관 규칙을 작성하기 위한 GUI입니다.

ACT(Active Correlation Technology) 런타임 환경은 ACT(Active Correlation Technology) 엔진이 임베드된 어플리케이션입니다.

제 2 장 규칙 언어 개요

이 개요에서는 ACT(Active Correlation Technology) 규칙 언어의 핵심 개념을 설명합니다.

규칙 패턴은 이벤트 연관 상황의 표시입니다(예: 임계값 조건 또는 중복 이벤트 감지). ACT(Active Correlation Technology) 규칙 언어에는 IBM 고객이 해결해야 하는 대부분의 이벤트 연관 상황을 표시할 일곱 개의 입증된 규칙 패턴이 포함되어 있습니다. 일곱 개의 규칙 패턴 중 여섯 개는 상태 규칙을 정의하고, 패턴 중 하나는 상태가 없는 규칙을 정의합니다.

상태 규칙은 특정 기간 동안 발생하는 여러 이벤트를 연관시키고 이러한 이벤트에 대한 응답을 생성합니다. 상태가 없는 규칙은 특정 기준에 맞는 단일 이벤트만을 처리하고 해당 이벤트에 대한 응답을 생성합니다.

상태 규칙(stateful rule)

특정 기간 동안 이벤트 콜렉션에 대해서 작동하려는 목적으로 규칙 인스턴스의 특성에 대한 정보와 같은 상태 정보를 보관하는 규칙입니다. 콜렉션, 계산, 중복, 연속, 임계값 또는 타이머와 같은 규칙 패턴에 의해 정의된 규칙은 상태 규칙입니다.

상태가 없는 규칙(stateless rule)

상태 정보를 보관하지 않으므로 한 번에 한 이벤트에 대해서만 작동하는 규칙입니다. 필터 패턴에 의해 정의된 규칙은 상태가 없는 규칙입니다.

관련 참조

51 페이지의 제 4 장『규칙 세트 조직 요약』

이 참조에서는 규칙 세트, 규칙 블록 및 각 규칙 유형의 언어 요소를 모두 나열합니다. 이 참조는 규칙 세트를 코딩하기 위한 빠른 참조용으로도 사용할 수 있습니다.

65 페이지의 제 5 장『언어 요소 참조』

이 참조는 ACT(Active Correlation Technology) 규칙 언어의 XML 스키마로 된 언어 요소의 세부사항을 설명합니다. 언어 요소는 알파벳 순서로 나열되어 있으며 각 요소에서 사용 가능한 속성은 해당 요소의 주제 내에 설명되어 있습니다.

129 페이지의 제 6 장『용어집』

이 용어집에는 ACT(Active Correlation Technology)의 중요 개념에 대한 용어 및 정의가 있습니다.

규칙의 구조

규칙의 가장 기본 파트는 이벤트 선택, 그룹화 키, 상태 규칙의 시간 창, 규칙 응답, 활성화 간격 및 수명 주기 조치입니다. 규칙에는 또한 표현식 및 변수가 포함됩니다. 표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다.

이벤트 선택

이벤트 선택 기준은 규칙이 어떤 이벤트를 처리하기 위해 승인할지를 판별합니다. <eventSelector> 요소는 규칙의 이벤트 선택 기준을 정의합니다. 이벤트 선택은 타이머 패턴이 정의한 규칙을 제외한 모든 규칙에 적용됩니다. 타이머 규칙은 이벤트를 처리하지 않으므로 이벤트 선택기준이 포함되어 있지 않습니다.

그룹화 키

일반적으로 각 활성 규칙에는 ACT(Active Correlation Technology) 엔진에서 실행 중인 하나의 규칙 인스턴스 또는 사본이 있습니다. 그러나 때로는 종종 다른 자원 그룹과 관련되는 다른 이벤트 그룹에 동일한 규칙이 필요하기도 합니다. 그룹화 키는 규칙에게 공통 특성을 공유하는 각 이벤트 그룹마다 별도의 규칙 인스턴스(또는 사본)를 작성하도록 지시하는 방법입니다.

그룹화 키는 추가적인 이벤트 선택 양식의 역할을 합니다. 규칙이 그룹화 키를 사용하여 정의된 경우, 규칙이 그룹화 키로 정의된 특성이 있는 이벤트를 받으면, 이벤트는 해당 특성을 공유하는 이벤트를 처리하는 규칙 인스턴스로 전송됩니다. 예를 들어 `Audit Failure` 유형의 모든 보안 이벤트를 수집하고 이벤트의 호스트 이름 속성이 될 그룹화 키를 정의하는 규칙을 정의할 수 있습니다. 이제 규칙은 호스트 이름 속성의 각 고유값에 대해 실행 중인 규칙의 개별 사본과 함께 여러 번 사용할 수 있습니다. 또한 `Audit Failure` 이벤트를 받는 모든 시스템을 모니터하여 이러한 이벤트가 각 호스트 이름당 2분의 기간 동안 11차례 이상 발생하는지 여부를 판별할 수도 있습니다.

<groupingKey> 요소는 규칙의 그룹화 키를 정의하며, 콜렉션, 계산, 중복, 연속 및 임계값 패턴으로 정의되는 규칙에만 사용할 수 있습니다.

상태 규칙의 시간 창

정직 규칙은 특정 기간 동안 발생하는 여러 이벤트를 상호 관련시키므로 상태 규칙의 기본 파트는 <timeWindow> 요소로 정의되는 시간 창입니다. 시간 창은 상태 규칙이 패턴을 일치시키기 위해 처리 중인 기간을 지정합니다.

규칙 응답

규칙 응답 조치는 규칙이 처리를 완료할 때 수행할 조치를 정의합니다. 다음 각 언어 요소는 규칙 응답 조치의 여러 다른 유형을 정의합니다.

- <onDetection> 내의 <action>

- <onNextEvent> 내의 <action>
- <onTimeOut> 내의 <action>
- <onTimeWindowComplete> 내의 <action>

규칙에 사용 가능한 규칙 응답 조치의 유형은 규칙 패턴에 따라 다릅니다.

활성화 간격

활성화 간격은 규칙이 활성 및 비활성인 시기를 정의합니다. <activationInterval> 요소는 규칙의 활성화 간격을 정의합니다.

규칙은 개별 지정 시간에 또는 특정 이벤트에 의해 활성화하거나 비활성화할 수 있습니다.

규칙을 개별 지정 시간에 및 특정 이벤트에 의해 활성화되거나 비활성화되도록 지정한 경우 규칙은 지정 시간 또는 이벤트 수신 중 먼저 발생하는 조건에 의해 활성화되거나 비활성화됩니다. 그러나 이 경우 규칙은 전체 수명 주기 동안 여러 이벤트에 의해 활성화되거나 비활성화될 수 있습니다. 예를 들어 규칙은 이벤트에 의해 활성화되고, 비활성화되고, 정의된 지정 시간에 활성화되고, 다시 비활성화된 다음 다른 이벤트에 의해 활성화될 수도 있습니다.

<activationByGroupingKey> 요소는 <activationInterval> 요소 내에 포함된 한 요소입니다. <activationByGroupingKey> 요소에는 <groupingKey> 요소로 정의되는 규칙 인스턴스를 활성화하고 비활성화할 수 있는 이벤트를 지정하는 요소가 포함되어 있습니다.

수명 주기 조치

수명 주기 조치는 규칙의 수명 주기의 네 가지 기본 단계인 로드, 활성화, 비활성화 및 로드 해제 시에 취할 조치를 정의합니다.

<lifeCycleActions> 요소에는 이러한 조치를 정의하는 다음과 같은 요소가 포함되어 있습니다.

- <onLoad> 내의 <action>
- <onActivation> 내의 <action>
- <onDeactivation> 내의 <action>
- <onUnload> 내의 <action>

규칙의 수명 주기

규칙의 수명 주기에서 각 단계에는 여러 개의 원인과 영향이 있을 수 있습니다. <lifeCycleActions> 요소에 정의된 대로 수명 주기 조치 내에 표현식을 작성하고 포함함으로써 규칙 작성자는 각 단계에서 수행할 조치를 정의할 수 있습니다.

규칙의 수명 주기에서의 단계

다음은 규칙의 수명 주기에서 네 개의 기본 단계입니다.

로드 규칙을 실행 중인 ACT(Active Correlation Technology) 엔진으로 로드합니다. 이는 <onLoad> 요소 내에서 조치를 트리거합니다.

활성화 <onActivation> 요소 내에서 조치를 트리거하는 규칙의 활성화입니다.

비활성화

<onDeactivation> 요소 내에서 조치를 트리거하는 규칙의 비활성화입니다.

로드 해제

규칙을 실행 중인 ACT(Active Correlation Technology) 엔진에서 로드 해제합니다. 이는 <onUnload> 요소 내에서 조치를 트리거합니다.

활성화 및 비활성화 단계는 규칙의 수명 주기에서 여러 번 발생할 수 있지만 로드와 로드 해제 단계는 한 번만 발생합니다.

일반적으로는 수명 주기 조치를 정의할 필요가 없습니다. 다음 예와 같은 경우에 사용자는 특정 수명 주기 조치를 정의하고 싶게 됩니다.

- 특정 규칙이 로드되면 해당 규칙 내에서 액세스해야 하는 외부 시스템(예: 데이터베이스 관리자)에 대한 연결을 작성하고 싶어할 수 있습니다. 이 동일한 규칙이 로드 해제될 때에는 연결을 끊고 필요한 모든 정리 프로세스를 실행하고 싶어합니다.
- 특정 규칙이 활성화되면 해당 규칙에서 특정 자원을 사용 가능한지 확인하고 싶어할 수 있습니다.
- 임계값 규칙이 비활성화되었지만 임계값에 도달하지 않았고 기간이 아직 종료되지 않은 경우 메시지를 이 정보와 함께 누군가에게 전달하고 싶어할 수 있습니다.

규칙의 활성화와 비활성화는 수명 주기에서 여러 번 발생할 수 있으므로 이러한 단계에 코딩한 조치를 자주 실행할 수 있습니다.

각 수명 주기 단계의 원인과 영향

표 1에서는 각 수명 주기 단계의 원인과 영향을 나열합니다.

표 1. 각 수명 주기 단계의 원인과 영향

수명 주기 단계	원인	영향
로드	다음과 같은 상황입니다. • 규칙이나 규칙 블록이 추가되거나 대체되고 이로 인해 새 규칙이 로드됩니다. • 규칙 세트가 ACT(Active Correlation Technology) 엔진에서 교체되고 이로 인해 새 규칙 세트의 규칙이 로드됩니다.	<onLoad> 요소 내의 조치가 실행됩니다.

표 1. 각 수명 주기 단계의 원인과 영향 (계속)

수명 주기 단계	원인	영향
활성화	규칙이 활성화됩니다. 규칙은 다음 방법을 사용하여 활성화할 수 있습니다. • <activationInterval> 요소 내의 정의에 따라 • act_lib 변수를 통해 사용 가능한 activate() 메소드를 통해서 • ACT(Active Correlation Technology) 엔진에서 activate() 메소드에 대한 어플리케이션 호출을 통해서	규칙이 비활성이면 <onActivation> 요소 내의 조치가 실행됩니다.
비활성화	규칙이 비활성화됩니다. 규칙은 다음 방법을 사용하여 비활성화할 수 있습니다. • <activationInterval> 요소 내의 정의에 따라서. 예외로 <activationByGroupingKey> 요소 내의 <deactivateOnEvent> 요소는 규칙 비활성화를 초래하지 않습니다. • act_lib 변수를 통해 사용 가능한 deactivate() 메소드를 통해서 • ACT(Active Correlation Technology) 엔진에서 deactivate() 메소드에 대한 어플리케이션 호출을 통해서	규칙이 활성이면 <onDeactivation> 요소 내의 조치가 실행됩니다.
로드 해제	다음과 같은 상황입니다. • ACT(Active Correlation Technology) 엔진이 종료되고 이로 인해 규칙이 로드 해제됩니다. • 규칙이나 규칙 블록이 추가되거나 대체되고 이로 인해 이전 규칙이 로드 해제됩니다. • 규칙 세트가 ACT(Active Correlation Technology) 엔진에서 제거되거나 교체되고 이로 인해 이전 규칙 세트의 규칙이 로드 해제됩니다.	규칙이 활성이면 <onDeactivation> 요소 내의 조치가 실행되고 뒤이어 <onUnload> 요소 내이 조치가 실행됩니다. 그렇지 않은 경우 <onUnload> 요소 내의 조치만이 실행됩니다.

규칙의 조직

ACT(Active Correlation Technology) 규칙 언어는 규칙 세트의 일부인 규칙 블록으로 규칙을 구성합니다.

규칙 세트

규칙 세트에는 규칙 블록으로 구성되어 있고 ACT(Active Correlation Technology) 엔진에 의해 실행될 규칙이 포함되어 있습니다. 이는 규칙 실행 단위입니다. 각 ACT(Active Correlation Technology) 엔진은 한 번에 하나의 규칙 세트에 대해서만 작동합니다.

규칙 세트에 포함된 규칙은 ACT(Active Correlation Technology) 엔진으로 전송된 이벤트에 의해 트리거됩니다. 이벤트는 각 규칙의 이벤트 선택 기준에 따라 적절한 규칙으로 순차적으로 전달되며 한 번에 하나의 규칙이 실행됩니다. 동일한 이벤트를 여러 규칙에 적용하고 트리거할 수 있습니다. 이러한 규칙은 서로 반드시 관련될 필요는 없지만 관련되어 있을 수 있습니다.

규칙 세트 내에서 규칙 블록 및 규칙의 순서는 이벤트가 규칙 세트를 통해 흐르는 방법을 판별합니다.

변수 및 가져오기는 규칙 세트 범위 전반에서 표현식(사용자 정의 논리를 포함하는 코드)에 사용할 수 있도록 규칙 세트 레벨에 정의할 수 있습니다. 가져오기는 외부 코드에 액세스하기 위한 프로그래밍 언어 특정 방법입니다. 규칙 작성자는 규칙 내의 표현식에 사용할 외부 모듈(예: Java™ 클래스)을 가져오도록 가져오기를 정의할 수 있습니다.

규칙 블록

규칙 블록은 규칙 세트 내에서 규칙을 기능별로 도메인으로 그룹화하기 위한 조직 단위입니다. 도메인은 규칙의 그룹이 기능을 적용하는 데 기반으로 사용하는 범주입니다. 예를 들어, 도메인은 특정 지리적 지역, IT 관리 규율(예: 보안 감지 또는 네트워크 이벤트 연관) 또는 비즈니스 조직(예: 특정 회사 또는 회사의 부서)를 표시할 수 있습니다.

규칙 블록에는 규칙 및 다른 규칙 블록을 포함할 수 있습니다. 규칙 블록은 중첩할 수 있으므로 규칙의 계층 구조를 구성할 수 있습니다. 예를 들어, 규칙 세트에는 네트워크 이벤트 연관의 규칙 블록을 포함할 수 있고 네트워크 이벤트 연관의 규칙 블록에는 계층 2 연관용과 IP 연관용으로 각각 하나씩 두 개의 다른 규칙 블록이 포함될 수 있습니다.

그러므로 규칙 세트는 다양한 도메인의 이벤트 연간 기능을 제공하며, 규칙 블록은 유사한 이벤트 세트에 대한 액세스를 필요로 할 수 있는 이러한 서로 다른 도메인의 조직을 제공합니다.

변수 및 가져오기는 규칙 블록의 범위 전반에서 표현식에 사용할 수 있도록 규칙 블록 레벨에 정의할 수 있습니다. 규칙 블록의 범위에는 규칙 및 규칙 블록에 포함된 다른 규칙 블록이 포함됩니다.

규칙

규칙은 이벤트 간의 관계를 인식하고 적절한 규칙 응답을 실행하는 데 사용되는 연관 단위입니다. 규칙은 다음 일곱 개의 규칙 패턴 중 하나의 구현이며 기능에 따라 규칙 세트의 일부인 규칙 블록으로 구성되어 있습니다.

- 콜렉션 패턴
- 계산 패턴

- 중복 패턴
- 필터 패턴
- 연속 패턴
- 임계값 패턴
- 타이머 패턴

각 규칙은 패턴에 따라 고유한 이벤트 연관 기능을 제공할 수 있으며 규칙은 이벤트 전달을 통해 체인 연결할 수 있습니다. 이러한 규칙 체인을 통해 패턴이 서로 다른 이벤트 연관 기능을 결합하거나 중첩할 수 있습니다.

변수는 규칙 범위 전반에서 표현식에 사용할 수 있도록 규칙 레벨에 정의할 수 있습니다.

요약

요약하면 규칙 세트는 실행 단위이고, 규칙 블록은 조직 단위이고, 규칙은 연관 단위입니다. 규칙 세트에는 하나 이상의 규칙 블록이 포함되어 있고, 이들 각각은 추가로 규칙 블록을 포함할 수 있습니다. 각 규칙 블록에는 특정 도메인의 규칙이 포함되어 있습니다. 규칙 블록은 규칙의 계층 구조를 구성하기 위해 중첩될 수 있습니다. 규칙 세트 내에서 규칙 블록 및 규칙의 순서는 이벤트가 규칙 세트를 통해 흐르는 방법을 결정합니다.

변수 및 가져오기는 규칙 내에서 표현식에 사용할 수 있도록 규칙 세트 또는 규칙 블록 레벨에 정의할 수 있습니다. 변수 또는 가져오기의 범위는 각각 규칙 세트 또는 규칙 블록입니다. 변수는 규칙 레벨에서도 정의할 수 있지만 이렇게 하면 범위가 규칙만으로 제한됩니다.

규칙 패턴

규칙 패턴은 이벤트 연관 상황의 표시입니다(예: 임계값 조건 또는 중복 이벤트 감지). ACT(Active Correlation Technology) 규칙 언어는 콜렉션, 계산, 중복, 필터, 연속, 임계값 및 타이머와 같은 규칙 패턴을 정의합니다.

규칙의 패턴은 규칙에 의해 정의된 상황이 발생할 때 일치합니다. 패턴이 일치하면 규칙은 적절한 규칙 응답 조치를 수행하여 처리를 종결합니다. 규칙이 활성이면 규칙 패턴은 여러 번 일치할 수 있습니다.

필터 패턴에 의해 정의된 규칙은 규칙 언어에서는 상태가 없는 규칙에 불과합니다. 다른 모든 규칙은 상태 규칙입니다.

콜렉션 패턴

콜렉션 패턴으로 정의되는 콜렉션 규칙입니다. 이는 시간 간격 내에서 선택한 이벤트 그룹을 수집합니다. 이는 상태 규칙입니다.

개요

콜렉션 패턴은 특정 기간 동안 유사한 이벤트를 수집하는 데 사용됩니다. 시간 기간은 <timeWindow> 요소가 규칙 언어로 정의한 대로 필수 시간 창으로 표시됩니다.

규칙 응답 실행 조건

콜렉션 패턴을 사용하면 규칙 응답은 <onTimeWindowComplete> 요소에 의해 정의된 대로 시간 창이 완료될 때 실행됩니다.

이 규칙 패턴의 사용 예제

콜렉션 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

이는 특정 기간 동안 특정 이벤트 선택자의 기준에 부합하는 이벤트를 수집합니다. 특정 기간이 종료되면 이는 수집한 이벤트를 총 이벤트 계수와 요약된 이벤트에 대한 특성 정보를 포함하는 하나의 이벤트로 요약합니다.

관련 참조

53 페이지의『콜렉션 규칙 요약』
이 요약에는 콜렉션 규칙의 모든 언어 요소가 나열되어 있습니다.

계산 패턴

계산 규칙은 계산 패턴으로 정의됩니다. 이는 표현식을 통해 시간 간격 내에서 각 이벤트를 받을 때 수집된 이벤트에 적용됩니다. 이는 상태 규칙입니다.

개요

계산 패턴은 <computeFunction> 요소에서 규칙 언어로 정의된 대로 특정 기간 동안 승인된 각 이벤트에 대해 계산 함수를 실행합니다. 특정 기간은 <timeWindow> 요소로 정의된 대로 필수 시간 창으로 표시됩니다.

규칙 응답 실행 조건

계산 패턴을 사용하면 규칙 응답은 <onTimeWindowComplete> 요소에 의해 정의된 대로 시간 창이 완료될 때 실행됩니다. 계산값은 <onTimeWindowComplete> 조치 동안에 사용 가능합니다.

이 규칙 패턴의 사용 예제

어플리케이션이 고객 주문 이벤트를 처리 중이라고 가정하십시오. 계산 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

> 이벤트를 받을 때마다 총 주문 값이 지정된 기간 동안 발생한 모든 주문의 총 값에 추가되고 갱신된 모든 총 주문 값이 사용자 인터페이스 내에 게시됩니다.

관련 참조

54 페이지의『계산 규칙 요약』
이 요약에는 계산 규칙의 모든 언어 요소가 나열되어 있습니다.

중복 패턴

중복 규칙은 중복 패턴에 의해 정의됩니다. 이는 지정된 시간 간격 내에서 승인된 두 번째 및 후속 이벤트는 포함하지만 이러한 이벤트의 규칙 세트 처리는 건너뜁니다. 이는 상태 규칙입니다.

개요

중복 패턴은 일반적으로 특정 기간 동안 유사한(중복) 이벤트를 분리하는 데 사용됩니다. 중복 이벤트는 어떤 면에서는 이전 이벤트와 유사하지만 해당 이벤트와 정확히 일치할 필요는 없습니다. 이벤트는 단순히 규칙의 이벤트 선택 기준에 부합하는 경우에만 중복됩니다. 시간 기간은 <timeWindow> 요소가 규칙 언어로 정의한 대로 필수 시간 창으로 표시됩니다.

규칙 응답 실행 조건

중복 패턴을 사용하면 규칙 응답은 다음과 같은 경우에 실행됩니다.

- <onDetection> 요소에 의해 정의된 대로 첫 번째 이벤트가 발견될 때
- <onNextEvent> 요소에 의해 정의된 대로 각 중복 이벤트가 처리될 때
- <onTimeWindowComplete> 요소에 의해 정의된 대로 시간 창이 완료될 때

첫 번째 이벤트는 중복 이벤트를 받지 않은 경우에도 <onDetection> 조치를 트리거합니다. 이는 첫 번째 이벤트를 전달하고 중복 이벤트의 규칙 세트 처리를 건너뛰기를 원하는 경우가 있을 수 있기 때문입니다. 이 경우 규칙의 <onDetection> 조치가 트리거될 때 첫 번째 이벤트를 전달하는 규칙 응답 조치를 추가할 수 있습니다.

중복 이벤트(두 번째 및 후속 이벤트)의 기본 처리는 중복 이벤트를 포함하지만 중복 이벤트의 규칙 세트 처리를 건너뛰는 것입니다. 중복 이벤트에 대해 추가로 조치를 취하려면 <onNextEvent> 조치를 명시적으로 정의하면 됩니다. 예를 들어, 어떤 경우에는 중복 이벤트는 데이터베이스나 다른 저장소에 이미 로깅되어 있는 이벤트를 표시합니다. 그러므로 중복 이벤트를 다른 위치에서 제거하기 위해 <onNextEvent> 조치를 코딩하고 싶을 수도 있습니다.

<onTimeWindowComplete> 조치를 처리된 중복 개수를 포함하는 모든 중복 이벤트의 요약 레코드를 작성하는 데 사용할 수 있습니다.

이 규칙 패턴의 사용 예제

『Denial of service』 메시지가 동일한 자원 유형(보안 모니터)에서 계속 발생한다고 가정해 보겠습니다. 이는 보안 위반 가능성을 표시합니다. 중복 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

> 보안 모니터에서 『Denial of service』 메시지가 발생한 후에 30초 동안 발생한 이벤트의 중복은 포함되지만 운영자 콘솔로 전송되지는 않습니다. 또한 30초가 지난 후에 규칙은 해당 기간 동안 발생한 『Denial of service』 메시지 개수를 표시하는 이벤트를 생성합니다.

관련 참조

56 페이지의 『중복 규칙 요약』
이 요약에는 중복 규칙의 모든 언어 요소가 나열되어 있습니다.

필터 패턴

필터 규칙은 필터 패턴으로 정의됩니다. 이는 이벤트를 승인할 때 특정 조치를 수행합니다. 이는 단일 이벤트에 대해서만 작동하므로 상태가 없는 규칙입니다.

개요

필터 패턴은 이벤트 선택 기준에 부합하는 개별 이벤트에서 작동하는데 사용됩니다. 다른 규칙 패턴과는 다르게 이는 연관된 상태 정보(예: 과거 이벤트의 히스토리)를 보관하지 않습니다.

규칙 응답 실행 조건

필터 패턴을 사용하면 규칙 응답은 <onDetection> 요소에 의해 정의된 대로 이벤트 선택 기준에 부합하는 이벤트를 받을 때 실행됩니다.

이 규칙 패턴의 사용 예제

필터 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

> `ServerStatus` 이벤트가 95%가 넘는 `serverLoad`를 표시하면 규칙은 관리자를 호출하는 조치를 실행합니다.

관련 참조

58 페이지의 『필터 규칙 요약』
이 요약에는 필터 규칙의 모든 언어 요소가 나열되어 있습니다..

연속 패턴

연속 규칙은 연속 패턴에 의해 정의됩니다. 이는 이벤트의 특정 순서가 시간 간격 내에 도달하는지 여부를 감지합니다. 연속된 순서가 있거나 무작위일 수 있습니다. 연속 규칙은 상태 규칙입니다.

개요

연속 패턴은 일정 기간 동안 이벤트의 순서를 확인하고 순서가 완료되었는지 여부를 감지합니다. 완료되지 않은 순서는 지정된 순서에 하나 이상의 이벤트를 포함하지만 모두를 포함하지는 않습니다.

시간 기간은 <timeWindow> 요소가 규칙 언어로 정의한 대로 필수 시간 창으로 표시됩니다. 순서에서 각 이벤트는 규칙 내의 별도의 <eventSelector> 요소로 정의됩니다. 이벤트의 순서는 다음과 같은 순서로 발견될 수 있습니다.

- 규칙에서 <eventSelector> 요소가 코딩된 순서. 이 경우에는 규칙이 첫 번째 <eventSelector> 요소에 의해 정의된 이벤트를 발견하면 순서 감지가 시작됩니다. 이제 규칙은 두 번째 <eventSelector> 요소가 정의한 이벤트를 기다립니다.
- 무작위 순서. 이 경우에는 규칙이 <eventSelector> 요소에 의해 정의된 이벤트 중 하나를 발견하면 순서 감지가 시작됩니다. 이제 규칙은 <eventSelector> 요소에 의해 정의된 또 다른 이벤트 중 하나를 기다립니다.

연속 패턴은 다음과 같은 기본 방법에서 다른 규칙 패턴과 다릅니다.

- 규칙이 어떤 이벤트를 승인하는지 정의하기 위한 여러 개의 <eventSelector> 요소가 있습니다. 최소 두 개의 <eventSelector> 요소가 필요합니다.
- 이벤트가 <eventSelector> 요소 중 하나가 정의한 기준에 부합하면 해당 <eventSelector> 요소는 해당 규칙 인스턴스 내에서 이후의 이벤트 처리에서 제외됩니다.
- <eventSelector> 요소의 별명 속성은 연속 규칙 내에서만 올바르며 연속 규칙에서 특정 이벤트 선택자가 선택한 이벤트에 고유한 이름을 지정합니다. 필터링 선언문 또는 조치 내의 표현식에서 act_eventList 변수를 사용하여 별명 이름으로 연속 규칙의 이벤트에 액세스할 수 있습니다.

규칙 응답 실행 조건

연속 패턴을 사용하면 규칙 응답은 다음과 같은 시간에 실행됩니다.

- <onDetection> 요소에 의해 정의된 대로 이벤트의 전체 순서가 시간 창 내에서 감지될 때
- <onTimeOut> 요소에 의해 정의된 대로 하나 이상의 이벤트가 도달했지만 전체 순서가 시간 창 내에 도달하지 않았을 때

연속 패턴은 해당 순서가 완료되지 않았음을 발견하는 데 유용하게 사용할 수 있습니다. 예를 들어, 『system down』 이벤트가 후속 『system up』 이벤트 없이 발생한 경우, 규칙 작성자는 이러한 누락된 이벤트 유형을 처리하기 위해 <onTimeOut> 조치를 코딩할 수 있습니다.

이 규칙 패턴의 사용 예제

전체 순서 감지를 설명하는 시나리오:

IT 환경에서, 관리자는 DB2® 힙 *크기* 값이 WebSphere® Application Server에 영향을 미치는지 여부를 확인한 후, 영향을 미칠 경우에 이 문제점을 정정하려고 합니다. 지정된 기간 동안에 다음 이벤트가 다음 순서대로 발생할 경우, 관리자는 DB2 힙 *크기*를 늘린 후 데이터베이스 관리자를 다시 시작합니다.

1. WebSphere Application Server 자원 할당 예외. 이 경우, 이벤트가 `WASResourceAllocationException` 유형이라고 가정합니다.
2. 『힙 부족으로 인해 명령문을 처리할 수 없음』이라는 DB2 오류 메시지가 발생합니다. 이 경우, 이벤트가 `DB2NotEnoughHeap` 유형이라고 가정합니다.

이 시나리오의 경우, 연속 규칙에 두 개의 <eventSelector> 요소가 정의되어 있고 이벤트는 무작위 순서가 아니라 <eventSelector> 요소가 코딩된 순서대로 도달해야 합니다. 첫 번째 <eventSelector> 요소에서 `WASResourceAllocationException` 이벤트를 확인하고 두 번째 <eventSelector> 요소에서 `DB2NotEnoughHeap` 이벤트를 확인합니다. 다음 이벤트가 지정된 시간 창 내에서 시스템에 제공되었다고 가정합니다.

1. `WASResourceAllocationException`
2. `DB2BackupStarted`
3. `WASResourceAllocationException`
4. `WASResourceAllocationException`
5. `DB2NotEnoughHeap`

규칙 작동은 다음과 같습니다.

1. 첫 번째 이벤트인 `WASResourceAllocationException`을 승인합니다. 첫 번째 <eventSelector> 요소의 기준에 부합하므로 첫 번째 <eventSelector> 요소는 이제 이 규칙 내에서 이후의 이벤트 처리에서 제외됩니다.
2. 두 번째 이벤트인 `DB2BackupStarted`는 무시합니다.
3. 세 번째 이벤트인 `WASResourceAllocationException`은 무시합니다.
4. 네 번째 이벤트인 `WASResourceAllocationException`은 무시합니다.
5. 다섯 번째 이벤트인 `DB2NotEnoughHeap`을 승인하고 순서를 완료합니다. <onDetection> 규칙 응답 조치가 실행됩니다. 이 조치는 DB2 힙 *크기*의 값을 늘리고 데이터베이스 관리자를 다시 시작하도록 정의되어 있습니다. 이 규칙은 초기 상태로 돌아갑니다.

첫 번째 <eventSelector> 요소가 이제는 이후에 이 규칙이 처리하는 이벤트에 포함됩니다.

완료되지 않은 순서 감지를 설명하는 시나리오:
비즈니스 조직에서는 모든 고객의 주문을 수령한 후 한 시간 이내에 주문이 배송 준비가 되기를 원하며 그렇지 않은 경우에는 통지를 받고자 합니다.

이 시나리오의 경우, 연속 규칙에 두 개의 <eventSelector> 요소가 정의되어 있고 이벤트는 무작위 순서가 아니라 <eventSelector> 요소가 코딩된 순서대로 도달해야 합니다. 첫 번째 <eventSelector> 요소는 `operationType=Order`가 있는 `Netsales` 이벤트를 확인하고 두 번째 <eventSelector> 요소는 `operationType=Delivery`가 있는 `Netsales` 이벤트를 확인합니다. 다음 이벤트가 지정된 한 시간의 시간 창 내에서 시스템에 제공되었다고 가정합니다.

1. `operationType=Order`가 있는 `Netsales` 이벤트
2. `operationType=Order`가 있는 `Netsales` 이벤트

규칙 작동은 다음과 같습니다.

1. 첫 번째 이벤트를 승인합니다. 첫 번째 <eventSelector> 요소의 기준에 부합하므로 첫 번째 <eventSelector> 요소는 이제 이 규칙 내에서 이후의 이벤트 처리에서 제외됩니다.
2. 두 번째 이벤트는 무시합니다.
3. `operationType=Delivery`가 있는 `Netsales` 이벤트가 지정된 시간 창 내에 수신되지 않았기 때문에 <onTimeOut> 규칙 응답 조치가 실행됩니다. 이 조치는 비즈니스 경영진에게 고객의 주문이 수령된 후 1시간 이내에 배송 준비가 되지 않았음을 통지하기 위해 정의됩니다. 이 규칙은 초기 상태로 돌아갑니다.

   첫 번째 <eventSelector> 요소가 이제는 이후에 이 규칙이 처리하는 이벤트에 포함됩니다.

   관련 개념

   24 페이지의『이벤트 관련 정보에 액세스』
   다음 예제는 ACT(Active Correlation Technology)가 제공하는 변수를 통해 이벤트 관련 정보에 액세스하는 방법을 보여줍니다.

   관련 참조

   59 페이지의『연속 규칙 요약』
   이 요약에는 연속 규칙의 모든 언어 요소가 나열되어 있습니다.

임계값 패턴

임계값 규칙은 임계값 패턴에 의해 정의됩니다. 이는 시간 간격 내에서 선택한 이벤트 그룹을 수집하고 각 이벤트를 받은 후에 임계값 조건에 부합하는지 여부를 판별합니다. 이는 상태 규칙입니다.

개요

임계값 패턴은 특정 기간 동안 임계값에 도달할 때까지 이벤트를 수집합니다. 시간 기간은 <timeWindow> 요소가 규칙 언어로 정의한 대로 필수 시간 창으로 표시됩니다.

임계값 패턴은 임계값 유형에 다음과 같은 세 가지의 옵션을 제공합니다.

이벤트 계수 임계값

이 임계값 유형을 사용하면 특정 기간 동안 이벤트 선택 기준에 부합해야 하는 이벤트 개수를 정의할 수 있습니다. 정의된 임계값은 승인된 이벤트 개수와 비교됩니다. 이벤트 계수가 시간 창 내에서 정의된 한계와 동일하면 임계값에 도달한 것입니다.

이 임계값 유형은 매우 단순한 이벤트 계수 확인에 유용합니다. 예를 들어, 다음과 같은 질문에 응답할 수 있습니다. 『1분 내에 다섯 번의 로그인 실패 이벤트가 발생했습니까?』

이 임계값은 <eventCountThreshold> 요소에 의해 정의됩니다. <eventCountThreshold> 요소는 또한 시간 창에서 다음과 같은 두 개의 가능한 시간 간격 모드 중 하나를 지정합니다.

고정 간격

고정 간격은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 받을 때 시작되고 다음 중 하나가 발생하면 종료됩니다.

- 규칙이 지정된 지속 기간 내에서 임계값에 도달할 때
- 지정된 지속 기간이 경과된 경우

슬라이딩 간격

슬라이딩 간격은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 받을 때 시작됩니다. 그러나 규칙이 임계값에 도달하지 않고 지정된 지속 기간이 경과된 경우, 시간 창은 시작 시간을 새로운 『첫 번째』 이벤트의 시작 시간으로 조정(슬라이드)합니다. 이는 일반적으로 다음 번에 승인되는 이벤트입니다. 슬라이딩 간격은 다음 중 하나가 발생할 때까지 이러한 방법으로 계속해서 조정합니다.

- 규칙이 지정된 지속 기간 내에서 임계값에 도달할 때
- 시간 창을 시작한 이벤트를 받은 후에 지정된 지속 기간 내에 후속 이벤트를 받지 않은 경우

시간 창을 시작한 이벤트(새로운 『첫 번째』 이벤트가 됨)는 이 기준에 부합하는 수신 시간이 있는 이벤트입니다. 규칙의 시간 간격 지속 기간에 추가된 수신 시간은 현재 시간 이후입니다. 다음은 방정식 양식으로 된 기준입니다.

```
event reception time + time interval duration for rule > current time
```

이러한 이벤트가 없으면 슬라이딩 간격이 더 이상 시간을 조정하지 않고 간격이 종료됩니다.

계산된 임계값

이 임계값 유형을 사용하면 승인된 각 이벤트에 대해 계산을 수행하는 코드를 작성하거나 다른 누군가가 작성한 코드를 사용할 수 있고, 이전에 정의된 변수에 보관된 계산된 임계값을 리턴합니다. 그런 다음 임계값에 도달했는지 여부를 판별하기 위해 이 계산된 임계값을 정의된 임계값과 비교합니다.

그러므로 이전 이벤트에서 저장된 데이터를 사용하여 계산된 임계값을 작성하거나 갱신하기 위해 복잡한 계산을 적용할 수 있으며, 규칙 작성자는 계산된 임계값을 계산하는 논리와 독립적으로 정의된 임계값을 설정할 수 있습니다.

이 임계값 유형은 값의 집합이나 정의된 임계값과의 비교 시에 유용합니다. 예를 들어, 특정 기간 동안 특정 고객에게 판매한 금액의 합계를 계산하고 이 합계를 정의된 임계값과 비교하는 데 사용할 수 있습니다.

이 임계값은 <computedThreshold> 요소에 의해 정의됩니다.

부울 임계값

이 임계값 유형을 사용하면 승인된 각 이벤트에 대해 `true` 또는 `false` 값을 리턴하는 코드를 작성하거나 다른 누군가가 작성한 코드를 사용할 수 있습니다. 값이 `true`이면 임계값에 도달한 것입니다. 값이 `false`이면 임계값 규칙은 기간이 종료될 때까지 또는 다른 이벤트를 승인할 때까지 계속해서 처리합니다.

이 임계값 유형은 값의 범위를 확인하는 데 유용합니다. 예를 들어, CPU 활용도가 항상 30% - 80% 사이여야 하면 이 임계값은 활용도가 해당 범위에 있는지를 지속적으로 확인할 수 있습니다.

이 임계값은 <booleanThreshold> 요소에 의해 정의됩니다.

규칙 응답 실행 조건

임계값 패턴을 사용하면 규칙 응답은 다음과 같은 시간에 실행됩니다.

- <onDetection> 요소에 의해 정의된 대로 임계값에 도달한 경우.
- <onTimeOut> 요소에 의해 정의된 대로 하나 이상의 이벤트가 승인되었지만 시간 창 내에서 임계값에 도달하지 않은 경우.

이 규칙 패턴의 사용 예제

이벤트 계수 임계값이 있는 임계값 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

30초의 슬라이딩 시간 간격 내에서 동일한 서브네트워크에서 다섯 개 이상의 `Server unreachable` 이벤트가 시작되면 규칙은 라우터의 상태를 확인하기 위해 조치를 실행합니다.

관련 참조

61 페이지의『임계값 규칙 요약』
이 요약에는 임계값 규칙의 모든 언어 요소가 나열되어 있습니다.

타이머 패턴

타이머 규칙은 타이머 패턴에 의해 정의됩니다. 이는 정기적인 간격으로 조치를 시작합니다. 이는 상태 규칙입니다. 타이머 규칙은 이벤트를 처리하지 않지만 이벤트에 의해 활성화되거나 비활성화될 수 있습니다.

개요

타이머 패턴은 기간 시작 시에 시작하고 기간 종료 시에 중지하는 타이머와 유사합니다. 시간 기간은 <timeWindow> 요소가 규칙 언어로 정의한 대로 필수 시간 창으로 표시됩니다.

반복하지 않도록 지정되지 않는 한 타이머 패턴은 타이머 규칙이 비활성화될 때까지 반복합니다. 그러므로 타이머 규칙이 시작되면 조치를 시작하기 전에 지정된 기간을 기다리고 이 규칙이 비활성화되거나 ACT(Active Correlation Technology) 엔진이 종료할 때까지 이 작동을 반복합니다.

타이머 규칙은 이벤트 선택 기준이 포함되어 있지 않다는 점에서 독특합니다. 타이머 규칙은 <activationInterval> 요소에 의해 정의된 대로 규칙의 활성화 간격에 따라 처리를 시작합니다. 기본 <activationInterval> 요소가 사용되고 타이머 패턴이 반복하도록 설정된 경우에는 타이머 규칙은 ACT(Active Correlation Technology) 엔진에 의해 로드될 때 시작되고 ACT(Active Correlation Technology) 엔진이 종료할 때 중지합니다. 이벤트와 함께 타이머 규칙을 활성화하려면 규칙의 <activationInterval> 요소 내의 <activateOnEvent> 요소에 이벤트를 지정해야 합니다.

규칙 응답 실행 조건

타이머 패턴을 사용하면 규칙 응답은 <onTimeWindowComplete> 요소에 의해 정의된 대로 시간 창이 완료될 때 실행됩니다.

이 규칙 패턴의 사용 예제

타이머 패턴은 정리 규칙을 구현하는 데 유용합니다. 타이머 패턴의 사용 예제는 다음을 수행하는 규칙입니다.

30분마다 규칙은 48시간 이상 열려 있는 무해하고 정보성인 이벤트를 정리하는 조치를 실행합니다.

관련 참조

63 페이지의『타이머 규칙 요약』
이 요약에는 타이머 규칙의 모든 언어 요소가 나열되어 있습니다.

서로 다른 규칙 패턴의 공통 및 고유 측면

이 메트릭스는 서로 다른 규칙 패턴의 공통 및 고유 측면에 대한 높은 수준의 개요를 제공합니다.

표 2에서는 규칙의 기본 언어 요소를 나열하고 각 규칙 유형의 열에 요소가 올바른 경우 X를 표시합니다. 기본 언어 요소는 서로 다른 규칙 유형의 직접 하위 요소입니다. 목록에는 규칙 유형에 따라 다른 이러한 직접 하위 요소에 포함된 요소는 나열되지 않습니다. 또한 특정 요소 속성의 유효성은 규칙 유형에 따라 다를 수 있습니다.

표 2. 서로 다른 규칙 패턴의 공통 및 고유 측면을 보여주는 메트릭

요소	콜렉션	계산	중복	필터	연속	임계값	타이머
<comment>	X	X	X	X	X	X	X
<variable>	X	X	X	X	X	X	X
<activationInterval>	X	X	X	X	X	X	X
<lifeCycleActions>	X	X	X	X	X	X	X
<eventSelector>	X	X	X	X	X	X	
<groupingKey>	X	X	X		X	X	
<timeWindow>	X	X	X		X	X	X
<computeFunction>		X					
<booleanThreshold>						X	
<computedThreshold>						X	
<eventCountThreshold>						X	
<onDetection>			X	X	X	X	
<onNextEvent>			X				
<onTimeOut>					X	X	
<onTimeWindowComplete>	X	X	X				X

표현식

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

규칙 작성자는 컨텍스트 및 원하는 결과에 따라 표현식을 다른 용도로 코딩할 수 있습니다. 표현식은 변수의 초기화, 이벤트 선택 기준의 정의 및 규칙 응답 및 수명 주기 조치의 스펙에 자주 사용됩니다.

표현식을 포함하는 언어 요소

표현식을 포함하는 각 언어 요소에는 표현식을 작성하는 데 사용한 프로그래밍 언어를 식별하는 expressionLanguage 속성이 있습니다. Java 프로그래밍 언어는 유일하게 지원하는 표현식 언어입니다.

표현식은 다음과 같은 규칙 언어 요소 내에 포함될 수도 있습니다.

- 규칙 세트, 규칙 블록 또는 규칙 변수의 <varInitializer>

- <eventSelector>의 <filteringPredicate>
- <groupingKey>의 <computedValue>
- 계산 규칙의 <computeFunction>
- 임계값 규칙의 <booleanThreshold>
- 임계값 규칙의 <computedThreshold>
- 규칙의 규칙 응답 조치:
  - <onDetection> 내의 <action>. 이 조치는 중복, 필터, 연속 및 임계값 규칙에서만 유효합니다.
  - <onNextEvent> 내의 <action>. 이 조치는 중복 규칙에서만 유효합니다.
  - <onTimeOut> 내의 <action>. 이 조치는 연속 및 임계값 규칙에서만 유효합니다.
  - <onTimeWindowComplete> 내의 <action>. 이 조치는 콜렉션, 계산, 중복 및 타이머 규칙에서만 유효합니다.
- 규칙의 수명 주기 조치:
  - <onLoad> 내의 <action>
  - <onActivation> 내의 <action>
  - <onDeactivation> 내의 <action>
  - <onUnload> 내의 <action>

표현식 코딩을 돕기 위해 ACT(Active Correlation Technology)가 제공하는 기능

규칙 작성자의 표현식 코딩을 돕기 위해 ACT(Active Correlation Technology)는 다음과 같은 기능을 제공합니다.

- 표현식에 사용할 외부 모듈(예: Java 클래스) 및 오브젝트를 가져옵니다.
- 규칙 세트, 규칙 블록 또는 규칙 변수를 초기화하고 액세스합니다.
- act_event 변수를 통해 규칙이 처리 중인 현재 이벤트에 액세스합니다.
- act_eventCount 변수를 통해 규칙이 승인한 이벤트 개수에 액세스합니다.
- act_eventList 변수를 통해 규칙이 승인한 이벤트 목록에 액세스합니다. 여기에는 이벤트의 다양한 속성에 액세스하는 기능과 별명 이름으로 연속 규칙에서 각 이벤트에 액세스하는 기능이 포함됩니다.
- act_lib 변수를 통해 변수를 가져오고 설정하는 기능 및 규칙 세트를 통해 이벤트 플로우를 제어하는 기능을 포함하는 메소드에 액세스합니다.
- act_location 변수를 통해 규칙 계층 구조 내에서 표현식의 위치에 액세스합니다.
- act_nodeName 변수를 통해 노드의 완전한 이름에 액세스합니다.
- act_threshold 변수를 통해 임계값 규칙의 정의된 임계값에 액세스합니다.

외부 모듈 및 오브젝트 가져오기 및 액세스

이 예제는 외부 코드(예: Java 클래스) 및 외부 오브젝트에서 표현식에 액세스를 가능하게 하는 방법을 보여줍니다. 외부 오브젝트는 어플리케이션과 표현식과의 통신을 위해 작성된 오브젝트입니다.

표현식에서 외부 코드에 액세스하려면 먼저 코드가 표현식에 액세스 가능해야 합니다.

가져오기는 프로그래밍 언어를 사용하여 외부 코드를 표현식에 액세스 가능하게 합니다. <import> 요소에는 규칙 내의 다른 표현식에서 사용하기 위해 가져오기할 외부 모듈(예: Java 클래스)을 지정하는 특수 유형의 표현식이 포함됩니다. 가져오기는 규칙 세트나 규칙 블록 레벨에서 정의할 수 있습니다.

다음 <import> 요소에는 다른 표현식에서 참조할 수 있는 StaticHelper 클래스 및 Queue 클래스를 가져오는 Java 프로그래밍 언어로 작성된 표현식이 포함됩니다.

```
<import expressionLanguage="java">
 import com.ibm.act.sample.StaticHelper;
 import com.ibm.act.test.Queue;
</import>
```

import 문에서는 전체 클래스 이름을 사용할 필요는 없지만 컴파일 시간을 줄이려면 전체 이름을 지정해야 합니다. 예를 들어, Java 클래스는 `com.ibm.act.*` 또는 `com.ibm.act.sample.*`가 아닌 `com.ibm.act.sample.StaticHelper`로 지정해야 합니다.

정적 메소드에 액세스

다음 예제는 규칙 응답 조치 내의 표현식이 클래스를 가져온 후에 StaticHelper 클래스를 참조하는 방법을 보여줍니다.

```
<onDetection>
 <action expressionLanguage="java">
  StaticHelper.pageAdministrator("Too many login attempts for " + act_event.getAttribute("userID"));
 </action>
</onDetection>
```

오브젝트용 인스턴스 메소드에 액세스

다음 예제는 클래스를 가져온 후 규칙 응답 조치 내의 표현식이 Queue 클래스를 참조하는 방법을 보여줍니다. 이 예제에서 이름이 `OutputQueueOne`이고 유형이 `Queue`인 외부 오브젝트를 확보하여 특정 대기열에 이벤트를 배치합니다.

```
<onDetection>
 <action expressionLanguage="java">
  Queue myQueue = (Queue)act_lib.getExternalContext("OutputQueueOne");
  myQueue.enqueue(act_event);
 </action>
</onDetection>
```

변수 초기화 및 액세스

이 예제는 규칙 세트, 규칙 블록 또는 규칙 변수를 초기화하고 여기에 액세스할 수 있는 방법을 보여줍니다.

변수는 규칙 세트, 규칙 블록 또는 규칙 레벨에서 정의할 수 있습니다. 변수에 액세스하려면 먼저 초기화 표현식을 사용하여 초기화되어야 합니다. 다음 표현식은 이름이 각각 hostsList 및 hostsString인 두 개의 변수를 초기화합니다.

```
<variable name="hostsList" dataType="java.util.ArrayList">
  <varInitializer expressionLanguage="java">
    return new ArrayList();
  </varInitializer>
</variable>
<variable name="hostsString" dataType="java.lang.String">
  <varInitializer expressionLanguage="java">
    return new String();
  </varInitializer>
</variable>
```

모든 변수는 표현식을 통해 액세스합니다. 다음 예제는 규칙 응답 조치 내의 표현식을 통해 위의 예제에서 초기화된 hostsList 및 hostsString 변수에 액세스하는 방법을 보여줍니다. 이 예제에서는 hostsList는 수정되고 hostsString에는 새 값이 제공됩니다.

```
<onNextEvent>
 <action expressionLanguage="java">
   String hostname = act_event.getStringAttribute("hostname");
    ArrayList hostsList = (ArrayList)act_lib.getVariable("hostsList");
    hostsList.add(hostname);
    String hostsString = act_lib.getStringVariable("hostsString");
    String newHostString = hostsString + ", " + hostname);
    act_lib.setStringVariable("hostsString", newHostsString);
 </action>
</onNextEvent>
```

이벤트 관련 정보에 액세스

다음 예제는 ACT(Active Correlation Technology)가 제공하는 변수를 통해 이벤트 관련 정보에 액세스하는 방법을 보여줍니다.

현재 이벤트에 액세스하는 예제:
다음 코드는 act_event 변수를 사용하여 이벤트의 호스트 이름 속성을 확보하는 방법을 보여줍니다.

```
act_event.getAttribute("hostname");
```

색인별 이벤트 목록을 통해 이벤트에 액세스하는 예제:
다음 코드는 act_eventList 변수를 사용하여 이벤트 목록의 첫 번째 이벤트를 확보하는 방법을 보여줍니다.

```
act_eventList.get(0);
```

별명별 이벤트 목록을 통해 이벤트에 액세스하는 예제:
다른 규칙 유형과는 달리 연속 규칙에서는 여러 이벤트 선택자를 사용할 수 있고 실제로 최소 두 개의 이벤트 선택자가 필요합니다. <eventSelector> 요소의 별명 속성은 연속 규칙 내에서만 올바르며 연속 규칙에서 특정 이벤트 선택자가 선택한 이벤트에 고유

한 이름을 지정합니다. 필터링 선언문 또는 조치 내의 표현식에서 act_eventList 변수를 사용하여 별명 이름으로 연속 규칙의 이벤트에 액세스할 수 있습니다.

다음 코드는 연속 규칙의 두 개의 이벤트 선택자를 표시합니다. 별명 이름은 TECevent 및 WASevent입니다.

```
<eventSelector alias="TECevent">
 <eventType type="serverStatus"/>
 <filteringPredicate expressionLanguage="java">
  return act_event.getStringAttribute("source").equals("TEC");
 </filteringPredicate>
</eventSelector>
<eventSelector alias="WASevent">
 <eventType type="serverStatus"/>
 <filteringPredicate expressionLanguage="java">
  return act_event.getStringAttribute("source").equals("WAS");
 </filteringPredicate>
</eventSelector>
```

다음 코드는 act_eventList 변수를 사용하여 TECevent라는 첫 번째 이벤트 선택자가 승인한 이벤트를 확보하는 방법을 보여줍니다.

```
act_eventList.get("TECevent");
```

표현식 코딩을 위한 Best Practices

표현식을 효과적이고 효율적으로 코딩할 수 있게 하는 몇 가지 Best Practices, 힌트 및 팁을 설명합니다.

- 제공된 표현식 예제는 이해하기 쉽도록 하기 위해 대부분 XML 내에 Java 코드가 포함되어 있습니다. 그러나, 실제로 규칙을 작성할 때에는 Java 코드는 외부 모듈에 작성하고 이러한 외부 모듈을 표현식의 일부로서 호출할 것을 권장합니다.

  기존의 조각을 사용 또는 편집하거나 규칙 빌더에 새 조각을 작성하여 외부 모듈을 호출하기 위한 코드를 제공할 수 있습니다. 조각은 표현식 내에서 사용할 수 있는 소스 코드의 인용구입니다. 규칙 빌더에서는 조각 보기를 통해 조각을 사용할 수 있습니다.

  이 방법을 사용하면 사용자가 선택한 IDE(Integrated Development Environment)를 통해 Java 코드의 설계, 개발, 편집, 테스트 및 디버깅 태스크를 수행하고 개발 프로세스의 일부로 제어할 수 있습니다.

- 표현식 코드가 XML로 구문 분석되는 것을 막으려면 코드를 CDATA 섹션으로 묶으십시오. 여기서 다음 예제와 같이 <![CDATA[는 코드 바로 앞에 오고]]>는 코드 바로 뒤에 나옵니다.

  ```
  <onTimeOut>
  <action expressionLanguage="java">
  <![CDATA[
    IEvent firstEvent = act_eventList.get(0);
    System.out.println("Expired Item: " + firstEvent.getAttribute("sourceComponentId.location"));
  ]]>
  </action>
  </onTimeOut>
  ```

XML 구문 분석기는 CDATA 섹션 내부의 모든 내용을 무시합니다.

- 표현식에서 act_lib.getExternalContext() 메소드를 사용하는 경우, 규칙 세트 또는 규칙 블록 변수의 메소드로부터 리턴된 오브젝트를 저장하지 마십시오. 이유는 어플리케이션이 오브젝트 인스턴스에 대한 참조를 변경할 수 있으나 연관된 규칙 세트 또는 규칙 블록 변수는 갱신되지 않기 때문입니다.
- 표현식에서 <action> 요소 내에 return 문(`return;`)을 사용하고, 개별 규칙 응답이나 수명 주기 조치에 대해 추가로 <action> 요소가 코딩되어 있으면 코드 실행은 return 문이 코딩된 곳에서 종료되고 표현식에서 다음 <action> 요소에서 다시 시작됩니다.
- 규칙 관리 및 기타 ACT(Active Correlation Technology) 엔진 메소드는 규칙 응답이나 수명 주기 조치 내에서 호출할 수 없습니다.

관련 정보

23 페이지의『외부 모듈 및 오브젝트 가져오기 및 액세스』
이 예제는 외부 코드(예: Java 클래스) 및 외부 오브젝트에서 표현식에 액세스를 가능하게 하는 방법을 보여줍니다. 외부 오브젝트는 어플리케이션과 표현식과의 통신을 위해 작성된 오브젝트입니다.

48 페이지의『규칙의 표현식에 조각 포함』
규칙 내의 표현식에 Eclipse Workbench의 조각 보기에 있는 조각을 포함시킬 수 있습니다.

변수

규칙 언어에서 특정 변수는 서로 다른 이벤트 발생이나 규칙에 이벤트 관련 정보를 저장하는 데 사용됩니다. 그러면 이 이벤트 관련 정보는 규칙 내의 표현식에서 액세스할 수 있습니다. 일부 변수 유형은 규칙 작성자가 정의하고 나머지 유형은 ACT(Active Correlation Technology)가 제공합니다. 일부 유형은 표현식 내에서 직접 액세스할 수 있으며 나머지 유형은 ACT(Active Correlation Technology)가 제공한 메소드를 통해서만 액세스할 수 있습니다.

<variable> 요소 내에 정의되어 있고 메소드를 통해 액세스되는 변수

규칙, 규칙 블록 또는 규칙 세트의 <variable> 요소 내에 변수를 정의할 수 있습니다. 다음 메소드 중 하나를 사용하여 표현식 내의 변수에 액세스할 수 있습니다.

- getVariable() 메소드 또는 get*javatype*Variable() 메소드 중 하나
- setVariable() 메소드 또는 set*javatype*Variable() 메소드 중 하나

예를 들어, 규칙의 <variable> 요소 내에 *rule_writer_variable* 변수를 정의할 경우 다음 코드를 사용하여 이 변수에 액세스할 수 있습니다.

```
int sample_variable = act_lib.getIntVariable("rule_writer_variable");
```

ACT(Active Correlation Technology)가 제공하고 표현식 내에서 직접 액세스되는 변수

다음 변수는 ACT(Active Correlation Technology)가 제공합니다. 표현식 내에서 이러한 변수를 즉시 사용할 수 있습니다.

- act_event
- act_eventList
- act_lib

예를 들어, 다음 코드를 사용하여 act_event 변수에 액세스하여 이벤트의 호스트 이름 속성을 확보할 수 있습니다.

```
act_event.getAttribute("hostname");
```

ACT(Active Correlation Technology)가 제공하고 메소드를 통해 액세스되는 변수

다음 변수는 ACT(Active Correlation Technology)가 제공합니다. getVariable() 메소드 및 get*javatype*Variable() 메소드 중 하나를 사용하여 표현식 내의 이러한 변수에 액세스할 수 있습니다.

- act_eventCount
- act_location
- act_nodeName
- act_threshold

예를 들어, 다음 코드를 사용하여 act_eventCount 변수에 액세스할 수 있습니다.

```
int eventcount_integer = act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);
```

표 3에서는 이러한 변수에 대해 IACTLibrary 인터페이스가 제공하는 상수를 보여줍니다. 코드에서, 런타임이 아닌 컴파일 중에 맞춤법 또는 인쇄상의 오류를 발견하려면 항상 변수 자체보다는 이러한 변수를 나타내는 상수를 사용하십시오. 예를 들어, `act_lib.getIntVariable("act_eventCount");`보다는 `act_lib.getIntVariable (IACTLibrary.EVENTCOUNT);`를 사용하십시오.

표 3. 상수에 연관된 변수

변수	연관된 상수
act_eventCount	EVENTCOUNT
act_location	LOCATION
act_nodeName	NODENAME
act_threshold	THRESHOLD

관련 참조

124 페이지의『variable 요소』

<variable> 요소는 변수를 정의하고 표현식이 참조할 수 있는 양식으로 된 정보를 포함합니다. 변수는 규칙 세트, 규칙 블록 또는 규칙 레벨에서 정의할 수 있습니다.

ACT(Active Correlation Technology) 변수의 데이터 유형

ACT(Active Correlation Technology)에서 제공하는 변수는 다른 데이터 유형을 갖고 있습니다.

표 4에서는 이러한 변수의 데이터 유형을 보여줍니다.

표 4. ACT(Active Correlation Technology) 변수의 데이터 유형

변수	데이터 유형
act_ event	com.ibm.correlation.IEvent 인터페이스에 정의된 유형
act_ eventCount	int
act_ eventList	com.ibm.correlation.IEventList 인터페이스에 정의된 유형
act_ lib	com.ibm.correlation.IACTLibrary 인터페이스에 정의된 유형
act_ location	java.lang.String
act_ nodeName	java.lang.String
act_ threshold	이 변수는 임계값 규칙의 <computedThreshold> 또는 <eventCountThreshold> 요소에 대한 임계값 속성의 값이므로 데이터 유형이 임계값 속성값의 데이터 유형과 동일해야 합니다.

변수가 올바른 표현식 컨텍스트

ACT(Active Correlation Technology)에서 제공하는 변수는 특정 표현식 컨텍스트에만 적용됩니다.

표 5에 이 변수에 사용할 수 있는 표현식 컨텍스트가 나열되어 있습니다. 테이블에서는 각 표현식 컨텍스트에 사용 가능한 각 변수의 열에 X를 표시합니다. 이러한 변수에 적용되는 추가적인 사용 제한사항은 각 변수에 대한 설명을 참조하십시오.

표 5. 변수에 사용 가능한 표현식 컨텍스트

표현식 컨텍스트	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
<onActivation> 내의 <action>	X			X	X	X	X
<onDeactivation> 내의 <action>	X	X	X	X	X	X	X
<onDetection> 내의 <action>	X	X	X	X	X	X	X
<onLoad> 내의 <action>				X	X	X	X
<onNextEvent> 내의 <action>	X	X	X	X	X	X	
<onTimeOut> 내의 <action>		X	X	X	X	X	X
<onTimeWindowComplete> 내의 <action>		X	X	X	X	X	
<onUnload> 내의 <action>				X	X	X	X
<booleanThreshold>	X	X	X	X	X	X	
<computedThreshold>	X	X	X	X	X	X	X
<computedValue>	X			X	X	X	
<computeFunction>	X	X	X	X	X	X	
<filteringPredicate>	X	X	X	X	X	X	X

표 5. 변수에 사용 가능한 표현식 컨텍스트 (계속)

표현식 컨텍스트	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
규칙의 <varInitializer>				X	X	X	X

act_event 변수

act_event 변수는 현재 이벤트에 적용되는 메소드에 대한 액세스를 제공합니다.

세부사항

타이머 규칙이 이벤트를 처리하지 않으므로 타이머 규칙 내의 act_event 변수는 규칙을 활성화하거나 비활성화하는 이벤트에만 적용됩니다.

act_event 변수는 이벤트가 규칙을 활성화 또는 비활성화 시키는 경우에만 <onActivation> 및 <onDeactivation> 조치에 적용됩니다. 그렇지 않으면 이 변수의 값은 널(null)입니다.

코딩 예제

다음 코드는 act_event 변수에 액세스하여 이벤트의 호스트 이름 속성을 가져옵니다.

```
String host = act_event.getStringAttribute("hostname");
```

액세스할 수 있는 메소드

act_event 변수가 액세스를 제공하는 메소드는 30 페이지의 표 6에 표시된 대로 IEvent 인터페이스에 정의되어 있습니다.

표 6. IEvent 인터페이스, 해당하는 메소드 및 Javadoc 메소드 위치 설명

인터페이스	메소드	**Javadoc 메소드 위치 설명**
IEvent	• get • getAttribute • getBooleanAttribute • getByteAttribute • getShortAttribute • getIntAttribute • getLongAttribute • getFloatAttribute • getDoubleAttribute • getStringAttribute • set • getTimeStamp • setTimeStamp • getType • getOriginal	com.ibm.correlation.IEvent

act_eventCount 변수

act_eventCount 변수는 규칙이 승인한 이벤트 개수와 동일한 정수입니다.

세부사항

중복 규칙의 경우 act_eventCount 변수의 값은 승인된 이벤트 총 개수이며 여기에는 원래 이벤트와 모든 중복이 포함됩니다. 모든 다른 규칙 유형의 경우 값은 이벤트 목록의 크기와 동일하며 이는 act_eventList.size() 메소드를 사용하여 act_eventList 변수를 통해 구할 수 있습니다.

act_eventCount 및 act_eventList 변수는 타이머 규칙이 이벤트를 처리하지 않으므로 타이머 규칙 내에서 올바르지 않습니다.

규칙이 그룹화 키로 정의된 경우에는 act_eventCount, act_eventList 및 act_threshold 변수는 다음 표현식 컨텍스트 내에서 올바르지 않습니다.

- 수명 주기 조치
- <activateOnEvent> 내의 <filteringPredicate> 또는 <activationInterval> 내의 <deactivateOnEvent>
- <computedValue>

이 경우 원인은 규칙 변수가 규칙 인스턴스에만 적용되고, 규칙 인스턴스가 이러한 표현식 실행 당시 존재하지 않기 때문입니다.

코딩 예제

다음 코드는 act_lib 변수에 액세스하여 규칙이 승인한 이벤트 수를 가져옵니다.

```
int eventCt = act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);
```

act_eventList 변수

act_eventList 변수는 규칙이 승인한 이벤트 목록에 적용되는 메소드에 대한 액세스를 제공합니다.

세부사항

필터 규칙 및 중복 규칙은 둘 모두 항상 하나의 이벤트 목록만 있습니다. 필터 규칙은 상태가 없는 규칙이고 중복 규칙은 첫 번째 분석된 이벤트만을 보존하기 때문입니다.

act_eventCount 및 act_eventList 변수는 타이머 규칙이 이벤트를 처리하지 않으므로 타이머 규칙 내에서 올바르지 않습니다.

규칙이 그룹화 키로 정의된 경우에는 act_eventCount, act_eventList 및 act_threshold 변수는 다음 표현식 컨텍스트 내에서 올바르지 않습니다.

- 수명 주기 조치
- <activateOnEvent> 내의 <filteringPredicate> 또는 <activationInterval> 내의 <deactivateOnEvent>
- <computedValue>

이 경우 원인은 규칙 변수가 규칙 인스턴스에만 적용되고, 규칙 인스턴스가 이러한 표현식 실행 당시 존재하지 않기 때문입니다.

코딩 예제

다음 코드는 act_eventList 변수에 액세스하여 이벤트 목록에서 두 번째 이벤트를 가져옵니다.

```
IEvent second_event = act_eventList.get(1);
```

액세스할 수 있는 메소드

act_eventList 변수가 액세스를 제공하는 메소드는 32 페이지의 표 7에 표시된 대로 IEventList 인터페이스에 정의되어 있습니다.

표 7. IEventList 인터페이스, 해당하는 메소드 및 Javadoc 메소드 위치 설명

인터페이스	메소드	**Javadoc 메소드 위치 설명**
IEventList	• get • size • isEmpty • listIterator	com.ibm.correlation.IEventList

act_lib 변수

act_lib 변수는 ACT(Active Correlation Technology)에서 라이브러리 메소드에 대한 액세스를 제공합니다.

세부사항

act_lib 변수가 액세스할 수 있는 메소드는 변수가 사용된 표현식이 포함되어 있는 규칙 언어 요소에 따라 다릅니다. 표 8의 내용을 참조하십시오.

표 8. act_lib 변수가 액세스할 수 있는 메소드는 포함된 표현식의 컨텍스트에 따라 다릅니다.

표현식 컨텍스트	**IACTLibrary 메소드**	**IExitableActionLibrary 메소드**	**IActionLibrary 메소드**
<onActivation> 내의 <action>	X		
<onDeactivation> 내의 <action>	X		
<onDetection> 내의 <action>	X	X	X
<onLoad> 내의 <action>	X		
<onNextEvent> 내의 <action>	X	X	X
<onTimeOut> 내의 <action>	X		X
<onTimeWindowComplete> 내의 <action>	X		X
<onUnload> 내의 <action>	X		
<booleanThreshold>	X		
<computedThreshold>	X		
<computedValue>	X		
<computeFunction>	X		
<filteringPredicate>	X		
<varInitializer>	X		

코딩 예제

다음 코드는 act_lib 변수에 액세스하여 규칙 세트를 종료하는 메소드를 호출합니다.

```
act_lib.exitRuleSet();
```

액세스할 수 있는 메소드

act_lib 변수가 액세스를 제공하는 메소드는 표 9에 표시된 대로 IACTLibrary, IExitableActionLibrary 및 IActionLibrary 인터페이스에 정의되어 있습니다.

표 9. 인스턴스 및 해당하는 메소드 및 Javadoc 메소드 위치 설명

인터페이스	메소드	**Javadoc 메소드 위치 설명**
IACTLibrary	• trace • getVariable • getBooleanVariable • getShortVariable • getIntVariable • getLongVariable • getFloatVariable • getDoubleVariable • getStringVariable • setVariable • setBooleanVariable • setShortVariable • setIntVariable • setLongVariable • setFloatVariable • setDoubleVariable • setStringVariable • getExternalContext	com.ibm.correlation.IACTLibrary
IActionLibrary	• forward • forwardOnCompletion • activate • deactivate IACTLibrary 인터페이스에 정의되어 있는 메소드는 IActionLibrary 인터페이스에서도 사용 가능합니다.	com.ibm.correlation.IActionLibrary
IExitableActionLibrary	• exitRuleSet • exitRuleBlock IACTLibrary 및 IActionLibrary 인터페이스에 정의되어 있는 메소드는 IExitableActionLibrary 인터페이스에서도 사용 가능합니다.	com.ibm.correlation.IExitableActionLibrary

act_location 변수

act_location 변수는 규칙 계층 구조 내에서 표현식의 위치를 식별하는 문자열입니다.

세부사항

위치는 규칙 계층 구조에서 표현식의 위치를 식별하는 완전한 이름입니다. 여기에는 *identifier.identifier....*가 있습니다. 여기서 *identifier*의 각 발생은 다음 중 하나입니다.

- 각 계층 구조에 있는 XML 요소의 이름 속성값입니다.
- 규칙 블록이나 규칙에서 여러 번 발생하고 이름 속성이 없는 요소의 경우: 표현식을 포함하고 뒤에 대괄호로 묶인 색인 번호가 나오는 XML 요소입니다. 이 색인 번호는 포함하는 요소 내에서 표현식의 위치를 표시합니다. 색인 번호를 지정하기 위한 카운터는 1이 아니라 0으로 시작합니다. 그러므로 예를 들어, 요소가 세 번째 <action> 요소에 포함되어 있는 경우, 색인 번호는 `action[2]`로 표시됩니다.

이러한 ID는 표현식을 포함하는 최상위 레벨 규칙 블록부터 최하위 레벨 요소까지 내림차순으로 되어 있습니다.

코딩 예제

다음 코드는 act_lib 변수에 액세스하여 표현식의 위치를 가져옵니다.

```
String location = act_lib.getStringVariable(IACTLibrary.LOCATION);
```

변수로부터 리턴된 위치의 예제

다음 값은 act_location 변수로부터 리턴된 위치의 예제입니다.

`ruleBlockA.ruleA.eventSelector[3].filteringPredicate`

이 표현식은 다음에 포함되어 있습니다.

- 이름 속성값이 `ruleBlockA`인 규칙 블록
- 이름 속성값이 `ruleA`인 규칙
- 네 번째 <eventSelector> 요소
- <filteringPredicate> 요소

`ruleBlockA.ruleA.onDetection.action[5]`

이 표현식은 다음에 포함되어 있습니다.

- 이름 속성값이 `ruleBlockA`인 규칙 블록
- 이름 속성값이 `ruleA`인 규칙
- <onDetection> 요소
- 여섯 번째 <action> 요소

`ruleBlockA.ruleA.variableA.varInitializer`

이 표현식은 다음에 포함되어 있습니다.

- 이름 속성값이 ruleBlockA인 규칙 블록
- 이름 속성값이 ruleA인 규칙
- 이름 속성값이 variableA인 변수
- <varInitializer> 요소

act_nodeName 변수

act_nodeName 변수는 노드의 완전한 이름을 식별하는 문자열입니다.

세부사항

ACT(Active Correlation Technology)에서 노드는 규칙 세트 내에서 개별적이고 독립적으로 추가하거나, 제거하거나, 대체할 수 있는 규칙 계층 구조 내의 오브젝트입니다. 특히 규칙, 규칙 블록, 규칙 블록 변수 및 규칙 세트 변수와 같은 오브젝트가 노드입니다. 오브젝트는 규칙 레벨 아래에서는 개별적이고 독립적으로 작동할 수 없으므로 규칙 변수는 노드가 아닙니다.

이름 속성값이 ruleBlockA인 규칙 블록 내에 있는 이름 속성값이 rule1인 규칙의 완전한 노드 이름은 ruleBlockA.rule1입니다. 규칙은 규칙 세트 내에서 계층 구조적으로 구성되어 있으므로 완전한 노드 이름에서 하위 레벨 노드로의 하강을 표시하기 위해 마침표(.)를 사용합니다.

코딩 예제

다음 코드는 act_nodeName 변수에 액세스하여 노드의 완전한 이름을 가져옵니다.

```
String nodeName = act_lib.getStringVariable(IACTLibrary.NODENAME);
```

act_threshold 변수

act_threshold 변수는 임계값 규칙의 <computedThreshold> 또는 <eventCountThreshold> 요소에 대해 정의된 임계값인 임계값 속성의 값입니다.

세부사항

act_threshold 변수는 임계값 규칙 내에서만 유효합니다.

규칙이 그룹화 키로 정의된 경우에는 act_eventCount, act_eventList 및 act_threshold 변수는 다음 표현식 컨텍스트 내에서 올바르지 않습니다.

- 수명 주기 조치
- <activateOnEvent> 내의 <filteringPredicate> 또는 <activationInterval> 내의 <deactivateOnEvent>
- <computedValue>

이 경우 원인은 규칙 변수가 규칙 인스턴스에만 적용되고, 규칙 인스턴스가 이러한 표현식 실행 당시 존재하지 않기 때문입니다.

코딩 예제

다음 코드는 act_lib 변수에 액세스하여 정의된 임계값을 가져옵니다.

```
int threshold = act_lib.getIntVariable(IACTLibrary.THRESHOLD);
```

규칙 세트를 통한 이벤트 플로우

이벤트는 규칙 블록과 규칙이 코딩된 순서로 규칙 세트를 통해 흐릅니다. ACT(Active Correlation Technology) 엔진이 이벤트를 받으면 엔진은 이벤트 유형을 판별하고 규칙 활성화, 이벤트 처리 또는 규칙 비활성화를 위해 이 이벤트 유형을 사용하는 규칙을 식별합니다.

규칙이 이벤트를 사용하는 방법

이벤트를 사용하는 각 규칙은 먼저 이벤트가 규칙 활성화, 이벤트 처리 또는 규칙 비활성화를 위해 지정된 모든 기준에 부합하는지 여부를 판별합니다. 기준에 부합하는 경우 규칙은 다음과 같은 조치를 취합니다.

규칙 활성화

규칙의 <onActivation> 요소 내의 조치가 실행됩니다(코딩된 경우).

이벤트 처리

규칙이 이벤트를 처리합니다. 규칙 패턴이 일치하는 경우에는 규칙 응답 조치가 실행됩니다(코딩된 경우). 어떤 경우에는 규칙 응답 조치가 다음을 수행할 수 있습니다.

- 이 조치는 이벤트가 나머지 규칙 블록이나 규칙 세트의 처리를 건너뛰도록 할 수 있습니다.
- 이 조치는 신규 또는 기존 이벤트 처리를 위해 또 다른 규칙이나 규칙 블록으로 전송할 수 있습니다.

규칙 비활성화

규칙의 <onDeactivation> 요소 내의 조치가 실행됩니다(코딩된 경우).

이벤트 플로우에 영향을 미칠 수 있는 메소드

ACT(Active Correlation Technology)는 규칙 세트를 통해 이벤트 플로우에 영향을 미치기 위해 호출할 수 있는 다음과 같은 메소드를 제공합니다. 이러한 메소드는 act_lib 변수를 통해 사용할 수 있습니다.

exitRuleSet

이 메소드는 현재 이벤트가 규칙 세트에서 추가적인 규칙에 의해 처리되지 않도록 지정합니다.

exitRuleBlock

이 메소드는 현재 이벤트가 현재 규칙 블록이나 이 규칙 블록이 포함하는 모든 규칙 블록에서 추가적인 규칙에 의해 처리되지 않도록 지정합니다. 그러나 이 이벤트는 현재 규칙 블록 범위 밖에 있는 추가적인 규칙에 의해 처리됩니다.

forward

이 메소드는 현재 규칙이 처리를 완료하지 않았어도 이벤트가 다른 규칙이나 규칙 블록으로 전송되도록 지정합니다. 그럼 다음 다른 규칙과 규칙 블록은 각각 이벤트를 forward 메소드를 호출한 규칙으로 되돌려 보내기 전에 이벤트를 완전히 처리합니다.

forwardOnCompletion

이 메소드는 현재 규칙이 처리를 완료한 후에 이벤트가 다른 규칙이나 규칙 블록으로 전송되도록 지정합니다.

제 3 장 규칙 작성 개요

이벤트를 연관시키기 위해 규칙을 작성하기 전에 이벤트 연관을 이해하고 계획을 세운 다음 규칙을 설계해야 합니다. ACT(Active Correlation Technology) 규칙 빌더를 사용하면 규칙을 작성하고 규칙 세트를 컴파일할 수 있습니다.

ACT(Active Correlation Technology) 규칙 빌더는 규칙을 작성하기 위한 GUI입니다. 여기에는 온라인 도움말이 포함됩니다. 규칙 빌더에서 결과로 나오는 결과 세트 파일은 actl 파일 유형의 XML 파일입니다.

ACT(Active Correlation Technology)는 변경 관리나 버전 제어 시스템을 제공하지 않습니다.

이벤트 연관 계획

이벤트 연관 계획에는 이벤트 연관의 개념과 이를 어플리케이션에 적용하는 방법을 이해하고 학습하는 과정이 포함됩니다.

다음 개념을 이해했는지 확인하십시오.

- 3 페이지의 제 1 장 『소개』 및 5 페이지의 제 2 장 『규칙 언어 개요』의 정보
- 어플리케이션이 처리하는 이벤트

각 어플리케이션은 다음 예제에 설명된 대로 서로 다른 이벤트 세트를 처리할 수 있습니다.

보험 비즈니스 예제

보험 비즈니스에서는 청구 프로세스 내내 비즈니스 프로세스가 시의적절하게 완료되고 있는지 여부를 판별하기 위해 작업 플로우를 추적하는 이벤트를 생성하고 연관지을 수 있습니다.

영업 예제

서로 다른 유형의 비즈니스에서 영업 결과를 주기적으로 요약하고, 보고하고, 목표치와 비교하여 특정 기간 동안의 영업 목표 달성 상태를 표시할 수 있습니다.

IT 환경 예제

IT 환경에서는 중요한 시스템이 매분 이벤트를 생성하여 데이터베이스 서버가 정상적으로 작동하고 있는지를 표시할 수 있습니다. 연관 규칙을 작성하여 하트 비트 이벤트의 수신을 모니터하고 예상된 하트 비트 이벤트를 받지 못한 경우 특정 규칙 응답 조치를 취할 수 있습니다.

어플리케이션이 처리하는 이벤트의 형식도 이해해야 합니다. ACT(Active Correlation Technology)는 ACT(Active Correlation Technology) 엔진이 처리 중인 이벤트 내의 데이터에 액세스할 수 있는 Java 클래스 및 메소드를 제공합니다. 그러나 이벤트가 처리 중일 때 이벤트에 액세스하거나 변경하기 위해 이러한 클래스와 메소드를 사용하려면 기반 이벤트 오브젝트에 대한 기본적인 이해가 중요합니다.

이벤트 연관을 계획하려면 다음 단계를 수행하십시오.

1. 어플리케이션에서 연관시킬 이벤트를 결정하십시오.
2. 이벤트를 연관하기 위한 규칙 패턴을 결정하십시오.

   규칙 패턴은 특정 이벤트 연관 상황을 표시하며 어떤 식으로든 해당 상황에 기여하는 이벤트를 연관시키는 데 사용할 수 있습니다. 어플리케이션이 처리하는 이벤트가 ACT(Active Correlation Technology) 규칙 언어가 정의하는 규칙 패턴과 어떻게 관련되는지에 대해 생각해 보십시오. 그러면 사용할 규칙 패턴을 결정하는 데 도움이 됩니다.

   항상 이벤트 연관 상황에 가장 적절한 패턴을 사용하십시오. 예를 들어, 규칙 세트가 특정 순서의 이벤트를 발견하게 하려면 필터 규칙의 규칙 응답 조치에 연속 패턴 작동을 포함하는 코드를 작성하지 마십시오. 대신 연속 패턴을 사용하여 연속 규칙을 작성하십시오.
3. 사용할 각 규칙 패턴의 구조를 식별하십시오.

   다음 정보는 규칙 언어의 기본 구조를 요약합니다. 그러나 이러한 구조의 세부사항은 규칙 패턴에서 고유합니다. 이 정보는 규칙 빌더 GUI를 통해 제공된 것과 거의 같은 방법으로 조직되어 있습니다.

   특성 규칙 이름, 설명 및 패턴을 포함하여 규칙 특성의 정의입니다. 자세한 내용은 다음 주제를 참조하십시오.

   - 82 페이지의 『collectionRule 요소』
   - 84 페이지의 『computationRule 요소』
   - 91 페이지의 『duplicateRule 요소』
   - 99 페이지의 『filterRule 요소』
   - 115 페이지의 『sequenceRule 요소』
   - 119 페이지의 『thresholdRule 요소』
   - 122 페이지의 『timerRule 요소』

   변수 각 변수의 이름, 유형, 설명 및 초기화 표현식을 포함하여 규칙 변수의 정의입니다. 자세한 내용은 124 페이지의 『variable 요소』의 내용을 참조하십시오.

이벤트 선택

규칙이 처리하기 위해 어떤 이벤트를 승인할지를 결정하는 기준의 정의입니다. 자세한 내용은 96 페이지의 『eventSelector 요소』의 내용을 참조하십시오.

그룹화 키

규칙에게 공통 특성을 공유하는 각 이벤트 그룹마다 별도의 규칙 인스턴스(또는 사본)를 작성하도록 지시하는 방법인 그룹화 키의 정의입니다. 자세한 내용은 100 페이지의 『groupingKey 요소』의 내용을 참조하십시오.

패턴 특성

상태 규칙이 패턴을 일치시키기 위해 처리 중인 기간의 스펙 및 특정 상태 규칙 패턴의 고유 측면에 대한 정의입니다. 자세한 내용은 124 페이지의 『timeWindow 요소』의 내용을 참조하십시오.

계산 규칙의 경우 여기에는 수집된 이벤트에 적용할 계산의 정의가 포함됩니다. 자세한 내용은 88 페이지의 『computeFunction 요소』의 내용을 참조하십시오.

임계값 규칙의 경우 여기에는 임계값 유형의 정의 및 임계값 유형에 특정적인 다른 정보가 포함됩니다. 자세한 내용은 다음 주제를 참조하십시오.

- 81 페이지의 『booleanThreshold 요소』
- 85 페이지의 『computedThreshold 요소』
- 93 페이지의 『eventCountThreshold 요소』

규칙 응답

규칙이 처리를 완료할 때 취할 조치의 정의입니다.

자세한 내용은 다음 주제를 참조하십시오.

- 중복, 필터, 연속 또는 임계값 규칙의 경우: 107 페이지의 『onDetection 요소』
- 중복 규칙의 경우: 108 페이지의 『onNextEvent 요소』
- 연속 또는 임계값 규칙의 경우: 109 페이지의 『onTimeOut 요소』
- 콜렉션, 계산, 중복 또는 타이머 규칙의 경우: 110 페이지의 『onTimeWindowComplete 요소』

활성화 간격

규칙이 활성 및 비활성인 시기의 정의입니다. 자세한 내용은 75 페이지의 『activationInterval 요소』의 내용을 참조하십시오.

수명 주기

규칙의 수명 주기에서 네 개의 기본 단계인 로드, 활성화, 비활성화 및 로

드 해제에 취할 조치의 정의입니다. 일반적으로 이러한 조치는 정의할 필요가 없습니다. 자세한 내용은 104 페이지의 『lifeCycleActions 요소』의 내용을 참조하십시오.

4. 규칙 표현식 내에서 호출할 Java 메소드 및 연관된 조각을 식별하십시오. 규칙 작성자는 규칙 표현식 내에 확장 Java 코드를 작성하는 것보다 Java 메소드를 사용하여 외부 모듈로 호출할 것을 권장합니다. ACT(Active Correlation Technology)를 임베드하는 어플리케이션이 이러한 외부 모듈을 제공할 수 있으며, 필요한 경우 규칙 작성자도 이를 작성할 수 있습니다. 각 Java 메소드에 연관된 조각도 식별해야 합니다. 자세한 정보는 25 페이지의 『표현식 코딩을 위한 Best Practices』의 내용을 참조하십시오.

『이벤트를 연관시키기 위한 규칙 설계』(으)로 진행하십시오.

이벤트를 연관시키기 위한 규칙 설계

이벤트 연관을 계획한 후에는 이벤트를 연관시키기 위한 규칙을 설계해야 합니다.

먼저 39 페이지의 『이벤트 연관 계획』을(를) 완료하십시오.

규칙을 설계하려면 다음 단계를 수행하십시오.

1. 규칙 세트 내에서 규칙 및 규칙 블록의 조직을 설계합니다.
2. 서로 다른 변수를 정의할 레벨을 설계하십시오(예: 규칙 세트, 규칙 블록 또는 규칙의 레벨).
3. 규칙 패턴을 기반으로 하여 각 규칙의 요소를 설계하십시오.

   이 단계는 계획 단계에서 식별한 각 규칙 패턴의 구조를 사용합니다.
4. 특히 이벤트의 전달 및 중복 이벤트의 경우 규칙 세트 처리 건너뛰기 등과 관련하여 규칙 간의 상호작용을 설계하십시오.

   자세한 내용은 36 페이지의 『규칙 세트를 통한 이벤트 플로우』의 내용을 참조하십시오.
5. 표현식 내에서 호출하기 위해 작성한 모든 Java 메소드 및 연관된 조각(snippet)을 설계, 개발 및 테스트하십시오.

43 페이지의 『규칙 빌더 시작하기』(으)로 진행하십시오.

규칙 빌더 시작하기

이벤트를 연관시키기 위해 규칙을 설계한 후에는 ACT(Active Correlation Technology) 규칙 빌더를 사용하여 규칙을 작성할 수 있습니다.

다음 단계는 규칙 빌더를 사용하여 규칙을 작성하는 기본 단계입니다.

1. Eclipse Workbench를 여십시오.
2. Eclipse Workbench에 perspective를 설정하십시오.
3. ACT(Active Correlation Technology)의 환경 설정을 입력하거나 기본 환경 설정을 승인하십시오.
4. 규칙 세트 파일을 저장할 프로젝트를 새로 작성하거나 기존의 프로젝트를 사용하십시오.
5. 규칙 세트 파일을 작성하고 이를 원하는 프로젝트에 저장하십시오.
6. 규칙 세트 내에 하나 이상의 규칙 블록을 작성하십시오. 또는 규칙 세트 내에서 다른 규칙 블록을 작성하고 규칙 블록 내에 규칙 블록을 작성할 수도 있습니다.
7. 규칙 블록 내에 규칙을 작성하십시오.
8. 결과 세트의 유효성을 검증하십시오.
9. 규칙 세트를 컴파일하십시오.
10. 필요에 따라 규칙 세트를 갱신하십시오.

규칙 내의 표현식에 Eclipse Workbench의 조각 보기에 있는 조각을 포함시킬 수도 있습니다.

Eclipse Workbench에 perspective 설정

다른 작업을 수행하기 전에 Eclipse Workbench에 perspective를 설정해야 합니다. 규칙 빌더 perspective를 여는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오.

규칙 빌더 perspective 대신 다른 perspective를 사용할 수는 있으나 여러 태스크를 수행하는 단계는 선택한 perspective에 따라 약간씩 다릅니다.

규칙 빌더 perspective를 열려면 다음 단계를 따르십시오.

1. 워크벤치에서 **창** → **Perspective 열기** → **기타...**를 누르십시오.
2. **규칙 빌더**를 누르고 **확인**을 누르십시오. 그러면 규칙 빌더 perspective가 표시됩니다.

44 페이지의 『환경 설정』(으)로 진행하십시오.

환경 설정

규칙 세트 파일을 작성하기 전에 Eclipse Workbench에 ACT(Active Correlation Technology)의 환경 설정이 올바르게 설정되었는지 확인해야 합니다. 해당 환경 설정을 설정하는 방법은 다음과 같습니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오.

ACT(he Active Correlation Technology)의 환경 설정을 설정하려면 다음 단계를 따르십시오.

1. 워크벤치에서 **창 → 환경 설정...**을 누르십시오.
2. **ACT(Active Correlation Technology)**를 누르고 ACT(Active Correlation Technology) 페이지에서 규칙 빌더에 새로 작성된 규칙 및 규칙 블록에 대해 접두부로서 『act』를 추가할지 여부를 지정하십시오. 기본적으로 『act』는 접두부로 추가되지 *않습니다*.
3. **ACT(Active Correlation Technology)**를 펼치십시오. 어플리케이션에 따라 다음과 같은 추가 항목이 표시될 수 있습니다. 연관된 환경 설정을 설정하려면 다음 항목을 누르십시오.

항목	연관된 환경 설정
CBE(Common Base Event) 정의 제공자	해당 이벤트 유형이 포함할 수 있는 속성의 이름과 유형을 포함하여 CBE(Common Base Event) 스펙을 준수하는 하나 이상의 이벤트 유형의 구조를 제공하는 XML 파일을 지정할 수 있습니다. 규칙을 작성하면 이러한 이벤트 유형 및 속성이 사용 가능합니다.
컴파일러	다음과 같은 기본 항목을 지정할 수 있습니다. • 컴파일된 규칙 세트를 저장할 수 있는지 여부 • 컴파일된 규칙 세트의 파일 유형 • 컴파일 시 사용할 코드의 클래스 경로 기본적으로, 컴파일된 규칙 세트는 이것이 직렬화된 규칙 세트임을 나타내는 파일 유형 `acts`와 함께 저장된 후 검색기 보기에 표시됩니다.

『규칙 세트 파일을 저장하기 위한 프로젝트 작성』(으)로 진행하십시오.

규칙 세트 파일을 저장하기 위한 프로젝트 작성

Eclipse Workbench에서 규칙 세트 파일을 작성할 때에는 파일을 저장할 프로젝트를 지정해야 합니다. 그러므로 규칙 세트 파일을 작성하기 전에 프로젝트를 작성하거나 기존 프로젝트를 사용해야 합니다. Eclipse Workbench에서 프로젝트를 작성하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

워크벤치 내에서 단순 프로젝트를 작성하려면 다음 단계를 수행하십시오.

1. **파일 → 새로 작성 → 프로젝트...**를 누르십시오.
2. 새 프로젝트 마법사에서 **단순 → 프로젝트**를 누르고 **다음**을 누르십시오.
3. **프로젝트 이름** 필드에 프로젝트의 고유 이름을 입력하십시오. 또한 프로젝트의 기본 위치를 사용하거나 다른 위치를 선택하십시오.
4. **완료**를 누르십시오. 그러면 새 프로젝트가 규칙 빌더 perspective의 검색기 보기에 표시됩니다.

『규칙 세트 작성』(으)로 진행하십시오.

규칙 세트 작성

규칙 세트를 작성하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

규칙 세트 파일을 작성하려면 다음 단계를 수행하십시오.

1. **파일 → 새로 작성 → 규칙 세트 파일**을 누르십시오.
2. 새 규칙 세트 파일 마법사에서 규칙 세트 파일을 저장할 프로젝트와 연관된 폴더의 이름을 누르십시오. 이 프로젝트는 44 페이지의 『규칙 세트 파일을 저장하기 위한 프로젝트 작성』에서 작성한 프로젝트일 수 있습니다. 그러면 폴더 이름이 첫 번째 필드에 표시됩니다.
3. **파일 이름** 필드에서 규칙 세트 파일의 이름을 입력하십시오. 파일 유형은 actl이어야 합니다.
4. **다음**을 누르십시오.
5. 규칙 세트의 고유 이름 및 선택적 설명을 입력하십시오. **XML 인코딩** 필드의 기본값을 사용하려면 XML 파일인 규칙 세트 파일의 인코딩 양식 또한 지정하십시오.
6. **완료**를 누르십시오. 그러면 검색기 보기의 해당 프로젝트 폴더에 규칙 세트 파일이 표시됩니다. 파일을 저장할 때 유효성이 검증되므로 파일 옆에 『유효성이 검증됨』이라고 표시됩니다. 개요 보기에도 규칙 세트 파일이 표시됩니다.

『규칙 블록 작성』(으)로 진행하십시오.

규칙 블록 작성

규칙 세트 또는 다른 규칙 블록 내에 규칙 블록을 작성하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

규칙 세트를 처음으로 작성하는 경우에는 규칙을 작성하기 전에 먼저 규칙 세트 내에서 하나 이상의 규칙 블록을 작성해야 합니다. 첫 번째 규칙 블록을 작성한 후에는 규칙 블록 내의 규칙 블록을 포함하여 추가로 규칙 블록을 작성할 수 있습니다.

규칙 블록을 작성하려면 다음 단계를 수행하십시오.

1. 검색기 보기에서 갱신할 규칙 세트 파일의 이름을 두 번 누르십시오. 그러면 파일이 개요 보기에서 열립니다. 개요 보기에서 규칙 세트를 누르면 편집기 영역에 규칙 세트의 현재 정보가 표시됩니다.
2. 개요 보기에서 규칙 세트를 마우스 오른쪽 단추로 누르십시오.
3. **새 하위 항목 → 규칙 블록**을 누르십시오. 그러면 규칙 블록이 개요 보기에서 규칙 세트 내에 표시되고, 규칙 블록의 현재 정보가 편집기 영역에 표시됩니다.

다음과 같은 방법으로 추가로 규칙 블록을 지금 작성할 수 있습니다.

- 기존 규칙 블록과 동등한 규칙 블록을 작성하려면 기존 규칙 블록을 마우스 오른쪽 단추로 누르고 **새 동등 항목 → 규칙 블록**을 누르십시오.
- 기존의 규칙 블록 내에 규칙 블록을 작성하려면 기존의 규칙 블록을 마우스 오른쪽 단추로 누른 다음 **새 하위 항목 → 규칙 블록**을 누르십시오.

또한 편집기를 사용하면 규칙 세트 및 각 규칙 블록의 정보를 갱신할 수도 있습니다. 『규칙 작성』(으)로 진행하십시오.

규칙 작성

규칙을 작성하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

규칙은 규칙 블록 내에서 작성해야 합니다. 규칙을 작성하려면 다음 단계를 수행하십시오.

1. 개요 보기에서 마우스 오른쪽 단추로 갱신할 규칙 블록을 누르십시오.
2. **새 하위 항목**을 누르고 작성할 규칙 유형을 누르십시오. 그러면 규칙이 개요 보기의 규칙 블록 내에 표시되고 규칙의 현재 정보가 편집기 영역에 표시됩니다.

동일한 방법으로 규칙 블록에 추가 규칙을 추가할 수 있습니다. 또한 편집기를 사용하여 각 규칙의 정보를 갱신할 수도 있습니다. 47 페이지의 『규칙 세트 유효성 검증』(으)로 진행하십시오.

규칙 세트 유효성 검증

규칙 세트를 컴파일하기 전에 규칙 세트의 유효성을 검증하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

규칙 세트의 유효성을 검증하려면 다음 단계를 따르십시오.

1. 개요 보기에서 규칙, 규칙 블록 또는 규칙 세트를 마우스 오른쪽 단추로 누르십시오.
2. **규칙 세트 유효성 검증**을 누르십시오. 유효성 검증이 완료되면 문제점이 있는지 여부를 표시하는 메시지 창이 표시됩니다. 유효성 검증이 완료되면 검색기 보기에 있는 규칙 세트의 파일 이름 오른쪽에 『유효성이 검증됨』이라고 표시됩니다.

유효성 검증이 완료되면 『규칙 세트 컴파일』(으)로 진행하십시오.

규칙 세트 컴파일

규칙 세트를 컴파일하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

검색기 또는 개요 보기에서 컴파일할 규칙 세트를 마우스 오른쪽 단추로 누른 다음 **규칙 세트 컴파일**을 누르십시오. 컴파일 오류가 문제점 보기에 표시됩니다.

기본적으로 컴파일 오류가 없으면 컴파일된 규칙 세트는 이것이 직렬화된 규칙 세트임을 나타내는 기본 파일 유형 `acts`와 함께 검색기 보기에 표시됩니다. 검색기 보기에서 규칙 세트의 파일 이름 오른쪽에 『컴파일됨』 이라고 표시됩니다.

규칙 세트 갱신

규칙 세트를 갱신하는 방법에 대해 설명합니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

개요 보기에서 갱신할 항목(규칙, 규칙 블록 또는 규칙 세트)을 누르십시오. 그러면 항목의 현재 정보가 편집기 영역에 표시되고 편집기를 사용하여 이 정보를 갱신할 수 있습니다.

기존 규칙 블록이나 규칙과 동등한 규칙 블록이나 규칙을 작성하려면 다음 단계를 수행하십시오.

1. 기존 규칙 블록이나 규칙을 마우스 오른쪽 단추로 누르십시오.
2. **새 동등 항목** 및 추가할 항목(규칙 블록 또는 규칙 유형)을 누르십시오.

기존 규칙 블록 내에서 규칙 블록이나 규칙을 작성하려면 다음 단계를 수행하십시오.

1. 기존 규칙 블록을 마우스 오른쪽 단추로 누르십시오.
2. **새 하위 항목** 및 추가할 항목(규칙 블록 또는 규칙 유형)을 누르십시오.

개요 보기에서 다른 갱신 기능에 액세스하려면 다음 단계를 수행하십시오.

1. 갱신할 항목(규칙 블록 또는 규칙)을 마우스 오른쪽 단추로 누르십시오.
2. 기능의 메뉴 항목(예: **삭제**, **복사** 또는 **붙여넣기**)을 누르십시오.

규칙의 표현식에 조각 포함

규칙 내의 표현식에 Eclipse Workbench의 조각 보기에 있는 조각을 포함시킬 수 있습니다.

먼저 Eclipse Workbench가 아직 열려 있지 않은 경우 이를 여십시오. 그리고 규칙 빌더 perspective도 여십시오.

조각 보기는 규칙 빌더 perspective에 표시됩니다. 조각은 기능에 따라 여러 범주로 구성됩니다.

조각 보기에 있는 조각을 포함시키려면 다음 단계를 따르십시오.

1. 범주에 있는 조각 이름을 보려면 조각 범주를 누르십시오.
2. 표현식에 포함시킬 조각을 누르십시오.
3. 조각을 해당 **표현식** 필드로 끄십시오. 코드가 **표현식** 필드의 커서 위치에 배치됩니다. 코드에서 규칙 작성자로부터 특정 메시지 텍스트 또는 변수 값과 같은 입력을 받아야 하는 경우 코드가 표현식에 포함되기 전에 입력을 제공하도록 프롬프트됩니다.

47 페이지의 『규칙 세트 유효성 검증』(으)로 진행하십시오.

제 2 부 규칙 작성자의 참조서

제 4 장 규칙 세트 조직 요약

이 참조에서는 규칙 세트, 규칙 블록 및 각 규칙 유형의 언어 요소를 모두 나열합니다. 이 참조는 규칙 세트를 코딩하기 위한 빠른 참조용으로도 사용할 수 있습니다.

표 10에서는 각 언어 요소를 뒤따르는 표기의 의미를 설명합니다. *n*은 무제한의 숫자를 표시합니다.

표 10. 언어 요소의 발생 개수를 정의하는 표기의 설명

표기	의미
(0, 1)	0은 언어 요소가 선택적임을 표시합니다. 1은 코딩된 경우 1 발생만이 허용됨을 의미합니다.
(0, *n*)	0은 언어 요소가 선택적임을 표시합니다. *n*은 코딩된 경우 여러 개의 발생이 허용됨을 표시합니다.
(1, 1)	첫 번째 1은 언어 요소가 필수임을 표시합니다. 두 번째 1은 1 발생만이 허용됨을 표시합니다.
(1, *n*)	1은 언어 요소가 필수임을 표시합니다. *n*은 여러 개의 발생이 허용됨을 표시합니다.
(2, *n*)	2는 언어 요소의 두 개의 발생이 필수임을 표시합니다. *n*은 추가적인 발생이 허용됨을 표시합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

규칙 세트 요약

이 요약은 규칙 세트의 언어 요소를 나열합니다.

규칙 세트 요소

<ruleSet>는 다음 요소를 포함합니다.

- <comment> (0, 1)
- <import> (0, n)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <ruleBlock> (0, n)

  관련 참조

  52 페이지의『규칙 블록 요약』
  이 요약은 규칙 블록의 언어 요소를 나열합니다.

규칙 블록 요약

이 요약은 규칙 블록의 언어 요소를 나열합니다.

규칙 블록 요소

<ruleBlock>은 다음 요소를 포함합니다.

<comment>, <import> 및 <variable> 요소는 코딩된 경우 표시된 순서대로 코딩되어야 합니다. 나머지 요소는 순서와 무관하게 코딩될 수 있습니다.

- <comment> (0, 1)
- <import> (0, n)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <ruleBlock> (0, n)
- <collectionRule> (0, n)
- <computationRule> (0, n)
- <duplicateRule> (0, n)
- <filterRule> (0, n)
- <sequenceRule> (0, n)
- <thresholdRule> (0, n)
- <timerRule> (0, n)

관련 참조

53 페이지의『콜렉션 규칙 요약』
이 요약에는 콜렉션 규칙의 모든 언어 요소가 나열되어 있습니다.

54 페이지의『계산 규칙 요약』
이 요약에는 계산 규칙의 모든 언어 요소가 나열되어 있습니다.

56 페이지의『중복 규칙 요약』
이 요약에는 중복 규칙의 모든 언어 요소가 나열되어 있습니다.

58 페이지의『필터 규칙 요약』
이 요약에는 필터 규칙의 모든 언어 요소가 나열되어 있습니다..

59 페이지의『연속 규칙 요약』
이 요약에는 연속 규칙의 모든 언어 요소가 나열되어 있습니다.

61 페이지의『임계값 규칙 요약』
이 요약에는 임계값 규칙의 모든 언어 요소가 나열되어 있습니다.

63 페이지의『타이머 규칙 요약』
이 요약에는 타이머 규칙의 모든 언어 요소가 나열되어 있습니다.

콜렉션 규칙 요약

이 요약에는 콜렉션 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<collectionRule>에는 다음 요소가 포함되어 있습니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <activationByGroupingKey> (0, 1)
    - <activateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)

        - <stopAfter> (0, 1)
      - <deactivateOnEvent> (0, 1)
        - <eventType> (0, n)
        - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)
  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)
    - <action> (0, n)
- <eventSelector> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <groupingKey> (0, 1)
  - 순서에 관계 없이 다음 세 가지 요소 중 하나입니다(1, n).
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

계산 규칙 요약

이 요약에는 계산 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<computationRule>에는 다음 요소가 포함되어 있습니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <activationByGroupingKey> (0, 1)
    - <activateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
      - <stopAfter> (0, 1)
    - <deactivateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)

  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)
    - <action> (0, n)
- <eventSelector> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <groupingKey> (0, 1)
  - 순서에 관계 없이 다음 세 가지 요소 중 하나입니다(1, n).
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <computeFunction> (1, 1)
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

중복 규칙 요약

이 요약에는 중복 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<duplicateRule>에는 다음 요소가 포함되어 있습니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)

- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <activationByGroupingKey> (0, 1)
    - <activateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
      - <stopAfter> (0, 1)
    - <deactivateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)
  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)

- <action> (0, n)

- <eventSelector> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <groupingKey> (0, 1)
  - 순서에 관계 없이 다음 세 가지 요소 중 하나입니다(1, n).
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onNextEvent> (0, 1)
  - <action> (0, n)
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

필터 규칙 요약

이 요약에는 필터 규칙의 모든 언어 요소가 나열되어 있습니다..

규칙 요소

<filterRule>에는 다음 요소가 포함되어 있습니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).

        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)
  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)
    - <action> (0, n)
- <eventSelector> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <onDetection> (0, 1)
  - <action> (0, n)

연속 규칙 요약

이 요약에는 연속 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<sequenceRule>은 다음 요소를 포함합니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <activationByGroupingKey> (0, 1)
    - <activateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
      - <stopAfter> (0, 1)
    - <deactivateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)

  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)
    - <action> (0, n)
- <eventSelector> (2, n)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <groupingKey> (0, 1)
  - 순서에 관계 없이 다음 세 가지 요소 중 하나입니다(1, n).
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

임계값 규칙 요약

이 요약에는 임계값 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<thresholdRule>은 다음 요소를 포함합니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)

  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 다음 세 가지 요소 중 하나입니다(1, 1).
        - <dateTime>
        - <never>
        - <after>
  - <activateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <deactivateOnEvent> (0, 1)
    - <eventType> (0, n)
    - <filteringPredicate> (0, 1)
  - <activationByGroupingKey> (0, 1)
    - <activateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
      - <stopAfter> (0, 1)
    - <deactivateOnEvent> (0, 1)
      - <eventType> (0, n)
      - <filteringPredicate> (0, 1)
- <lifeCycleActions> (0, 1)
  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)

  - <onUnload> (0, 1)
    - <action> (0, n)
- <eventSelector> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <groupingKey> (0, 1)
  - 순서에 관계 없이 다음 세 가지 요소 중 하나입니다(1, n).
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- 다음 세 가지 요소 중 하나입니다(1, 1).
  - <booleanThreshold>
  - <computedThreshold>
  - <eventCountThreshold>
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

타이머 규칙 요약

이 요약에는 타이머 규칙의 모든 언어 요소가 나열되어 있습니다.

규칙 요소

<timerRule>은 다음 요소를 포함합니다.

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)

- <activationTime> (0, 1)
  - <start> (0, 1)
    - 다음 세 가지 요소 중 하나입니다(1, 1).
      - <dateTime>
      - <whenLoaded>
      - <inactiveWhenLoaded>
  - <stop> (0, 1)
    - 다음 세 가지 요소 중 하나입니다(1, 1).
      - <dateTime>
      - <never>
      - <after>
- <activateOnEvent> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)
- <deactivateOnEvent> (0, 1)
  - <eventType> (0, n)
  - <filteringPredicate> (0, 1)

- <lifeCycleActions> (0, 1)
  - <onLoad> (0, 1)
    - <action> (0, n)
  - <onActivation> (0, 1)
    - <action> (0, n)
  - <onDeactivation> (0, 1)
    - <action> (0, n)
  - <onUnload> (0, 1)
    - <action> (0, n)
- <timeWindow> (1, 1)
  - 다음 두 가지 요소 중 하나입니다(1, 1).
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

제 5 장 언어 요소 참조

이 참조는 ACT(Active Correlation Technology) 규칙 언어의 XML 스키마로 된 언어 요소의 세부사항을 설명합니다. 언어 요소는 알파벳 순서로 나열되어 있으며 각 요소에서 사용 가능한 속성은 해당 요소의 주제 내에 설명되어 있습니다.

XML 및 다른 마크업 언어(예: SGML 및 HTML)에서 요소는 시작 태그, 종료 태그, 연관된 속성, 해당 값 및 시작과 종료 태그 사이에 포함된 모든 텍스트로 구성된 기본 단위입니다. 속성은 요소의 특정 특성을 정의하기 위해 요소에서 코딩된 이름-값 쌍입니다. 속성에는 값으로 제공되는 정보의 유형(예: 숫자, 텍스트 또는 부울 정보)을 식별하는 데이터 유형이 있습니다.

XML에서 이름 공간은 스키마에서 요소와 유형 정의를 연관시키기 위한 고유한 이름을 제공하는 URI(Uniform Resource Identifier)입니다. URI는 어떤 XML 스키마에 요소의 정의가 포함되어 있는지를 표시합니다. 이름 공간은 콜론이 뒤따르는 접두부 문자열로 지정되어 있습니다. ACT(Active Correlation Technology) 규칙 언어 스키마는 세 개의 서로 다른 파일에 정의되어 있으며 다음과 같은 세 개의 이름 공간을 사용합니다.

xsd: 이 이름 공간은 언어 요소가 http://www.w3.org에 설명되어 있는 표준 XML 스키마에 정의되어 있음을 표시합니다.

br: 이 이름 공간은 언어 요소가 com/ibm/correlation/ruleparser/xml/RuleSetBase.xsd 서브디렉토리의 ACTparser.jar 파일에 있는 ACT(Active Correlation Technology) 기반 규칙 세트 스키마에 정의되어 있음을 표시합니다. 예를 들어, `br:ruleSet`는 RuleSetBase.xsd 파일에 정의되어 있는 ruleSet 요소를 가리킵니다.

act: 이 이름 공간은 com/ibm/correlation/ruleparser/xml/ACTL.xsd 서브디렉토리의 ACTparser.jar 파일에 있는 ACT(Active Correlation Technology) 언어 스키마에 정의되어 있는 언어 요소를 표시합니다. 예를 들어, `act:ruleSet`는 ACTL.xsd 파일에 정의되어 있는 ruleSet 요소를 가리킵니다.

규칙 언어 스키마에서 언어 요소는 요소 또는 복합 유형으로 정의되어 있습니다. 예:

```
<xsd:element name="symbol" minOccurs="1" maxOccurs="unbounded"></element>
<xsd:complexType name="symbol"></complexType>
```

스키마에서, minOccurs 및 maxOccurs 속성은 각각 언어 요소의 최소 및 최대 발생 횟수를 정의합니다. 66 페이지의 표 11에 minOccurs 및 maxOccurs 속성의 서로 다른 값의 의미가 설명되어 있습니다.

표 11. 언어 요소의 발생 횟수를 정의하는 스키마의 속성

속성	속성값	의미
minOccurs	0	언어 요소는 선택적입니다.
minOccurs	1	언어 요소는 한 번 이상 발생해야 합니다. 1은 minOccurs 속성의 기본값입니다.
minOccurs	2	언어 요소는 두 번 이상 발생해야 합니다.
maxOccurs	1	언어 요소는 두 번 이상 발생할 수 없습니다. 1은 maxOccurs 속성의 기본값입니다.
maxOccurs	unbounded	언어 요소는 얼마든지 여러 번 발생할 수 있습니다.

조치 요소

<action> 요소에는 규칙 응답 조치 또는 수명 주기 조치를 정의하는 표현식이 포함되어 있습니다.

세부사항

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의 『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<action>에는 다음과 같은 속성이 있습니다.

표 12. <action> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예
이름	조치를 식별합니다. 이 ID는 문제점 해결용으로 유용합니다. 특히 특정 규칙 응답이나 수명 주기 조치에 정의된 모든 <action> 요소 사이에서 고유한 이름인 경우에 더욱 유용합니다.	xsd:NMTOKEN	아니오

포함 위치

<action>은 다음 요소 내에 포함되어 있습니다.

- <onActivation>
- <onDeactivation>
- <onDetection>
- <onLoad>
- <onNextEvent>

- <onTimeOut>
- <onTimeWindowComplete>
- <onUnload>

포함 요소

<action>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

activateOnEvent 요소

<activateOnEvent> 요소는 규칙을 활성화할 수 있거나 <groupingKey> 요소로 정의된 규칙의 경우 규칙 인스턴스를 활성화할 수 있는 이벤트를 정의합니다.

다음은 이벤트를 선택하는 세 가지 가능한 방법입니다.

- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소와 함께 사용
- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소 없이 사용
- <filteringPredicate> 요소를 <eventType> 요소 없이 사용

규칙이 비활성이고 <eventType> 또는 <filteringPredicate> 요소가 코딩되지 않은 경우에는 발생하는 모든 이벤트가 선택됩니다.

시스템 성능에 부정적인 영향을 미칠 수 있는 <eventType> 요소를 코딩하지 마십시오.

`Audit Failure` 유형의 모든 이벤트를 선택하고 싶어한다고 가정합니다. 필터링 선언문을 사용하면 특정값의 이벤트 속성이 있는 이벤트만을 포함하도록 선택 기준을 추가로 구체화할 수 있습니다. 예를 들어, `Audit Failure` 유형의 모든 이벤트를 선택하기 위해 <eventType> 요소를 코딩하고 `MyCriticalSystem` 값이 있는 호스트 이름 속성이 있는 이벤트만을 선택하도록 <filteringPredicate> 요소를 코딩할 수 있습니다.

속성

<activateOnEvent>에는 속성이 없습니다.

포함 위치

<activateOnEvent>는 다음 요소 내에 포함되어 있습니다.

- <activationInterval>

- <activationByGroupingKey>

포함 요소

<activateOnEvent>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 13. <activateOnEvent> 요소에 포함된 요소

요소	필수 또는 선택적 여부
<eventType>	선택적. 0 이상의 발생이 허용됩니다.
<filteringPredicate>	선택적. 0 또는 1 발생이 허용됩니다.
<stopAfter>	이 요소는 <activateOnEvent> 요소가 <activationByGroupingKey> 요소에 포함된 경우에만 유효합니다. 선택적. 0 또는 1 발생이 허용됩니다.

activationByGroupingKey 요소

<activationByGroupingKey> 요소에는 <groupingKey> 요소로 정의되는 규칙 인스턴스를 활성화하고 비활성화할 수 있는 이벤트를 지정하는 요소가 포함되어 있습니다. <groupingKey> 요소는 필터 및 타이머 규칙의 경우에는 사용할 수 없으므로 <activationByGroupingKey> 요소는 이러한 규칙에는 적용되지 않습니다.

세부사항

<activationByGroupingKey> 요소가 제공하는 함수는 그룹화 키를 정의하는 규칙에서 사용하기 위한 것입니다. 이를 사용하면 그룹화 키를 기반으로 규칙 인스턴스의 활성화 및 비활성화를 제어할 수 있습니다. <activationByGroupingKey> 요소를 코딩할 때 각 규칙 인스턴스는 <activationByGroupingKey> 내의 <activateOnEvent> 및 <deactivateOnEvent> 조건에 따라 개별적으로 활성화하고 비활성화할 수 있습니다.

다음 예제는 계산 규칙 내에서 <activationByGroupingKey> 요소의 사용을 보여줍니다.

- 다음 계산 규칙은 `StockSharesTraded` 유형의 이벤트를 승인합니다. 이러한 이벤트는 여러 개의 다른 회사의 거래 중인 주식 개수를 나타냅니다.
- 그룹화 키는 이벤트의 stockSymbol 속성입니다. stockSymbol 속성의 값은 특정 회사의 이름입니다.
- 이벤트의 sharesTraded 속성 값은 각 회사의 거래 주식 개수입니다(회사는 stockSymbol 속성값으로 표시됩니다).

- 특정 회사의 경우 10분 동안 해당 회사의 거래 주식 수를 나타내는 보고서가 작성됩니다. 그러나 계산 규칙이 이 보고서를 작성하려면 먼저 각 회사의 이름을 stockSymbol 속성값으로 사용하는 StartReporting 유형의 이벤트를 받아야 합니다.

```
<computationRule name="StockReporter">
 <variable dataType="java.lang.Integer" name="totalSharesTraded">
  <varInitializer expressionLanguage="java">
   return new Integer(0);
  </varInitializer>
 </variable>

 <activationInterval>
  <activationTime>
   <start>
    <inactiveWhenLoaded/>
   </start>
  </activationTime>
  <activationByGroupingKey>
   <activateOnEvent>
    <eventType type="StartReporting"/>
   </activateOnEvent>
  </activationByGroupingKey>
 </activationInterval>

 <eventSelector>
  <eventType type="StockSharesTraded"/>
 </eventSelector>

 <groupingKey>
  <attributeName>stockSymbol</attributeName>
 </groupingKey>

 <computeFunction assignTo="totalSharesTraded" expressionLanguage="java">
  return new Integer(act_lib.getIntVariable("totalSharesTraded")
            + act_event.getIntAttribute("sharesTraded"));
 </computeFunction>

 <timeWindow>
  <timeInterval unit="ISO-8601" duration="PT10M"/>
 </timeWindow>

 <onTimeWindowComplete>
  <action expressionLanguage="java">
   StockReport.createReport(act_eventList.get(0).getStringAttribute("stockSymbol"),
                act_lib.getIntVariable("totalSharesTraded"));
  </action>
 </onTimeWindowComplete>
</computationRule>
```

속성

<activationByGroupingKey>에는 속성이 없습니다.

포함 위치

<activationByGroupingKey>는 다음 요소 내에 포함되어 있습니다.

- <activationInterval>

포함 요소

<activationByGroupingKey>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 14. <activationByGroupingKey> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<activateOnEvent>	선택적. 0 또는 1 발생이 허용됩니다.
<deactivateOnEvent>	선택적. 0 또는 1 발생이 허용됩니다.

<activationInterval> 및 <activationByGroupingKey> 요소 간의 관계

<activateOnEvent> 및 <deactivateOnEvent> 요소는 이러한 두 요소 내에 포함되어 있습니다.

- <activationInterval>
- <activationInterval>에 포함되어 있는 <activationByGroupingKey>

규칙 작동은 현재 규칙 활동 및 <activationInterval>과(와) <activationByGroupingKey> 요소 내의 <activateOnEvent>과(와) <deactivateOnEvent> 정의 간의 상호작용에 따라 다를 수 있습니다. 다음 예제는 이러한 정의가 상호작용하는 방법을 보여줍니다.

이 예제에서는 유지보수 모드에 있는 시스템에서는 이벤트를 억제하고, 유지보수 기간 종료 시에 억제된 이벤트 개수의 요약 보고서를 제공하도록 중복 규칙이 정의되어 있습니다.

기본적으로 그룹화 키가 정의되어 있는 규칙을 사용하면 모든 그룹화 키 값을 처리할 수 있습니다. 그러므로 이벤트가 규칙의 이벤트 선택 기준에 부합하면 모든 규칙 인스턴스가 활성이 되고 그룹화 키의 값에 따라 이러한 이벤트를 승인할 준비가 됩니다. 규칙의 활성화 간격은 규칙에 그룹화 키가 없는 것과 마찬가지입니다. 왜냐하면 기본적으로 규칙의 이벤트 선택 기준에 부합하는 모든 이벤트가 처리되기 때문입니다.

다음 예제에서 그룹화 키는 `hostname`이고 <activationInterval> 요소 내의 정의는 다음 조치를 지정합니다.

1. `StartMaintenanceModeAllHosts` 이벤트 유형을 받을 때 모든 규칙 인스턴스를 활성화합니다.

2. 2시간 후나 StopMaintenanceModeAllHosts 이벤트 유형을 받을 때 모든 규칙을 비활성화합니다.

```
<duplicateRule name="Maintenance_Supression">
  <activationInterval>
    <activationTime>
      <start>
        <inactiveWhenLoaded/>
      </start>
      <stop>
        <after duration="PT2H" unit="ISO-8601"/>
      </stop>
    </activationTime>
    <activateOnEvent>
      <eventType type="StartMaintenanceModeAllHosts"/>
    </activateOnEvent>
    <deactivateOnEvent>
      <eventType type="StopMaintenanceModeAllHosts"/>
    </deactivateOnEvent>
  </activationInterval>
  <groupingKey missingAttributeHandling="ignoreEvent">
    <attributeName>hostname</attributeName>
  </groupingKey>
  <timeWindow>
    <runUntilDeactivated/>
  </timeWindow>
  <onDetection>
    <action expressionLanguage="java" name="DropEvent">
      <![CDATA[act_lib.exitRuleSet();]]>
    </action>
  </onDetection>
  <onTimeWindowComplete>
    <action expressionLanguage="java" name="CreateSummaryOfSupressedEvents">
      <![CDATA[Helper.createSummaryEvent("MaintenanceSummary", act_eventList, act_lib);]]>
    </action>
  </onTimeWindowComplete>
</duplicateRule>
```

어떤 경우에는 어떤 규칙 인스턴스를 활성화할지와 이를 활성화하는 시기를 제어하고 싶을 수 있습니다. 이런 경우 <activationByGroupingKey> 요소를 코딩해야 합니다.

다음 예제는 이전의 예제를 확장하고 그룹화 키 값을 사용하여 어떤 규칙 인스턴스의 처리를 허용할지를 선택하는 방법을 보여줍니다. <activationByGroupingKey> 요소 내의 정의는 다음 조치를 지정할 수 있습니다.

1. 특정 호스트 이름의 규칙 인스턴스가 이러한 호스트 이름에 대한 StartMaintenanceMode 이벤트 유형을 받을 때 처리할 수 있도록 허용합니다.
2. 활성화 2시간 후나 개별 호스트 이름에 대한 StopMaintenanceMode 이벤트 유형을 받을 때 이러한 규칙 인스턴스를 비활성화합니다.

```
<activationByGroupingKey>
  <activateOnEvent>
    <eventType type="StartMaintenanceMode"/>
    <stopAfter duration="PT2H" unit="ISO-8601"/>
  </activateOnEvent>
  <deactivateOnEvent>
    <eventType type="StopMaintenanceMode"/>
  </deactivateOnEvent>
  </activationByGroupingKey>
```

다음 명령문은 <activationByGroupingKey> 요소를 코딩할 때 발생하는 일을 요약합니다.

- <activateOnEvent> 요소가 <activationByGroupingKey> 요소 내에서 코딩되면 <activationByGroupingKey> <activateOnEvent> 조건에 부합하는 이벤트의 그룹화 키 값을 공유하는 이벤트만을 처리할 수 있습니다.
- <deactivateOnEvent> 요소가 <activationByGroupingKey> 요소 내에서 코딩되면 <activationByGroupingKey> <deactivateOnEvent> 조건에 부합하는 이벤트의 그룹화 키 값을 공유하는 이벤트는 처리가 허용되지 *않습니다*.

활성화 및 비활성화 정의가 규칙 상태에 미치는 서로 다른 영향

표 15 및 74 페이지의 표 16에서는 서로 다른 활성화 및 비활성화 정의에 의해 규칙 상태가 영향을 어떤 받는지를 보여줍니다. 이러한 테이블에서는 다음과 같은 규칙이 사용됩니다.

- *A*는 활성화 이벤트입니다.
- 『*A[x]* 표기에서, 』 *x*는 그룹화 키 값을 표시합니다. 예를 들어, *A[1]*는 그룹화 키 값 1을 사용하는 활성화 이벤트입니다.
- *D*는 비활성화 이벤트입니다.
- 『*D[x]* 표기에서, 』 *x*는 그룹화 키 값을 표시합니다. 예를 들어, *D[1]*는 그룹화 키 값 1을 사용하는 비활성화 이벤트입니다.

표 15. 규칙 상태는 서로 다른 활성화 정의에 따라 변경됨

시작 규칙 상태	규칙 상태는 잠재적으로 다음에 영향 받음	종료 규칙 상태
비활성	<activationInterval> <activationTime> <start> 내에 정의된 시간	1. 규칙이 활성화됩니다. 2. <onActivation> 조치가 실행됩니다. 3. 모든 그룹화 키 값이 허용됩니다.
	activate() 메소드	
	<activationInterval> <activateOnEvent> 내에 정의된 *A* 이벤트	
	<activationByGroupingKey> <activateOnEvent>(<stopAfter> *없이*) 내에 정의된 *A[1]* 이벤트	1. 규칙이 활성화됩니다. 2. <onActivation> 조치가 실행됩니다. 3. 그룹화 키 값 1만이 허용됩니다. 규칙 패턴이 이 규칙 인스턴스와 일치하면 그룹화 키 값 1은 더 이상 허용되지 않습니다.
	<activateOnEvent>(<stopAfter> *와*) 내에 정의된 *A[2]* 이벤트	1. 규칙이 활성화됩니다. 2. <onActivation> 조치가 실행됩니다. 3. 그룹화 키 값 2만이 허용되며 <stopAfter> 요소로 지정된 기간 동안에만 허용됩니다. 이 규칙 인스턴스의 규칙 패턴은 이 기간 동안에 여러 번 일치할 수 있습니다.

표 15. 규칙 상태는 서로 다른 활성화 정의에 따라 변경됨 (계속)

시작 규칙 상태	규칙 상태는 잠재적으로 다음에 영향 받음	종료 규칙 상태
• 활성 • 모든 그룹화 키 값 허용	<activationInterval> <activationTime> <start> 내에 정의된 시간	규칙 상태가 변경되지 않았습니다. 시작 규칙 상태와 같습니다.
	activate() 메소드	
	<activationInterval> <activateOnEvent> 내에 정의된 *A* 이벤트	
	<activationByGroupingKey> <activateOnEvent>(<stopAfter> *없이*) 내에 정의된 *A[1]* 이벤트	
	<activateOnEvent>(<stopAfter>*와*) 내에 정의된 *A[2]* 이벤트	
• 활성 • <activationByGroupingKey> <activateOnEvent> 정의를 기반으로 규칙 인스턴스를 트리거한 그룹화 키 값만 허용	<activationInterval> <activationTime> <start> 내에 정의된 시간	규칙 상태가 변경되지 않았습니다. 시작 규칙 상태와 같습니다.
	activate() 메소드	
	<activationInterval> <activateOnEvent> 내에 정의된 *A* 이벤트	모든 그룹화 키 값이 허용됩니다.
	<activationByGroupingKey> <activateOnEvent>(<stopAfter> *없이*) 내에 정의된 *A[1]* 이벤트	• 이전에 허용된 그룹화 키 값에 추가로 그룹화 키 값 1이 허용됩니다. • 규칙 패턴이 이 규칙 인스턴스와 일치하면 그룹화 키 값 1은 더 이상 허용되지 않습니다.
	<activateOnEvent>(<stopAfter>*와*) 내에 정의된 *A[2]* 이벤트	• 이전에 허용된 그룹화 키 값에 추가로 그룹화 키 값 2가 허용됩니다. • 이 값은 <stopAfter> 요소로 지정된 기간 동안에만 허용됩니다. • 이 규칙 인스턴스의 규칙 패턴은 이 기간 동안에 여러 번 일치할 수 있습니다.
• 활성 • <activationByGroupingKey> <deactivateOnEvent> 정의에 따라 허용되지 않은 값을 제외한 모든 그룹화 키 값 허용	<activationInterval> <activationTime> <start> 내에 정의된 시간	규칙 상태가 변경되지 않았습니다. 시작 규칙 상태와 같습니다.
	activate() 메소드	
	<activationInterval> <activateOnEvent> 내에 정의된 *A* 이벤트	모든 그룹화 키 값이 허용됩니다.
	<activationByGroupingKey> <activateOnEvent>(<stopAfter> *없이*) 내에 정의된 *A[1]* 이벤트	이전에 허용된 그룹화 키 값에 추가로 그룹화 키 값 1이 허용됩니다.
	<activateOnEvent>(<stopAfter> *와*) 내에 정의된 *A[2]* 이벤트	이전에 허용된 그룹화 키 값에 추가로 그룹화 키 값 2가 허용됩니다.

표 16. 규칙 상태는 서로 다른 비활성화 정의에 따라 변경됨

시작 규칙 상태	규칙 상태는 잠재적으로 다음의 영향을 받습니다.	종료 규칙 상태
비활성	<activationInterval> <activationTime> <stop> 내에 정의된 시간	규칙 상태가 변경되지 않았습니다. 시작 규칙 상태와 같습니다.
	deactivate() 메소드	
	<activationInterval> <deactivateOnEvent> 내에 정의된 *D* 이벤트	
	<activationByGroupingKey> <deactivateOnEvent> 내에 정의된 *D[1]* 이벤트	
	<activationByGroupingKey> <activateOnEvent> <stopAfter> 내에 정의된 지속 기간이 *A[2]* 이벤트로 인해 종료됨	
• 활성 • 모든 그룹화 키 값 허용	<activationInterval> <activationTime> <stop> 내에 정의된 시간	1. 모든 규칙 인스턴스가 비활성화됩니다. 2. <onDeactivation> 조치가 실행됩니다. 3. 규칙이 비활성화됩니다.
	deactivate() 메소드	
	<activationInterval> <deactivateOnEvent> 내에 정의된 *D* 이벤트	
	<activationByGroupingKey> <deactivateOnEvent> 내에 정의된 *D[1]* 이벤트	• 그룹화 키 값 1이 더 이상 허용되지 않습니다. • 그룹화 키 값 1이 있는 규칙 인스턴스가 활성이면 이는 비활성화됩니다.
	<activationByGroupingKey> <activateOnEvent> <stopAfter> 내에 정의된 지속 기간이 *A[2]* 이벤트로 인해 종료됨	그룹화 키 값 2가 있는 규칙 인스턴스가 비활성화됩니다.
• 활성 • <activationByGroupingKey> <activateOnEvent> 정의를 기반으로 규칙 인스턴스를 트리거한 그룹화 키 값만 허용	<activationInterval> <activationTime> <stop> 내에 정의된 시간	1. 모든 규칙 인스턴스가 비활성화됩니다. 2. <onDeactivation> 조치가 실행됩니다. 3. 규칙이 비활성화됩니다.
	deactivate() 메소드	
	<activationInterval> <deactivateOnEvent> 내에 정의된 *D* 이벤트	
	<activationByGroupingKey> <deactivateOnEvent> 내에 정의된 *D[1]* 이벤트	• 그룹화 키 값 1이 더 이상 허용되지 않습니다. • 그룹화 키 값 1이 있는 규칙 인스턴스가 활성이면 이는 비활성화됩니다.
	<activationByGroupingKey> <activateOnEvent> <stopAfter> 내에 정의된 지속 기간이 *A[2]* 이벤트로 인해 종료됨	• 그룹화 키 값 2는 더 이상 허용되지 않습니다. • 그룹화 키 값 2가 있는 규칙 인스턴스가 비활성화됩니다.

표 16. 규칙 상태는 서로 다른 비활성화 정의에 따라 변경됨 (계속)

시작 규칙 상태	규칙 상태는 잠재적으로 다음의 영향을 받습니다.	종료 규칙 상태
• 활성 • <activationByGroupingKey> <deactivateOnEvent> 정의에 따라 허용되지 않은 값을 제외한 모든 그룹화 키 값 허용	<activationInterval> <activationTime> <stop> 내에 정의된 시간	1. 모든 규칙 인스턴스가 비활성화됩니다. 2. <onDeactivation> 조치가 실행됩니다. 3. 규칙이 비활성화됩니다.
	deactivate() 메소드	
	<activationInterval> <deactivateOnEvent> 내에 정의된 *D* 이벤트	
	<activationByGroupingKey> <deactivateOnEvent> 내에 정의된 *D[1]* 이벤트	• 그룹화 키 값 1이 더 이상 허용되지 않습니다. • 그룹화 키 값 1이 있는 규칙 인스턴스가 활성이면 이는 비활성화됩니다.
	<activationByGroupingKey> <activateOnEvent> <stopAfter> 내에 정의된 지속 기간이 *A[2]* 이벤트로 인해 종료됨	그룹화 키 값 2가 있는 규칙 인스턴스가 비활성화됩니다.

activationInterval 요소

<activationInterval> 요소에는 규칙이 활성 및 비활성인 시기를 정의하는 요소가 포함되어 있습니다.

세부사항

규칙은 개별 지정 시간에 또는 특정 이벤트에 의해 활성화하거나 비활성화할 수 있습니다.

규칙을 개별 지정 시간에 및 특정 이벤트에 의해 활성화되거나 비활성화되도록 지정한 경우 규칙은 지정 시간 또는 이벤트 수신 중 먼저 발생하는 조건에 의해 활성화되거나 비활성화됩니다. 그러나 이 경우 규칙은 전체 수명 주기 동안 여러 이벤트에 의해 활성화되거나 비활성화될 수 있습니다. 예를 들어 규칙은 이벤트에 의해 활성화되고, 비활성화되고, 정의된 지정 시간에 활성화되고, 다시 비활성화된 다음 다른 이벤트에 의해 활성화될 수도 있습니다.

비즈니스 환경에서는 비즈니스의 주식 거래가 개장되었음을 나타내는 이벤트를 받을 때 규칙을 활성화하고 싶을 수 있습니다. IT 환경에서는 2005년 10월 29일 06:00에 유지보수 창을 시작하고 다음 조건 중 먼저 발생하는 조건에 따라 다음과 같은 시간 중 하나에 이를 종료하고 싶을 수 있습니다.

- 2005년 10월 30일 11:30
- 유지보수 작업이 완료되었음을 나타내는 이벤트를 받을 때

속성

<activationInterval>에는 속성이 없습니다.

포함 위치

<activationInterval>은 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>
- <filterRule>
- <sequenceRule>
- <thresholdRule>
- <timerRule>

포함 요소

<activationInterval>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 17. <activationInterval> 요소에 포함된 요소

요소	필수 또는 선택적 여부
<activationTime>	선택적. 0 또는 1 발생이 허용됩니다.
<activateOnEvent>	선택적. 0 또는 1 발생이 허용됩니다.
<deactivateOnEvent>	선택적. 0 또는 1 발생이 허용됩니다.
<activationByGroupingKey>	선택적. 0 또는 1 발생이 허용됩니다.

포함된 요소 간 관계

<activationTime> 요소에 포함된 <start> 및 <stop> 요소는 규칙을 활성화하고 비활성화하는 정적인 방법입니다. 이러한 요소를 사용하면 규칙을 개별 지정 시간에 활성화하거나 비활성화할 수 있습니다. 이와 반대로 <activateOnEvent> 및 <deactivateOnEvent> 요소는 규칙을 활성화하고 비활성화하는 동적인 방법입니다. 이러한 요소를 사용하면 특정 이벤트가 발생할 때 규칙이 활성화되거나 비활성화됩니다. 예를 들어 규칙은 아직 활성이 아닌 경우 <activateOnEvent> 요소에 정의된 기준에 부합하는 이벤트에 의해 활성화됩니다. 규칙은 아직 비활성이 아닌 경우 <deactivateOnEvent> 요소에 정의된 기준에 부합하는 이벤트에 의해 비활성화됩니다. 그러므로 특정 이벤트는 규칙이 활성화되거나 비활성화되는 시기의 정적 정의를 변경할 수 있습니다.

77 페이지의 표 18에서는 다음 요소가 코딩되어 있을 수 있는 특정 조합을 기준으로 규칙이 활성화되거나 비활성화되는 방법과 시기를 설명합니다.

- <start>

- <stop>
- <activateOnEvent>
- <deactivateOnEvent>

표 18에서, *X*는 규칙을 활성화하는 이벤트의 이름을 표시하고, *Y*는 규칙을 비활성화하는 이벤트의 이름을 표시합니다.

<start> 요소가 아예 코딩되어 있지 않으면, 기본 시작 시간은 <whenLoaded> 요소가 정의한 시간과 같습니다.

<stop> 요소가 아예 코딩되어 있지 않으면, 기본 중지 시간은 <never> 요소가 정의한 시간과 같습니다.

표 18. <activationInterval> 내에 포함된 요소의 여러 다른 조합의 코딩을 기반으로 한 규칙 활동

<activationTime>				
<start>	**<stop>**	**<activateOnEvent>**	**<deactivateOnEvent>**	**규칙 활동**
<whenLoaded>	<never>			규칙은 로드될 때 활성 상태가 되며 ACT(Active Correlation Technology) 엔진이 실행되는 동안 계속 활성 상태로 남아 있습니다.
<whenLoaded>	<never>		*Y*	규칙은 로드될 때 활성화됩니다. *Y* 이벤트는 규칙을 비활성화합니다.
<whenLoaded>	<never>	*X*	*Y*	규칙은 로드될 때 활성화됩니다. *Y* 이벤트는 규칙을 비활성화하고 *X* 이벤트는 이를 다시 활성화합니다. 이러한 비활성화와 재활성화는 여러 번 발생할 수 있습니다.
<whenLoaded>	<after>			규칙은 로드될 때 활성화 되며 지정된 시간 간격 후에 비활성화됩니다.
<whenLoaded>	<dateTime>			규칙은 로드될 때 활성화되며 지정된 날짜와 시간에 비활성화됩니다.
<inactiveWhenLoaded>	<never>	*X*		규칙은 로드될 때 비활성 상태입니다. *X* 이벤트는 규칙을 활성화하고 규칙은 ACT(Active Correlation Technology) 엔진이 실행되는 동안 계속 활성 상태로 남아 있습니다.
<inactiveWhenLoaded>	<never>	*X*	*Y*	규칙은 로드될 때 비활성 상태입니다. *X* 이벤트는 규칙을 활성화하고 *Y* 이벤트는 이를 비활성화합니다. 이 활성화와 비활성화는 여러 번 발생할 수 있습니다.
<dateTime>	<dateTime>			규칙은 지정된 날짜와 시간에 활성화되고 지정된 날짜와 시간에 비활성화됩니다.
<dateTime>	<dateTime>	*X*	*Y*	규칙은 지정된 날짜와 시간에 활성화되고 지정된 날짜와 시간에 비활성화됩니다. *X* 이벤트는 규칙을 활성화하고 *Y* 이벤트는 이를 비활성화합니다. *X* 및 *Y* 이벤트는 규칙을 여러 번 활성화 및 비활성화할 수 있습니다.
<dateTime>	<never>			규칙은 지정된 날짜와 시간에 활성화되고 ACT(Active Correlation Technology) 엔진이 실행되는 동안 계속 활성 상태로 남아 있습니다.
<dateTime>	<never>		*Y*	규칙은 지정된 날짜와 시간에 활성화됩니다. *Y* 이벤트는 규칙을 비활성화합니다.
<dateTime>	<never>	*X*	*Y*	규칙은 지정된 날짜와 시간에 활성화됩니다. *Y* 이벤트는 규칙을 비활성화하고 *X* 이벤트는 이를 다시 활성화합니다. 이러한 비활성화와 재활성화는 여러 번 발생할 수 있습니다.
<dateTime>	<after>			규칙은 지정된 날짜와 시간에 활성화되고 지정된 시간 간격 이후에 비활성화됩니다.
<dateTime>	<after>	*X*	*Y*	규칙은 지정된 날짜와 시간에 활성화되고 지정된 시간 간격 이후에 비활성화됩니다. *X* 이벤트는 규칙을 활성화하고 *Y* 이벤트는 이를 비활성화합니다. 이 활성화와 비활성화는 여러 번 발생할 수 있습니다.

activationTime 요소

<activationTime> 요소는 규칙이 활성화되거나 비활성화되는 개별 지정 시간을 정의합니다.

속성

<activationTime>에는 속성이 없습니다.

포함 위치

<activationTime>은 다음 요소 내에 포함되어 있습니다.

- <activationInterval>

포함 요소

<activationTime>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 19. <activationTime> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<start>	선택적. 0 또는 1 발생이 허용됩니다.
<stop>	선택적. 0 또는 1 발생이 허용됩니다.

after 요소

<after> 요소는 규칙이 활성화된 후에 활성 상태로 남아 있을 지속 기간을 지정합니다. 이 지속 기간 후에는 규칙이 비활성화됩니다.

속성

<after>에는 다음과 같은 속성이 있습니다.

표 20. <after> 요소의 속성

이름	설명	데이터 유형	필수 여부
duration	지속 기간의 총 시간을 지정합니다. 이 속성의 데이터 유형은 단위 속성값에 따라 다릅니다.	• 단위 속성값이 ISO-8601이면 데이터 유형은 xsd:duration입니다. • 단위 속성값이 milliseconds이면 데이터 유형은 xsd:positiveInteger입니다.	예

표 20. <after> 요소의 속성 (계속)

이름	설명	데이터 유형	필수 여부
unit	사용할 시간 단위를 지정합니다. 이 속성의 올바른 값은 다음과 같습니다. • ISO-8601 • milliseconds	xsd:string	예

시간 지속 기간에 ISO 8601 표준 사용

ISO-8601을 단위 속성값으로 코딩하면 지속 기간 속성값이 시간 지속 기간을 하나의 문자열로 지정하기 위해 ISO 8601 표준에 따라 코딩됩니다. 표준 XML 스키마 데이터 유형 스펙은 duration이라는 데이터 유형을 제공하기 위해 ISO 8601을 사용합니다. 이 데이터 유형은 http://www.w3.org/TR/xmlschema-2/#duration에 자세히 설명되어 있습니다.

표준 XML 스키마에서 duration 데이터 유형의 형식은 다음 문자열입니다.

P*n*Y*n*M*n*DT*n*H*n*M*n*S

- P는 항상 문자열을 시작하는 문자입니다.
- *n*Y는 연도를 표시합니다. 1년은 365일과 동일합니다. 그러므로 1Y를 코딩하면 365D를 코딩하는 것과 같습니다.
- *n*M은 개월 수를 표시합니다. 한달은 30일과 동일합니다. 그러므로 1M을 코딩하면 30D를 코딩하는 것과 같습니다.
- *n*D는 일 수를 표시합니다.
- T는 날짜 단위(연도, 월 및 일)를 시간 단위(시간, 분 및 초)와 구분하는 분리 문자입니다. 시간 단위는 항상 T 다음에 옵니다.
- *n*H는 시간을 표시합니다.
- *n*M은 분을 표시합니다.
- *n*S는 초를 표시합니다.

다음은 형식 예제입니다.

- P5DT12H는 5.5일입니다.
- PT59M59S는 59분 59초입니다.
- P1M은 1개월입니다.

포함 위치

<after>는 다음 요소 내에 포함되어 있습니다.

- <stop>

포함 요소

<after>에 포함된 요소가 없습니다.

attributeAlias 요소

<attributeAlias> 요소는 서로 다른 이벤트에서 동일한 의미이지만 이름이 다른 이벤트 속성을 연관시키는 별명 이름을 제공합니다. 예를 들어, 세 개의 서로 다른 이벤트는 이벤트를 시작한 시스템의 이름을 나타내는 이벤트 속성에 서로 다른 세 개의 이름(호스트, 호스트 이름 및 소스)을 사용할 수 있습니다. <attributeAlias> 요소에는 그룹화 키의 하나의 이벤트 속성으로 연관되어야 하는 개별 이벤트 속성을 설명하는 <eventAttribute> 요소가 포함되어 있습니다.

세부사항

<attributeAlias> 요소 및 해당 aliasName 속성은 그룹화 키의 컨텍스트에서만 유효합니다. 이 요소 및 해당 속성은 <computedValue> 요소 내의 표현식을 포함하여 어떠한 표현식에서도 참조될 수 없습니다.

속성

<attributeAlias>에는 다음 속성이 있습니다.

표 21. <attributeAlias> 요소의 속성

이름	설명	데이터 유형	필수 여부
aliasName	<eventAttribute> 요소에 설명되어 있으며 그룹화 키의 한 이벤트 속성으로 연관될 이벤트 속성의 이름을 정의합니다. 이 이름은 규칙 내에서 고유해야 합니다.	xsd:NMTOKEN	예

포함 위치

<attributeAlias>에는 다음 요소 내에 포함되어 있습니다.

- <groupingKey>

포함 요소

<attributeAlias>는 다음 요소를 포함합니다.

표 22. <attributeAlias> 요소에 포함된 요소

요소	필수 또는 선택적 여부
<eventAttribute>	이 요소의 두 개의 발생이 필요합니다. 추가 발생은 허용되지 않습니다.

attributeName 요소

<attributeName> 요소에는 그룹화 키의 일부인 특정 이벤트 속성이 포함되어 있습니다. 이 이름은 act_event 변수에서 getAttribute 메소드 호출에 사용되는 이름과 일치해야 합니다.

속성

<attributeName>에는 속성이 없습니다.

포함 위치

<attributeName>은 다음 요소 내에 포함되어 있습니다.

- <groupingKey>

포함 요소

<attributeName>에 포함된 요소가 없습니다.

booleanThreshold 요소

<booleanThreshold> 요소는 임계값 규칙에서만 유효합니다. 여기에는 각 이벤트를 받을 때 호출되는 표현식이 포함되어 있습니다. 표현식은 현재 이벤트 및 규칙이 승인한 다른 모든 이벤트를 기준으로 임계값을 계산하고 비교합니다. 표현식은 임계값이 일치하는지 여부를 표시하기 위해 `true` 또는 `false` 부울값을 리턴합니다.

세부사항

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의 『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<booleanThreshold>에는 다음 속성이 있습니다.

표 23. <booleanThreshold> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예

포함 위치

<booleanThreshold>는 다음 요소 내에 포함되어 있습니다.

- <thresholdRule>

포함 요소

<booleanThreshold>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

collectionRule 요소

<collectionRule> 요소는 콜렉션 패턴에 따라 규칙을 정의합니다.

속성

<collectionRule>에는 다음과 같은 속성이 있습니다.

표 24. <collectionRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 `false`입니다.	xsd:boolean	아니오

포함 위치

<collectionRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<collectionRule>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 25. <collectionRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	선택적. 0 또는 1 발생이 허용됩니다.
<groupingKey>	선택적. 0 또는 1 발생이 허용됩니다.
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onTimeWindowComplete>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

12 페이지의『콜렉션 패턴』

콜렉션 패턴으로 정의되는 콜렉션 규칙입니다. 이는 시간 간격 내에서 선택한 이벤트 그룹을 수집합니다. 이는 상태 규칙입니다.

comment 요소

<comment> 요소에는 포함하는 규칙 세트, 규칙 블록, 규칙 또는 변수의 기능과 목적의 설명을 포함할 수 있습니다.

속성

<comment>에는 속성이 없습니다.

포함 위치

<comment>는 다음 요소 내에 포함되어 있습니다.

- <ruleSet>
- <ruleBlock>
- <collectionRule>
- <computationRule>
- <duplicateRule>
- <filterRule>
- <sequenceRule>
- <thresholdRule>
- <timerRule>
- <variable>

포함 요소

<comment>에 포함된 요소가 없습니다.

computationRule 요소

<computationRule> 요소는 계산 패턴에 따라 규칙을 정의합니다.

속성

<computationRule>에는 다음과 같은 속성이 있습니다.

표 26. <computationRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 `false`입니다.	xsd:boolean	아니오

포함 위치

<computationRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<computationRule>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 27. <computationRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	선택적. 0 또는 1 발생이 허용됩니다.
<groupingKey>	선택적. 0 또는 1 발생이 허용됩니다.
<computeFunction>	필수. 1 발생만이 허용됩니다.

표 27. <computationRule> 요소 내에 포함된 요소 (계속)

요소	필수 또는 선택적 여부
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onTimeWindowComplete>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

12 페이지의『계산 패턴』

계산 규칙은 계산 패턴으로 정의됩니다. 이는 표현식을 통해 시간 간격 내에서 각 이벤트를 받을 때 수집된 이벤트에 적용됩니다. 이는 상태 규칙입니다.

computedThreshold 요소

<computedThreshold> 요소는 임계값 규칙에서만 유효합니다. 여기에는 각 이벤트를 받을 때마다 호출되고 현재 이벤트 및 규칙의 이벤트 선택 기준과 일치하는 다른 모든 이벤트를 기준으로 임계값을 계산하는 표현식이 있습니다. 이 표현식은 규칙에 대해 정의된 변수에 저장될 계산된 임계값을 리턴합니다. 그런 다음 규칙은 계산된 임계값을 사용하여 정의된 임계값과 비교합니다.

세부사항

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<computedThreshold>에는 다음과 같은 속성이 있습니다.

표 28. <computedThreshold> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java` 입니다.	xsd:NMTOKEN	예
threshold	도달해야 하는 임계값을 정의합니다. 이 정의된 임계값은 규칙 변수에서 유효한 데이터 유형으로 변환할 수 있는 숫자값의 문자열 표시여야 합니다.	xsd:string	예

표 28. <computedThreshold> 요소의 속성 (계속)

이름	설명	데이터 유형	필수 여부
assignTo	이 표현식으로부터 리턴된 계산된 임계값을 보유하는 변수의 이름을 식별합니다. 이 변수는 <variable> 요소를 사용하는 규칙에 대해 규칙 세트, 규칙 블록 또는 규칙 레벨에서 이미 정의되어 있어야 합니다. 이는 다음과 같은 숫자 데이터 유형 중 하나로 정의되어야 합니다. • java.lang.Double • java.lang.Float • java.lang.Integer • java.lang.Long • java.lang.String 변수가 규칙 세트 또는 규칙 블록 레벨에서 정의되어 있으면 규칙 패턴이 일치한 후에 다시 초기화되지 않습니다.	xsd:NMTOKEN	예
thresholdComparison	계산된 임계값을 정의된 임계값과 비교하기 위한 연산자를 정의합니다. 이 연산자의 올바른 값은 다음과 같습니다. • `lessThan` • `lessThanOrEqualTo` • `greaterThan` • `greaterThanOrEqualTo`	xsd:string	예

포함 위치

<computedThreshold>는 다음 요소 내에 포함되어 있습니다.

- <thresholdRule>

포함 요소

<computedThreshold>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

computedValue 요소

<computedValue> 요소에는 규칙이 이벤트의 하나 이상의 속성값을 기준으로 하는 문자열 값을 작성하기 위한 이벤트를 받을 때 실행되는 표현식이 있습니다. 그런 다음 이 문자열 값은 그룹화 키에 사용할 수 있습니다.

세부사항

때로는 규칙 작성자가 그룹화 키의 항목과 같은 항목을 사용하고 싶어할 수 있습니다.

- 이벤트 속성값의 하위 문자열 예를 들어, 이벤트 속성값에 임베드된 IP 주소가 포함된 경우, <computedValue> 요소 내의 표현식은 해당 IP 주소를 그룹화 키에서 사용할 고유값으로 추출할 수 있습니다.
- 여러 가지 서로 다른 이벤트 속성값의 하위 문자열입니다. 예를 들어, <computedValue> 요소 내의 표현식은 하위 문자열을 추출하고 이를 결합하여 그룹화 키에서 사용할 고유값을 작성할 수 있습니다.

<computedValue> 요소 내의 표현식이 널값을 리턴하면 이 규칙은 이 널값을 누락된 속성값으로 취급합니다.

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의 『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<computedValue>에는 다음 속성이 있습니다.

표 29. <computedValue> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예

포함 위치

<computedValue>는 다음 요소 내에 포함되어 있습니다.

- <groupingKey>

포함 요소

<computedValue>에 포함된 요소가 없습니다.

관련 개념

21 페이지의 『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표

현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

computeFunction 요소

<computeFunction> 요소는 계산 규칙에서만 유효합니다. 여기에는 각 이벤트를 받을 때마다 호출되고 규칙에 정의된 변수에 저장될 값을 리턴하는 표현식이 있습니다. 이 표현식으로부터 리턴된 값은 <computeFunction> 요소의 assignTo 속성에서 명명된 변수의 데이터 유형과 일치해야 합니다.

세부사항

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의 『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<computeFunction>에는 다음과 같은 속성이 있습니다.

표 30. <computeFunction> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예
assignTo	이 표현식으로부터 리턴된 값을 보유하는 변수의 이름을 식별합니다. 이 변수는 <variable> 요소를 사용하는 규칙에 대해 규칙 세트, 규칙 블록 또는 규칙 레벨에서 이미 정의되어 있어야 합니다. 변수가 규칙 세트 또는 규칙 블록 레벨에서 정의되어 있으면 규칙 패턴이 일치한 후에 다시 초기화되지 않습니다.	xsd:NMTOKEN	예

포함 위치

<computeFunction>은 다음 요소 내에 포함되어 있습니다.

- <computationRule>

포함 요소

<computeFunction>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』
표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

dateTime 요소

<dateTime> 요소는 규칙이 활성화되거나 비활성화되는 날짜와 시간을 지정합니다. 그러나 규칙은 규칙이 지정된 시간 전에 실행 중인 ACT(Active Correlation Technology) 엔진으로 로드된 경우에만 활성화되거나 비활성화됩니다.

세부사항

규칙이 활성화로 지정된 시간 전에 실행 중인 ACT(Active Correlation Technology) 엔진으로 로드되지 않은 경우에는 규칙은 활성화되지 않습니다. 규칙이 비활성화로 지정된 시간 전에 실행 중인 ACT(Active Correlation Technology) 엔진으로 로드되지 않은 경우에는 규칙이 <start> 요소에 의해 정의된 상태로 설정되고 <stop> 요소에 의해 비활성화되지 않습니다.

<dateTime> 요소의 내용은 표준 XML 스키마로 된 dateTime 데이터 유형의 형식을 따르는 문자열이어야 합니다. 예를 들어, dateTime은 다음 양식으로 된 한정된 길이의 연속된 문자로 구성되어 있습니다.

yyyy '-' *mm* '-' *dd* 'T' *hh* ':' *mm* ':' *ss* ('.' *s*+)? (*zzzzzz*)?

- *yyyy*는 연도를 표시하는 4자 이상의 숫자입니다. 4자리수를 넘으면 맨 앞에는 영을 사용할 수 없으며 0000도 사용할 수 없습니다.
- 나머지 '-'은 날짜 부분의 파트 간 분리 문자입니다.
- 첫 번째 *mm*은 01부터 시작하며 월을 표시하는 2자리 숫자입니다.
- *dd*는 01부터 시작하며 월의 날짜를 표시하는 2자리 숫자입니다.
- T는 뒤에 시간이 따라옴을 표시하는 분리 문자입니다.
- *hh*는 하루의 시간을 24시간 시스템으로 표시하는 2자리 숫자이며 00으로 시작하고 23으로 끝납니다.
- :은 시간 부분의 파트 간 분리 문자입니다.
- 두 번째 *mm*은 분을 표시하는 2자리 숫자이며 00으로 시작하고 59로 끝납니다.
- *ss*는 전체 초를 표시하는 2자리 숫자이며 00으로 시작하고 59로 끝납니다.
- '.' *s*+가 있는 경우 이는 분수 초를 표시합니다.
- *zzzzzz*가 있는 경우 이는 시간대를 표시합니다. 시간대는 (('+' | '-') *hh* ':' *mm*) | 'Z' 양식으로 한정된 길이의 연속된 문자로 구성되어 있습니다. 여기서

- '+'는 음이 아닌 지속 기간을 표시하며, 이 경우 '-'는 사용하면 안됩니다.
- '-'는 양이 아닌 지속 기간을 표시하며, 이 경우 '+'는 사용하면 안됩니다.
- *hh*는 시간을 표시하는 2자리 숫자이며 00으로 시작하고 14로 끝납니다.
- *mm*은 분을 표시하는 2자리수 숫자이며 00으로 시작하고 59로 끝납니다. 그러나 시간값이 14이면 분값은 00이어야 합니다.
- Z는 UTC(+00:00 또는 -00:00)의 축약이며 따라서 다른 시간대 요소는 사용하면 안됩니다.

다음은 <dateTime> 요소 내용에 대한 두 개의 예제입니다.

- 2005-06-01T13:05:06.07은 로컬 시간으로 2005년 6월 1일 1:05 p.m.에서 6초와 100분의 7초가 지난 시간입니다.
- 2005-06-01T13:05:06.07Z는 UTC 시간으로 2005년 6월 1일 1:05 p.m.에서 6초와 100분의 7초가 지난 시간이며 이는 EDT 시간으로 2005년 6월 1일 9:05 a.m.에서 6초와 100분의 7초가 지난 시간입니다(또는 2005-06-01T09:05:06.07-04:00).

속성

<dateTime>에는 속성이 없습니다.

포함 위치

<dateTime>은 다음 요소 내에 포함되어 있습니다.

- <start>
- <stop>

포함 요소

<dateTime>에 포함된 요소가 없습니다.

deactivateOnEvent 요소

<deactivateOnEvent> 요소는 규칙을 비활성화할 수 있거나 <groupingKey> 요소로 정의된 규칙의 경우 규칙 인스턴스를 비활성화할 수 있는 이벤트를 정의합니다.

다음은 이벤트를 선택하는 세 가지 가능한 방법입니다.

- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소와 함께 사용
- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소 없이 사용
- <filteringPredicate> 요소를 <eventType> 요소 없이 사용

규칙이 활성이고 <eventType> 또는 <filteringPredicate> 요소가 코딩되지 않은 경우에는 발생하는 모든 이벤트가 선택됩니다.

시스템 성능에 부정적인 영향을 미칠 수 있는 <eventType> 요소를 코딩하지 마십시오.

Audit Failure 유형의 모든 이벤트를 선택하고 싶어한다고 가정합니다. 필터링 선언문을 사용하면 특정값의 이벤트 속성이 있는 이벤트만을 포함하도록 선택 기준을 추가로 구체화할 수 있습니다. 예를 들어, Audit Failure 유형의 모든 이벤트를 선택하기 위해 <eventType> 요소를 코딩하고 MyCriticalSystem 값이 있는 호스트 이름 속성이 있는 이벤트만을 선택하도록 <filteringPredicate> 요소를 코딩할 수 있습니다.

속성

<deactivateOnEvent>에는 속성이 없습니다.

포함 위치

<deactivateOnEvent>는 다음 요소 내에 포함되어 있습니다.

- <activationInterval>
- <activationByGroupingKey>

포함 요소

<deactivateOnEvent>는 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 31. <deactivateOnEvent> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<eventType>	선택적. 0 이상의 발생이 허용됩니다.
<filteringPredicate>	선택적. 0 또는 1 발생이 허용됩니다.

duplicateRule 요소

<duplicateRule> 요소는 중복 패턴에 따라 규칙을 정의합니다.

속성

<duplicateRule>에는 다음과 같은 속성이 있습니다.

표 32. <duplicateRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예

표 32. <duplicateRule> 요소의 속성 (계속)

이름	설명	데이터 유형	필수 여부
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 `false`입니다.	xsd:boolean	아니오

포함 위치

<duplicateRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<duplicateRule>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 33. <duplicateRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	선택적. 0 또는 1 발생이 허용됩니다.
<groupingKey>	선택적. 0 또는 1 발생이 허용됩니다.
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onDetection>	선택적. 0 또는 1 발생이 허용됩니다.
<onNextEvent>	선택적. 0 또는 1 발생이 허용됩니다.
<onTimeWindowComplete>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

13 페이지의『중복 패턴』

중복 규칙은 중복 패턴에 의해 정의됩니다. 이는 지정된 시간 간격 내에서 승인된 두 번째 및 후속 이벤트는 포함하지만 이러한 이벤트의 규칙 세트 처리는 건너뜁니다. 이는 상태 규칙입니다.

eventAttribute 요소

<eventAttribute> 요소는 <attributeAlias> 요소에 의해 정의된 속성 별명 이름의 일부로서 이벤트 유형과 이벤트 속성을 연관시키는 방법을 제공합니다.

속성

<eventAttribute>에는 다음과 같은 속성이 있습니다.

표 34. <eventAttribute> 요소의 속성

이름	설명	데이터 유형	필수 여부
type	이벤트 유형의 이름을 정의합니다. 이는 <eventType> 요소에서 유형 속성에 사용되는 이름과 같습니다.	xsd:NMTOKEN	예
attributeName	속성 별명 이름을 통해 다른 이벤트 속성과 연관되는 중인 이벤트 속성의 완전한 이름을 지정합니다. 이 이름은 act_event 변수에서 getAttribute 메소드 호출에 사용되는 이름과 일치해야 합니다.	xsd:string	예

포함 위치

<eventAttribute>는 다음 요소 내에 포함되어 있습니다.

- <attributeAlias>

포함 요소

<eventAttribute>에 포함된 요소가 없습니다.

---

eventCountThreshold 요소

<eventCountThreshold> 요소는 임계값 규칙에서만 유효합니다. 이는 특정 기간에서 이벤트 선택 기준에 부합해야 하는 이벤트 개수를 정의합니다. <eventCountThreshold> 요소는 또한 시간 창에서 두 개의 가능한 시간 간격 모드(고정 또는 슬라이딩) 중 하나를 지정합니다.

세부사항

고정 간격

고정 간격은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 받을 때 시작되고 다음 중 하나가 발생하면 종료됩니다.

- 규칙이 지정된 지속 기간 내에서 임계값에 도달할 때
- 지정된 지속 기간이 경과된 경우

슬라이딩 간격

슬라이딩 간격은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 받을 때 시작됩니다. 그러나 규칙이 임계값에 도달하지 않고 지정된 지속 기간이 경과된 경우, 시간 창은 시작 시간을 새로운 『첫 번째』 이벤트의 시작 시간으로 조정(슬라이드)합니다. 이는 일반적으로 다음 번에 승인되는 이벤트입니다. 슬라이딩 간격은 다음 중 하나가 발생할 때까지 이러한 방법으로 계속해서 조정합니다.

- 규칙이 지정된 지속 기간 내에서 임계값에 도달할 때
- 시간 창을 시작한 이벤트를 받은 후에 지정된 지속 기간 내에 후속 이벤트를 받지 않은 경우

시간 창을 시작한 이벤트(새로운『첫 번째』이벤트가 됨)는 이 기준에 부합하는 수신 시간이 있는 이벤트입니다. 규칙의 시간 간격 지속 기간에 추가된 수신 시간은 현재 시간 이후입니다. 다음은 방정식 양식으로 된 기준입니다.

event reception time + time interval duration for rule > current time

이러한 이벤트가 없으면 슬라이딩 간격이 더 이상 시간을 조정하지 않고 간격이 종료됩니다.

임계값 규칙은 임계값에 도달하거나 기간이 종료될 때까지 각 승인된 이벤트를 셈합니다. 그런 다음 이는 <onDetection> 요소 또는 <onTimeOut> 요소 내에 정의된 조치를 적절하게 실행합니다.

<onDetection> 조치

이러한 조치는 이벤트 계수가 <eventCountThreshold> 요소의 임계값 속성에 의해 정의된 값과 동일할 때 실행됩니다. 이는 임계값에 부합됨을 표시합니다.

<onTimeOut> 조치

이러한 조치가 실행되는 시기는 시간 간격 모드가 고정 또는 슬라이딩인지 여부에 따라 다릅니다.

고정 모드

고정 모드를 사용하면 시간 창이 만기될 때 이러한 조치가 실행됩니다.

슬라이딩 모드

슬라이딩 모드를 사용하면 시간 창을 시작한 이벤트를 받은 후에 지정된 지속 기간 내에 후속 이벤트를 받지 않은 경우에 이러한 조치가 실행됩니다. 다시 말하면 규칙의 시간 간격 지속 기간에 추가된 수신 시간이 현재 시간 이후인 이벤트를 받지 않았습니다.

시간 창의 시간 간격 모드는 <eventCountThreshold> 요소의 timeIntervalMode 속성에 의해 정의되어 있습니다. 다음 시나리오는 두 개의 가능한 시간 간격 모드 간의 차이점과 작동을 설명합니다.

고정 및 슬라이딩 모드를 설명하는 시나리오

규칙이 이벤트 선택 기준에 부합하는 네 개의 이벤트를 8:00, 8:04, 8:06 및 8:07에 하나씩 받았다고 가정합니다. 이벤트 계수 임계값은 3이고 시간 창의 지속 기간은 5분입니다.

fixed 모드의 규칙 작동

이 시간 간격 모드를 사용하면 임계값 규칙은 8:00에 처리를 시작하고 8:05에 <onTimeOut> 조치를 실행합니다. 이는 5분 동안 두 개의 이벤트만을 받기 때

문입니다. 그러므로 시간 창 내에서 임계값에 도달하지 않습니다. 8:06에 세 번째 이벤트를 받으면 임계값 규칙이 처리를 다시 시작하고 8:11에 <onTimeOut> 조치를 실행합니다. 이는 5분 내에 두 개의 이벤트만을 받기 때문입니다.

고정 모드는 정적입니다.

sliding 모드의 규칙 작동

이 시간 간격 모드를 사용하면 임계값 규칙은 8:00에 처리를 시작합니다. 시간 창이 완료 예약되어 있는 8:05에 규칙은 두 개의 이벤트만을 받았다고 판별합니다. 그런 다음 규칙은 8:00에 받은 이벤트를 버리고 지속 기간을 8:09에 종료하도록 다시 계산합니다(첫 번째 이벤트가 8:04에 받은 이벤트이기 때문). 규칙이 8:07에 이벤트를 받으면 <onDetection> 조치를 실행합니다. 최신 시간 창(8:04 – 8:09)에서 임계값에 도달했기 때문입니다(8:04, 8:06 및 8:07에 세 개의 이벤트).

슬라이딩 모드는 시간 창에서 임계값에 도달하기 위해 시작 시간을 계속해서 조정(슬라이드)한다는 점에서 동적입니다.

이제 규칙이 이벤트 선택 기준에 부합하는 네 개의 이벤트를 8:00, 8:04, 8:06 및 8:10에 하나씩 받았다고 가정합니다. 이벤트 계수 임계값은 3이고 시간 창의 지속 기간은 5분입니다.

sliding 모드의 규칙 작동

이 경우 임계값 규칙은 8:00에 처리를 시작합니다. 시간 창이 완료 예약되어 있는 8:05에 규칙은 두 개의 이벤트만을 받았다고 판별합니다. 그런 다음 규칙은 8:00에 받은 이벤트를 버리고 지속 기간을 8:09에 종료하도록 다시 계산합니다(첫 번째 이벤트가 8:04에 받은 이벤트이기 때문).

시간 창이 완료 예약되어 있는 8:09에 규칙은 두 개의 이벤트만을 받았다고 판별합니다. 그런 다음 규칙은 8:04에 받은 이벤트를 버리고 지속 기간을 8:11에 종료하도록 다시 계산합니다(첫 번째 이벤트가 8:06에 받은 이벤트이기 때문).

시간 창이 완료 예약되어 있는 8:11에 규칙은 두 개의 이벤트만을 받았다고 판별합니다. 그런 다음 규칙은 8:06에 받은 이벤트를 버리고 지속 기간을 8:15에 종료하도록 다시 계산합니다(첫 번째 이벤트가 8:10에 받은 이벤트이기 때문).

시간 창이 완료 예약되어 있는 8:15에는 규칙은 시간 창이 시작된 8:10의 이벤트 이후로 이벤트를 받지 않았다고 판별합니다. 그런 다음 규칙은 <onTimeOut> 조치를 실행합니다.

속성

<eventCountThreshold>에는 다음과 같은 속성이 있습니다.

표 35. <eventCountThreshold> 요소의 속성

이름	설명	데이터 유형	필수 여부
임계값	특정 기간에서 이벤트 선택 기준에 부합해야 하는 이벤트 개수를 정의합니다. 도달해야 하는 이벤트 계수 임계값입니다. 이 값은 양의 정수여야 합니다.	xsd:positiveInteger	예
timeIntervalMode	시간 창의 시간 간격이 고정 또는 슬라이딩인지 여부를 정의합니다. 이 속성의 올바른 값은 다음과 같습니다. • `fixed`(기본값) • `sliding`	xsd:string	아니오

포함 위치

<eventCountThreshold>는 다음 요소 내에 포함되어 있습니다.

- <thresholdRule>

포함 요소

<eventCountThreshold>에 포함된 요소가 없습니다.

eventSelector 요소

<eventSelector> 요소는 규칙이 처리하기 위해 선택한 이벤트를 정의합니다.

세부사항

다음은 이벤트를 선택하는 세 가지 가능한 방법입니다.

- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소와 함께 사용
- 하나 이상의 <eventType> 요소를 <filteringPredicate> 요소 없이 사용
- <filteringPredicate> 요소를 <eventType> 요소 없이 사용

규칙이 모든 이벤트를 처리하기를 원하는 특별한 경우 다음과 같은 옵션이 있습니다.

- <eventSelector> 요소를 코딩하지 마십시오.
- 요소를 포함하지 않는 <eventSelector> 요소를 코딩하십시오.

시스템 성능에 부정적인 영향을 미칠 수 있는 <eventType> 요소를 코딩하지 마십시오.

`Audit Failure` 유형의 모든 이벤트를 선택하고 싶어한다고 가정합니다. 필터링 선언문을 사용하면 특정값의 이벤트 속성이 있는 이벤트만을 포함하도록 선택 기준을 추가로 구체화할 수 있습니다. 예를 들어, `Audit Failure` 유형의 모든 이벤트를 선택하기

위해 <eventType> 요소를 코딩하고 MyCriticalSystem 값이 있는 호스트 이름 속성이 있는 이벤트만을 선택하도록 <filteringPredicate> 요소를 코딩할 수 있습니다.

속성

<eventSelector>에는 다음 속성이 있습니다.

표 36. <eventSelector> 요소의 속성

이름	설명	데이터 유형	필수 여부
alias	이 속성은 여러 <eventSelector> 요소를 가지는 유일한 규칙인 연속 규칙 내에서만 올바릅니다. 이는 연속 규칙에서 특정 이벤트 선택자에 의해 선택된 이벤트에 고유하게 이름을 지정합니다. 그런 다음 필터링 선언문 및 조치는 이 별명 이름을 사용하여 해당 이벤트에 액세스합니다.	xsd:NMTOKEN	아니오

포함 위치

<eventSelector>는 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>
- <filterRule>
- <sequenceRule>
- <thresholdRule>

포함 요소

<eventSelector>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 37. <eventSelector> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<eventType>	선택적. 0 이상의 발생이 허용됩니다.
<filteringPredicate>	선택적. 0 또는 1 발생이 허용됩니다.

eventType 요소

<eventType> 요소는 규칙이 처리를 위해 선택했거나 규칙을 활성화하거나 비활성화하는 이벤트의 유형을 정의합니다.

속성

<eventType>에는 다음 속성이 있습니다.

표 38. <eventType> 요소의 속성

이름	설명	데이터 유형	필수 여부
type	이벤트 유형을 정의합니다. CBE(Common Base Event) 스펙을 준수하는 이벤트의 경우 이 이름은 extensionName 속성의 값입니다. IBM Tivoli Enterprise Console® 이벤트의 경우, 이 이름은 BAROC 파일에 정의된 이벤트 클래스 이름입니다. 다른 형식을 기반으로 하는 이벤트는 다른 속성을 사용하여 이벤트 유형을 지정할 수도 있습니다.	xsd:NMTOKEN	예

포함 위치

<eventType>은 다음 요소 내에 포함되어 있습니다.

- <activateOnEvent>
- <deactivateOnEvent>
- <eventSelector>

포함 요소

<eventType>에 포함된 요소가 없습니다.

filteringPredicate 요소

<filteringPredicate> 요소에는 규칙이 처리하기 위해 선택하거나 규칙을 활성화하거나 비활성화하기 위해 선택되는 이벤트를 추가로 제한하는 표현식이 포함되어 있습니다. 그러므로 <eventType> 요소를 통해 이벤트 유형만으로 필터링하는 것보다 보다 종합적으로 이벤트를 필터링할 수 있습니다.

세부사항

표현식은 조건을 정의하고 `true`(조건이 맞음) 또는 `false`(조건이 맞지 않음)의 부울값을 리턴합니다.

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의 『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<filteringPredicate>에는 다음 속성이 있습니다.

표 39. <filteringPredicate> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예

포함 위치

<filteringPredicate>는 다음 요소 내에 포함되어 있습니다.

- <activateOnEvent>
- <deactivateOnEvent>
- <eventSelector>

포함 요소

<filteringPredicate>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

filterRule 요소

<filterRule> 요소는 필터 패턴에 따라 규칙을 정의합니다.

속성

<filterRule>에는 다음과 같은 속성이 있습니다.

표 40. <filterRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예

표 40. <filterRule> 요소의 속성 (계속)

이름	설명	데이터 유형	필수 여부
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 false입니다.	xsd:boolean	아니오

포함 위치

<filterRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<filterRule>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 41. <filterRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	선택적. 0 또는 1 발생이 허용됩니다.
<onDetection>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

14 페이지의『필터 패턴』

필터 규칙은 필터 패턴으로 정의됩니다. 이는 이벤트를 승인할 때 특정 조치를 수행합니다. 이는 단일 이벤트에 대해서만 작동하므로 상태가 없는 규칙입니다.

groupingKey 요소

일반적으로 각 활성 규칙에는 ACT(Active Correlation Technology) 엔진에서 실행 중인 하나의 규칙 인스턴스 또는 사본이 있습니다. 그러나 때로는 종종 다른 자원 그룹과 관련되는 다른 이벤트 그룹에 동일한 규칙이 필요하기도 합니다. 그룹화 키는 선택한 이벤트를 한 그룹으로 고유하게 처리하기 위해 다른 그룹으로 분리하는 데 사용할 수 있는 하나 이상의 이벤트 속성 또는 이벤트 속성의 일부입니다. <groupingKey> 요소는 규칙의 그룹화 키를 정의합니다. <groupingKey> 요소의 용도는 규칙에게 공통 특성을

공유하는 각 이벤트 그룹의 규칙 인스턴스(또는 사본)를 작성하도록 지시하는 것입니다(그룹화 키를 구성하는 해당 속성값에 의해 정의됨).

세부사항

다음 두 개의 시나리오는 그룹화 키의 중요성을 설명합니다.

시나리오 1:
DB2down 이벤트와 DB2up 이벤트의 두 개의 이벤트가 발생합니다. DB2 프로그램은 A, B 및 C라는 세 대의 컴퓨터에서 실행됩니다. DB2down 이벤트를 DB2up 이벤트에 연관시키고 DB2 프로그램이 중지된 후 다시 시작되지 않으면 운영자에게 경보를 보내는 연속 규칙이 정의되어 있습니다.

연속 규칙이 그룹화 키 없이 정의되어 있고 컴퓨터 A로부터 DB2down 이벤트를 받은 경우, 다른 컴퓨터로부터 받은 DB2up 이벤트의 순서는 완료되지만 이 작업은 계획대로 수행되지 않습니다. 그러나 그룹화 키가 호스트 이름 속성으로 정의된 경우 규칙의 고유 사본 또는 인스턴스는 선택된 이벤트의 호스트 이름 속성의 고유값마다 작성됩니다. ACT(Active Correlation Technology) 엔진은 각 이벤트를 올바른 규칙 인스턴스(해당 이벤트의 호스트 이름 값의 규칙 인스턴스)로 보냅니다. 그러므로 컴퓨터 A로부터 DB2down 이벤트를 받으면 ACT(Active Correlation Technology) 엔진은 컴퓨터 A의 규칙 인스턴스를 작성합니다. 컴퓨터 B로부터 DB2down 이벤트를 받으면 ACT(Active Correlation Technology) 엔진은 컴퓨터 B의 두 번째 규칙 인스턴스를 작성합니다. 컴퓨터 B로부터 DB2up 이벤트를 받으면 ACT(Active Correlation Technology) 엔진은 두 번째 규칙 인스턴스에서 해당 이벤트를 처리합니다. 컴퓨터 B로부터의 DB2down 및 DB2up 이벤트가 올바르게 연관되지 않았으므로 순서가 완료되고 운영자는 경보를 받습니다.

시나리오 2:
특정 환경에 있는 모든 컴퓨터에서 `Incorrect login attempted` 메시지의 이벤트가 발생합니다. 이벤트에는 사용자 ID가 포함되어 있습니다. 임계값 규칙은 이 이벤트가 5분안에 10차례 넘게 발생한 경우 운영자에게 경고를 발행하도록 정의되어 있습니다.

그룹화 키는 사용자 ID로서 정의될 수 있습니다. 그런 다음 각 고유 사용자 ID마다 새 규칙 인스턴스가 작성됩니다. 각 사용자의 로그인 시도는 각 인스턴스가 해당 사용자의 로그인 시도 횟수를 별도로 카운트하여 고유 임계값 규칙 인스턴스에서 추적됩니다. 사용자 ID가 5분 동안 10회의 잘못된 로그인 횟수를 넘으면 운영자는 경고를 받게 됩니다.

이 개념의 다른 변수는 다음과 같습니다.

- 그룹화 키는 사용자 ID가 아닌 호스트 이름으로서 정의될 수 있습니다. 이 옵션은 단일 컴퓨터에서 대량의 잘못된 로그인 시도를 발견할 수 있습니다.

- 그룹화 키는 호스트 이름 및 사용자 ID의 조합으로 정의될 수도 있습니다. 이 옵션은 특정 컴퓨터에 대한 특정 사용자 ID의 잠재적인 해킹 시도를 발견할 수 있습니다.

규칙에 지정된 모든 이벤트 유형에 동일한 속성이 있으면 <attributeName> 요소를 사용하는 것이 그룹화 키를 정의하기 위한 가장 단순하고 일반적인 방법입니다.

속성

<groupingKey>에는 다음 속성이 있습니다.

표 42. <groupingKey> 요소의 속성

이름	설명	데이터 유형	필수 여부
missingAttributeHandling	규칙이 다음과 같은 조건에서 수행해야 하는 조치를 정의합니다. • 선택한 이벤트에 그룹화 키에 참여하는 속성이 있지만 해당 속성값이 누락된 경우. • <computedValue> 요소의 표현식이 널값을 리턴하는 경우. 규칙은 이 널값을 누락된 속성값으로 취급합니다. missingAttributeHandling 속성의 올바른 값은 다음과 같습니다. • `ignoreEvent`(기본값) - 규칙이 이벤트를 무시하고 아무런 조치를 취하지 않음을 의미합니다. • `ignoreAttribute` - 규칙이 이벤트를 승인하지만 값이 누락된 속성을 무시함을 의미합니다. 그러면 ACT (Active Correlation Technology) 엔진에는 속성의 대체값이 포함됩니다.	xsd:string	아니오

포함 위치

<groupingKey>는 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>
- <sequenceRule>
- <thresholdRule>

포함 요소

<groupingKey>에는 다음 요소가 포함되어 있습니다.

표 43. <groupingKey> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<attributeAlias>	이러한 요소 중 하나는 필수입니다. 이러한 요소를 둘 이상 코딩하는 것은 선택적입니다. 세 요소 모두 여러 번 발생이 허용됩니다. 이러한 요소는 어떤 순서로든지 코딩될 수 있습니다.
<attributeName>	
<computedValue>	

import 요소

<import> 요소에는 규칙 내의 다른 표현식에 사용하기 위해 가져올 외부 모듈 (예: Java 클래스)을 지정하는 표현식이 포함되어 있습니다.

세부사항

표현식 코드는 <import> 요소 내의 문자열입니다. ACT(Active Correlation Technology) 컴파일러는 <import> 요소가 제공하는 import 문을 사용하여 외부 메소드를 호출하는 규칙 내에서 표현식 코드를 컴파일합니다.

속성

<import>에는 다음 속성이 있습니다.

표 44. <import> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예

포함 위치

<import>는 다음 요소 내에 포함되어 있습니다.

- <ruleSet>
- <ruleBlock>

포함 요소

<import>에 포함된 요소가 없습니다.

관련 개념

23 페이지의『외부 모듈 및 오브젝트 가져오기 및 액세스』
이 예제는 외부 코드(예: Java 클래스) 및 외부 오브젝트에서 표현식에 액세스를 가능하게 하는 방법을 보여줍니다. 외부 오브젝트는 어플리케이션과 표현식과의 통신을 위해 작성된 오브젝트입니다.

inactiveWhenLoaded 요소

<inactiveWhenLoaded> 요소는 ACT(Active Correlation Technology) 엔진이 규칙을 로드할 때 규칙이 비활성화되록 지정합니다. 규칙은 다른 수단을 사용하여 활성화될 때까지 비활성 상태로 남아 있습니다.

속성

<inactiveWhenLoaded>에는 속성이 없습니다.

포함 위치

<inactiveWhenLoaded>는 다음 요소 내에 포함되어 있습니다.

- <start>

포함 요소

<inactiveWhenLoaded>에 포함된 요소가 없습니다.

lifeCycleActions 요소

<lifeCycleActions> 요소에는 규칙의 수명 주기에서 네 개의 기본 단계에서 취할 조치를 정의하는 요소가 포함되어 있습니다.

세부사항

로드 및 활성화 단계에 정의된 조치는 규칙이 실제로 로드되거나 활성화된 후 규칙이 아직 처리를 시작하기 전에 호출됩니다. 비활성화 및 로드 해제 단계에 정의된 조치는 규칙이 실제로 비활성화되거나 로드 해제되기 직전에 호출됩니다.

속성

<lifeCycleActions>에는 속성이 없습니다.

포함 위치

<lifeCycleActions>는 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>

- <filterRule>
- <sequenceRule>
- <thresholdRule>
- <timerRule>

포함 요소

<lifeCycleActions>에는 다음 요소가 포함되어 있습니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 45. <lifeCycleActions> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<onLoad>	선택적. 0 또는 1 발생이 허용됩니다.
<onActivation>	선택적. 0 또는 1 발생이 허용됩니다.
<onDeactivation>	선택적. 0 또는 1 발생이 허용됩니다.
<onUnload>	선택적. 0 또는 1 발생이 허용됩니다.

never 요소

<never> 요소는 규칙이 특정 시간에 절대로 비활성화되지 않도록 지정합니다. 그러나 이벤트나 다른 수단을 사용하여 규칙을 비활성화할 수 있습니다.

속성

<never>에는 속성이 없습니다.

포함 위치

<never>는 다음 요소 내에 포함되어 있습니다.

- <stop>

포함 요소

<never>에 포함된 요소가 없습니다.

onActivation 요소

<onActivation> 요소는 규칙이 활성화될 때 취할 조치 또는 조치 세트를 지정합니다. <onActivation> 조치는 규칙이 활성화된 후 규칙이 처리를 시작하기 전에 호출됩니다.

세부사항

규칙 세트에 동일한 날짜와 시간에 또는 동일한 이벤트에 의해 활성화되는 여러 개의 규칙이 있거나 동일한 시간 창을 가지는 여러 개의 규칙이 있으면 이러한 규칙에 대한 다음과 같은 조치는 정확히 같은 시간에 실행되지 않습니다.

- <onTimeOut> 및 <onTimeWindowComplete> 요소 내의 규칙 응답 조치
- <onActivation> 및 <onDeactivation> 요소 내의 수명 주기 조치

이러한 조치는 어떤 순서로든지 실행할 수 있습니다. 이러한 조치를 규칙 세트에 코딩된 순서대로 반드시 실행할 필요는 없습니다. 각 조치는 순서에서 다음 조치가 시작되기 전에 완료되어야 하므로 조치는 동일한 시간에 실행되지 않습니다.

속성

<onActivation>에는 속성이 없습니다.

포함 위치

<onActivation>은 다음 요소 내에 포함되어 있습니다.

- <lifeCycleActions>

포함 요소

<onActivation>은 다음 요소를 포함합니다.

표 46. <onActivation> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onDeactivation 요소

<onActivation> 요소는 규칙이 비활성화될 때 취할 조치 또는 조치 세트를 지정합니다. <onDeactivation> 조치는 규칙이 비활성화되기 직전에 호출됩니다.

세부사항

규칙 세트에 동일한 날짜와 시간에 또는 동일한 이벤트에 의해 활성화되는 여러 개의 규칙이 있거나 동일한 시간 창을 가지는 여러 개의 규칙이 있으면 이러한 규칙에 대한 다음과 같은 조치는 정확히 같은 시간에 실행되지 않습니다.

- <onTimeOut> 및 <onTimeWindowComplete> 요소 내의 규칙 응답 조치
- <onActivation> 및 <onDeactivation> 요소 내의 수명 주기 조치

이러한 조치는 어떤 순서로든지 실행할 수 있습니다. 이러한 조치를 규칙 세트에 코딩된 순서대로 반드시 실행할 필요는 없습니다. 각 조치는 순서에서 다음 조치가 시작되

기 전에 완료되어야 하므로 조치는 동일한 시간에 실행되지 않습니다.

속성

<onDeactivation>에는 속성이 없습니다.

포함 위치

<onDeactivation>은 다음 요소 내에 포함되어 있습니다.

- <lifeCycleActions>

포함 요소

<onDeactivation>은 다음 요소를 포함합니다.

표 47. <onDeactivation> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onDetection 요소

<onDetection> 요소는 중복, 필터, 연속 및 임계값 규칙에서만 유효합니다. 이는 규칙 패턴이 발견될 때 취할 조치 또는 조치 세트를 지정합니다.

세부사항

표 48에서는 <onDetection> 조치가 올바른 각 규칙 유형에 대한 규칙 패턴을 발견하는 방법을 설명합니다.

표 48. 규칙 유형을 기반으로 규칙 패턴을 발견하는 방법

규칙 유형	규칙 패턴 발견 방법
중복	이 규칙 패턴은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 받을 때 발견됩니다.
필터	이 규칙 패턴은 이벤트 선택 기준에 부합하는 이벤트를 받을 때 발견됩니다.
연속	이 규칙 패턴은 이벤트 선택 기준에 부합하는 이벤트를 순서대로 일정 시간 내에 받을 때 발견됩니다.
임계값	이 규칙 패턴은 이벤트 선택 기준에 부합하는 이벤트를 시간 창 내에서 받고 임계값에 도달할 때 발견됩니다.

속성

<onDetection>에는 속성이 없습니다.

포함 위치

<onDetection>은 다음 요소 내에 포함되어 있습니다.

- <duplicateRule>

- <filterRule>
- <sequenceRule>
- <thresholdRule>

포함 요소

<onDetection>은 다음 요소를 포함합니다.

표 49. <onDetection> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onLoad 요소

<onLoad> 요소는 실행 중인 ACT(Active Correlation Technology) 엔진에서 규칙이 로드되거나 전개될 때 취할 조치 또는 조치 세트를 지정합니다. <onLoad> 조치는 규칙이 로드된 후 규칙이 처리를 시작하기 전에 호출됩니다.

속성

<onLoad>에는 속성이 없습니다.

포함 위치

<onLoad>는 다음 요소 내에 포함되어 있습니다.

- <lifeCycleActions>

포함 요소

<onLoad>는 다음 요소를 포함합니다.

표 50. <onLoad> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onNextEvent 요소

<onNextEvent> 요소는 중복 규칙에서만 유효합니다. 이는 중복 규칙이 지정된 시간 창 내에서 이벤트 선택 기준에 부합하는 두 번째 및 각 후속 이벤트를 받을 때 취할 조치 또는 조치 세트를 지정합니다.

세부사항

중복 규칙의 경우 ACT(Active Correlation Technology) 엔진은 지정된 시간 창 내에서 이벤트 선택 기준에 부합하는 두 번째 및 각 후속 이벤트에 대한 규칙 세트 처리를 건너뜁니다. 그러므로 <onNextEvent> 조치를 코딩하는 유일한 이유는 두 번째 및 각 후속 이벤트의 대체 처리를 지정하기 위한 것입니다.

속성

<onNextEvent>에는 속성이 없습니다.

포함 위치

<onNextEvent>는 다음 요소 내에 포함되어 있습니다.

- <duplicateRule>

포함 요소

<onNextEvent>는 다음 요소를 포함합니다.

표 51. <onNextEvent> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onTimeOut 요소

<onTimeOut> 요소는 연속 및 임계값 규칙에서만 유효합니다. 이는 규칙의 시간 창이 만기되는 경우 취할 조치 또는 조치 세트를 지정합니다.

세부사항

표 52에서는 <onTimeOut> 조치가 올바른 각 규칙 유형별 시간 창이 만기하는 방법을 설명합니다.

표 52. 규칙 유형별 시간 창 만기 방법

규칙 유형	시간 창 만기 방법
연속	하나 이상의 이벤트가 승인되었지만 이벤트의 전체 순서를 시간 창 내에서 받지 않은 경우에 시간 창이 만기합니다.
임계값	하나 이상의 이벤트가 승인되었지만 시간 창 내에서 임계값에 도달하지 않은 경우 시간 창이 만기합니다.

규칙 세트에 동일한 날짜와 시간에 또는 동일한 이벤트에 의해 활성화되는 여러 개의 규칙이 있거나 동일한 시간 창을 가지는 여러 개의 규칙이 있으면 이러한 규칙에 대한 다음과 같은 조치는 정확히 같은 시간에 실행되지 않습니다.

- <onTimeOut> 및 <onTimeWindowComplete> 요소 내의 규칙 응답 조치

- <onActivation> 및 <onDeactivation> 요소 내의 수명 주기 조치

이러한 조치는 어떤 순서로든지 실행할 수 있습니다. 이러한 조치를 규칙 세트에 코딩된 순서대로 반드시 실행할 필요는 없습니다. 각 조치는 순서에서 다음 조치가 시작되기 전에 완료되어야 하므로 조치는 동일한 시간에 실행되지 않습니다.

속성

<onTimeOut>에는 속성이 없습니다.

포함 위치

<onTimeOut>은 다음 요소 내에 포함되어 있습니다.

- <sequenceRule>
- <thresholdRule>

포함 요소

<onTimeOut>은 다음 요소를 포함합니다.

표 53. <onTimeOut> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onTimeWindowComplete 요소

<onTimeWindowComplete> 요소는 콜렉션, 계산, 중복 및 타이머 규칙에서만 유효합니다. 이는 규칙의 시간 창이 종료될 때 취할 조치 또는 조치 세트를 지정합니다.

세부사항

규칙 세트에 동일한 날짜와 시간에 또는 동일한 이벤트에 의해 활성화되는 여러 개의 규칙이 있거나 동일한 시간 창을 가지는 여러 개의 규칙이 있으면 이러한 규칙에 대한 다음과 같은 조치는 정확히 같은 시간에 실행되지 않습니다.

- <onTimeOut> 및 <onTimeWindowComplete> 요소 내의 규칙 응답 조치
- <onActivation> 및 <onDeactivation> 요소 내의 수명 주기 조치

이러한 조치는 어떤 순서로든지 실행할 수 있습니다. 이러한 조치를 규칙 세트에 코딩된 순서대로 반드시 실행할 필요는 없습니다. 각 조치는 순서에서 다음 조치가 시작되기 전에 완료되어야 하므로 조치는 동일한 시간에 실행되지 않습니다.

속성

<onTimeWindowComplete>에는 속성이 없습니다.

포함 위치

<onTimeWindowComplete>는 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>
- <timerRule>

포함 요소

<onTimeWindowComplete>는 다음 요소를 포함합니다.

표 54. <onTimeWindowComplete> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

onUnload 요소

<onUnload> 요소는 규칙이 실행 중인 ACT(Active Correlation Technology) 엔진에서 로드 해제되거나 제거될 때 취할 조치 또는 조치 세트를 지정합니다. <onUnload> 조치는 규칙이 로드 해제되기 직전에 호출됩니다.

속성

<onUnload>에는 속성이 없습니다.

포함 위치

<onUnload>는 다음 요소 내에 포함되어 있습니다.

- <lifeCycleActions>

포함 요소

<onUnload>는 다음 요소를 포함합니다.

표 55. <onUnload> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<action>	선택적. 0 이상의 발생이 허용됩니다.

ruleBlock 요소

<ruleBlock> 요소는 관련된 규칙을 그룹화하고 규칙을 계층 구조로 조직하는 방법을 제공합니다.

속성

<ruleBlock>에는 다음 속성이 있습니다.

표 56. <ruleBlock> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙 블록을 식별합니다. 이 ID는 규칙 세트 또는 이 규칙 블록을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예

포함 위치

<ruleBlock>은 다음 요소 내에 포함되어 있습니다.

- <ruleSet>
- <ruleBlock>

포함 요소

<ruleBlock>는 다음 요소를 포함합니다.

<comment>, <import> 및 <variable> 요소는 코딩된 경우 표시된 순서대로 코딩되어야 합니다. 나머지 요소는 순서와 무관하게 코딩될 수 있습니다.

표 57. <ruleBlock> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<import>	선택적. 영 이상의 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<ruleBlock>	선택적. 영 이상의 발생이 허용됩니다.
<collectionRule>	선택적. 영 이상의 발생이 허용됩니다.
<computationRule>	선택적. 영 이상의 발생이 허용됩니다.
<duplicateRule>	선택적. 영 이상의 발생이 허용됩니다.
<filterRule>	선택적. 영 이상의 발생이 허용됩니다.
<sequenceRule>	선택적. 영 이상의 발생이 허용됩니다.
<thresholdRule>	선택적. 영 이상의 발생이 허용됩니다.
<timerRule>	선택적. 영 이상의 발생이 허용됩니다.

ruleSet 요소

act:ruleSet에 의해 정의된 <ruleSet> 요소는 ACT(Active Correlation Technology) 규칙 언어의 루트 요소입니다. 다른 모든 요소는 이 <ruleSet> 요소 내에 포함되어 있습니다.

세부사항

ACT(Active Correlation Technology) 언어 스키마(act:ruleSet) 및 ACT(Active Correlation Technology) 기반 규칙 세트 스키마(br:ruleSet)에 의해 정의된 <ruleSet> 요소는 중복입니다. 그러나 규칙 세트를 작성할 때에는 <ruleSet> 요소에 다음 이름 공간을 지정해야 합니다: act:ruleSet.

속성

<ruleSet>에는 다음 속성이 있습니다.

표 58. <ruleSet> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙 세트를 식별합니다. 이 ID는 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예

포함 위치

<ruleSet>는 규칙 언어의 루트 요소이므로 다른 요소 내에 포함될 수 없습니다.

포함 요소

<ruleSet>는 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 59. <ruleSet> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<import>	선택적. 0 이상의 발생이 허용됩니다.
<variable>	선택적. 0 이상의 발생이 허용됩니다.
<ruleBlock>	선택적. 0 이상의 발생이 허용됩니다.

runUntilDeactivated 요소

<runUntilDeactivated> 요소는 시간 창이 규칙이 비활성화될 때까지 계속해서 열려 있도록 지정합니다. 그러므로 이 규칙의 시작 창은 규칙이 처리를 시작할 때 시작되고, 규칙이 비활성화되거나 규칙 세트에서 제거되거나 ACT(Active Correlation Technology) 엔진이 종료되기 전까지는 중지하지 않습니다.

세부사항

<runUntilDeactivated> 요소를 포함하는 규칙의 특정 작동은 규칙 유형에 따라 다릅니다. 114 페이지의 표 60 에서는 <timeWindow> 요소가 유효하고

<runUntilDeactivated> 요소를 포함하는 각 규칙 유형별 규칙 작동을 설명합니다.

표 60. <runUntilDeactivated>가 코딩될 때 규칙 작동

규칙 유형	**<runUntilDeactivated>가 코딩될 때 규칙 작동**
콜렉션	콜렉션 규칙은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 승인하고 규칙이 비활성화될 때까지 계속해서 이벤트를 승인하고 처리합니다. 규칙이 비활성화될 때는 <onTimeWindowComplete> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다.
계산	계산 규칙은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 승인하고 규칙이 비활성화될 때까지 계속해서 이벤트를 승인하고 처리합니다. 규칙이 비활성화될 때는 <onTimeWindowComplete> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다.
중복	중복 규칙은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 승인하고 규칙이 비활성화될 때까지 계속해서 이벤트를 승인하고 처리합니다. 규칙이 비활성화될 때는 <onTimeWindowComplete> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다.
연속	연속 규칙은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 승인하고 다음 상황이 발생할 때까지 계속해서 이벤트를 승인하고 처리합니다. • 연속 패턴이 발견되었습니다. 이 경우 <onDetection> 요소 내에 정의된 조치가 실행되고 규칙은 초기 상태로 되돌아갑니다. 이 규칙이 처리하는 이벤트가 다시 시작되고 이 프로세스는 규칙이 비활성화될 때까지 여러 번 반복될 수 있습니다. • 규칙이 이벤트를 처리하는 도중에 비활성화되었습니다. 이 경우 <onTimeOut> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다.
임계값	임계값 규칙은 이벤트 선택 기준에 부합하는 첫 번째 이벤트를 승인하고 다음 상황이 발생할 때까지 계속해서 이벤트를 승인하고 처리합니다. • 임계값 패턴이 발견되었습니다. 이 경우 <onDetection> 요소 내에 정의된 조치가 실행되고 규칙은 초기 상태로 되돌아갑니다. 이 규칙이 처리하는 이벤트가 다시 시작되고 이 프로세스는 규칙이 비활성화될 때까지 여러 번 반복될 수 있습니다. • 규칙이 이벤트를 처리하는 도중에 비활성화되었습니다. 이 경우 <onTimeOut> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다.
타이머	타이머 규칙은 활성이 된 후에 비활성화될 때까지 아무 것도 수행하지 않습니다. 규칙이 비활성화될 때에는 <onTimeWindowComplete> 요소 내에 정의된 조치가 실행되고 <onDeactivation> 요소 내에 정의된 조치가 즉시 뒤따라옵니다. <timerRule> 요소에서 반복 속성은 무시됩니다.

속성

<runUntilDeactivated>에는 속성이 없습니다.

포함 위치

<runUntilDeactivated>는 다음 요소 내에 포함되어 있습니다.

- <timeWindow>

포함 요소

<runUntilDeactivated>에 포함된 요소가 없습니다.

sequenceRule 요소

<sequenceRule> 요소는 연속 패턴에 따라 규칙을 정의합니다. 연속 규칙은 여러 이벤트 선택자를 허용하는 유일한 규칙입니다. 또한 최소 두 개의 이벤트 선택자가 필요합니다.

속성

<sequenceRule>에는 다음과 같은 속성이 있습니다.

표 61. <sequenceRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 `false`입니다.	xsd:boolean	아니오
arrivalOrder	이벤트가 규칙에 대해 <eventSelector> 요소가 코딩된 순서대로 도착해야 하는지 여부를 정의합니다. 올바른 값은 다음과 같습니다. • `inOrder`(기본값) • `randomOrder`	xsd:string	아니오

arrivalOrder 속성값이 `randomOrder`이면, <eventSelector> 요소가 코딩된 순서가 중요합니다. 가장 제한적인 이벤트 선택 기준을 갖는 <eventSelector>요소는 보다 덜 제한적인 이벤트 선택 기준을 갖는 <eventSelector> 요소보다 먼저 코딩되어야 합니다. 그렇지 않으면, 순서가 감지되어야 할 때 감지되지 않습니다.

예를 들어, 다음과 같은 상황에 있다고 가정해 보겠습니다.

- 세 개의 <eventSelector> 요소가 정의되어 있습니다.
- 첫 번째 <eventSelector> 요소는 `eventA` 이벤트를 확인합니다.
- 두 번째 <eventSelector> 요소는 모든 이벤트를 확인합니다.
- 세 번째 <eventSelector> 요소는 `eventB` 이벤트를 확인합니다.

• eventA, eventB 및 eventC 이벤트가 지정된 시간 창 내에서 시스템에 제공됩니다.

규칙 작동이 다음과 같고 순서가 발견되어야 할 때 순서가 발견되지 않습니다.

1. 첫 번째 이벤트 eventA는 첫 번째 <eventSelector> 요소가 승인합니다.
2. 두 번째 이벤트 eventB는 두 번째 <eventSelector> 요소가 승인합니다.
3. 세 번째 이벤트 eventC는 무시됩니다.

예를 들어 다음과 같은 상황이 있습니다. <eventSelector> 요소는 올바르게 코딩되어 있고 가장 제한적인 이벤트 선택 기준이 덜 제한적인 이벤트 선택 기준보다 앞에 놓여 있습니다.

• 첫 번째 <eventSelector> 요소는 eventA 이벤트를 확인합니다.
• 두 번째 <eventSelector> 요소는 eventB 이벤트를 확인합니다.
• 세 번째 <eventSelector> 요소는 모든 이벤트를 확인합니다.

규칙 작동이 다음과 같고 순서가 발견됩니다.

1. 첫 번째 이벤트 eventA는 첫 번째 <eventSelector> 요소가 승인합니다.
2. 두 번째 이벤트 eventB는 두 번째 <eventSelector> 요소가 승인합니다.
3. 세 번째 이벤트 eventC는 세 번째 <eventSelector> 요소가 승인합니다.

포함 위치

<sequenceRule>은 다음 요소 내에 포함되어 있습니다.

• <ruleBlock>

포함 요소

<sequenceRule>은 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 62. <sequenceRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	연속 규칙에는 이 요소의 두 개의 발생이 필요합니다. 추가 발생이 허용됩니다.
<groupingKey>	선택적. 0 또는 1 발생이 허용됩니다.
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onDetection>	선택적. 0 또는 1 발생이 허용됩니다.
<onTimeOut>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

15 페이지의『연속 패턴』

연속 규칙은 연속 패턴에 의해 정의됩니다. 이는 이벤트의 특정 순서가 시간 간격 내에 도달하는지 여부를 감지합니다. 연속된 순서가 있거나 무작위일 수 있습니다. 연속 규칙은 상태 규칙입니다.

start 요소

<start> 요소는 특정 날짜의 특정 시간에 규칙이 활성화되는지와 규칙이 ACT(Active Correlation Technology) 엔진에 의해 로드될 때 활성화되는지 여부를 정의합니다.

세부사항

<start> 요소가 아예 코딩되어 있지 않으면, 기본 시작 시간은 <whenLoaded> 요소가 정의한 시간과 같습니다.

속성

<start>에는 속성이 없습니다.

포함 위치

<start>는 다음 요소 내에 포함되어 있습니다.

- <activationTime>

포함 요소

<start>는 다음 요소를 포함합니다.

표 63. <start> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<dateTime>	이러한 요소 중 하나가 필수이며 선택한 요소의 하나의 발생만이 허용됩니다.
<whenLoaded>	
<inactiveWhenLoaded>	

stop 요소

<stop> 요소는 특정 지속 기간이 지난 후에 특정 날짜의 특정 시간에 규칙이 비활성화되는지와 특정 시간에는 절대 비활성화되지 않는지 여부를 정의합니다.

세부사항

<stop> 요소가 아예 코딩되어 있지 않으면, 기본 중지 시간은 <never> 요소가 정의한 시간과 같습니다.

속성

<stop> 요소에는 속성이 없습니다.

포함 위치

<stop> 요소는 다음 요소 내에 포함되어 있습니다.

- <activationTime>

포함 요소

<stop> 요소는 다음 요소를 포함합니다.

표 64. <stop> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<dateTime>	이러한 요소 중 하나가 필수이며 선택한 요소의 하나의 발생만이 허용됩니다.
<never>	
<after>	

stopAfter 요소

<stopAfter> 요소는 <groupingKey> 요소가 정의한 대로 규칙 인스턴스가 활성이 된 후에 활성 상태로 남아 있을 지속 기간을 지정합니다. 이 지속 기간 후에는 규칙 인스턴스는 비활성화됩니다.

속성

<stopAfter>에는 다음과 같은 속성이 있습니다.

표 65. <stopAfter> 요소의 속성

이름	설명	데이터 유형	필수 여부
duration	지속 기간의 총 시간을 지정합니다. 이 속성의 데이터 유형은 단위 속성값에 따라 다릅니다.	• 단위 속성값이 ISO-8601이면 데이터 유형은 xsd:duration입니다. • 단위 속성값이 milliseconds이면 데이터 유형은 xsd:positiveInteger입니다.	예
unit	사용할 시간 단위를 지정합니다. 이 속성의 올바른 값은 다음과 같습니다. • ISO-8601 • milliseconds	xsd:string	예

시간 지속 기간에 ISO 8601 표준 사용

ISO-8601을 단위 속성값으로 코딩하면 지속 기간 속성값이 시간 지속 기간을 하나의 문자열로 지정하기 위해 ISO 8601 표준에 따라 코딩됩니다. 표준 XML 스키마 데이터 유형 스펙은 duration이라는 데이터 유형을 제공하기 위해 ISO 8601을 사용합니다. 이 데이터 유형은 http://www.w3.org/TR/xmlschema-2/#duration에 자세히 설명되어 있습니다.

표준 XML 스키마에서 duration 데이터 유형의 형식은 다음 문자열입니다.

P*n*Y*n*M*n*DT*n*H*n*M*n*S

- P는 항상 문자열을 시작하는 문자입니다.
- *n*Y는 연도를 표시합니다. 1년은 365일과 동일합니다. 그러므로 1Y를 코딩하면 365D를 코딩하는 것과 같습니다.
- *n*M은 개월 수를 표시합니다. 한달은 30일과 동일합니다. 그러므로 1M을 코딩하면 30D를 코딩하는 것과 같습니다.
- *n*D는 일 수를 표시합니다.
- T는 날짜 단위(연도, 월 및 일)를 시간 단위(시간, 분 및 초)와 구분하는 분리 문자입니다. 시간 단위는 항상 T 다음에 옵니다.
- *n*H는 시간을 표시합니다.
- *n*M은 분을 표시합니다.
- *n*S는 초를 표시합니다.

다음은 형식 예제입니다.

- P5DT12H는 5.5일입니다.
- PT59M59S는 59분 59초입니다.
- P1M은 1개월입니다.

포함 위치

<stopAfter>는 <activateOnEvent> 요소 내에 포함되어 있지만 <activateOnEvent>가 <activationByGroupingKey> 요소 내에서 코딩된 경우에만 포함됩니다.

포함 요소

<stopAfter>에 포함된 요소가 없습니다.

thresholdRule 요소

<thresholdRule> 요소는 임계값 패턴에 따라 규칙을 정의합니다.

속성

<thresholdRule>에는 다음과 같은 속성이 있습니다.

표 66. <thresholdRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예
processOnlyForwardedEvents	규칙이 모든 이벤트를 받을지 또는 다른 규칙에서 전달된 이벤트만을 받을지 여부를 결정합니다. 기본값은 규칙이 다른 규칙에서 전달된 이벤트를 포함하여 모든 이벤트를 받음을 표시하는 `false`입니다.	xsd:boolean	아니오

포함 위치

<thresholdRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<thresholdRule>은 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 67. <thresholdRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 0 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<eventSelector>	선택적. 0 또는 1 발생이 허용됩니다.
<groupingKey>	선택적. 0 또는 1 발생이 허용됩니다.
<booleanThreshold>	이러한 요소 중 하나가 필수이며 선택한 요소의 하나의 발생만이 허용됩니다.
<computedThreshold>	
<eventCountThreshold>	
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onDetection>	선택적. 0 또는 1 발생이 허용됩니다.
<onTimeOut>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

17 페이지의『임계값 패턴』
임계값 규칙은 임계값 패턴에 의해 정의됩니다. 이는 시간 간격 내에서 선택한 이벤트 그룹을 수집하고 각 이벤트를 받은 후에 임계값 조건에 부합하는지 여부를 판별합니다. 이는 상태 규칙입니다.

timeInterval 요소

<timeInterval> 요소는 시간 창의 지속 기간을 지정합니다.

속성

<timeInterval>에는 다음과 같은 속성이 있습니다.

표 68. <timeInterval> 요소의 속성

이름	설명	데이터 유형	필수 여부
duration	지속 기간의 총 시간을 지정합니다. 이 속성의 데이터 유형은 단위 속성값에 따라 다릅니다.	• 단위 속성값이 ISO-8601이면 데이터 유형은 xsd:duration입니다. • 단위 속성값이 milliseconds이면 데이터 유형은 xsd:positiveInteger입니다.	예
unit	사용할 시간 단위를 지정합니다. 이 속성의 올바른 값은 다음과 같습니다. • ISO-8601 • milliseconds	xsd:string	예

시간 지속 기간에 ISO 8601 표준 사용

ISO-8601을 단위 속성값으로 코딩하면 지속 기간 속성값이 시간 지속 기간을 하나의 문자열로 지정하기 위해 ISO 8601 표준에 따라 코딩됩니다. 표준 XML 스키마 데이터 유형 스펙은 duration이라는 데이터 유형을 제공하기 위해 ISO 8601을 사용합니다. 이 데이터 유형은 http://www.w3.org/TR/xmlschema-2/#duration에 자세히 설명되어 있습니다.

표준 XML 스키마에서 duration 데이터 유형의 형식은 다음 문자열입니다.

P*n*Y*n*M*n*DT*n*H*n*M*n*S

- P는 항상 문자열을 시작하는 문자입니다.
- *n*Y는 연도를 표시합니다. 1년은 365일과 동일합니다. 그러므로 1Y를 코딩하면 365D를 코딩하는 것과 같습니다.
- *n*M은 개월 수를 표시합니다. 한달은 30일과 동일합니다. 그러므로 1M을 코딩하면 30D를 코딩하는 것과 같습니다.
- *n*D는 일 수를 표시합니다.

- T는 날짜 단위(연도, 월 및 일)를 시간 단위(시간, 분 및 초)와 구분하는 분리 문자입니다. 시간 단위는 항상 T 다음에 옵니다.
- *n*H는 시간을 표시합니다.
- *n*M은 분을 표시합니다.
- *n*S는 초를 표시합니다.

다음은 형식 예제입니다.

- P5DT12H는 5.5일입니다.
- PT59M59S는 59분 59초입니다.
- P1M은 1개월입니다.

포함 위치

<timeInterval>은 다음 요소 내에 포함되어 있습니다.

- <timeWindow>

포함 요소

<timeInterval>에 포함된 요소가 없습니다.

timerRule 요소

<timerRule> 요소는 타이머 패턴에 따라 규칙을 정의합니다.

속성

<timerRule>에는 다음과 같은 속성이 있습니다.

표 69. <timerRule> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	규칙을 식별합니다. 이 ID는 이 규칙을 포함하는 규칙 블록 내에서 고유해야 합니다. 마침표를 포함할 수 없습니다.	xsd:NMTOKEN	예
processOnlyForwardedEvents	타이머 규칙이 이벤트를 처리하지 않으므로 이 속성은 무시됩니다.	xsd:boolean	아니오

표 69. <timerRule> 요소의 속성 (계속)

이름	설명	데이터 유형	필수 여부
repeat	타이머 규칙이 비활성화될 때까지 반복 실행되는지 여부를 정의합니다. 올바른 값은 다음과 같습니다. • `true`(기본값) • `false` 값을 `false`로 설정하면 규칙은 시간 간격 내에서 한 번만 실행되고 각 시간 창이 완료될 때 규칙 응답 조치를 실행하고 중지합니다. 타이머 규칙의 <timeWindow> 요소에 <runUntilDeactivated> 요소가 포함되어 있으면 repeat 속성은 무시됩니다.	xsd:boolean	아니오

포함 위치

<timerRule>은 다음 요소 내에 포함되어 있습니다.

- <ruleBlock>

포함 요소

<timerRule>은 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 70. <timerRule> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<variable>	선택적. 영 이상의 발생이 허용됩니다.
<activationInterval>	선택적. 0 또는 1 발생이 허용됩니다.
<lifeCycleActions>	선택적. 0 또는 1 발생이 허용됩니다.
<timeWindow>	필수. 1 발생만이 허용됩니다.
<onTimeWindowComplete>	선택적. 0 또는 1 발생이 허용됩니다.

관련 개념

20 페이지의『타이머 패턴』

타이머 규칙은 타이머 패턴에 의해 정의됩니다. 이는 정기적인 간격으로 조치를 시작합니다. 이는 상태 규칙입니다. 타이머 규칙은 이벤트를 처리하지 않지만 이벤트에 의해 활성화되거나 비활성화될 수 있습니다.

timeWindow 요소

<timeWindow> 요소에는 규칙이 처리 중인 동안 시간 간격을 정의하는 요소가 포함되어 있습니다.

세부사항

예를 들어 중복 규칙의 시간 창은 규칙이 승인된 첫 번째 이벤트의 중복인 이벤트를 확인해야 하는 기간을 정의합니다. 시간 창이 30초이면 중복 규칙은 승인한 첫 번째 이벤트의 30초 내에 발생한 모든 중복 이벤트를 처리합니다.

속성

<timeWindow>에는 속성이 없습니다.

포함 위치

<timeWindow>는 다음 요소 내에 포함되어 있습니다.

- <collectionRule>
- <computationRule>
- <duplicateRule>
- <sequenceRule>
- <thresholdRule>
- <timerRule>

포함 요소

<timeWindow>는 다음 요소를 포함합니다.

표 71. <timeWindow> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<timeInterval>	이러한 요소 중 하나가 필수이며 선택한 요소의 하나의 발생만이 허용됩니다.
<runUntilDeactivated>	

variable 요소

<variable> 요소는 변수를 정의하고 표현식이 참조할 수 있는 양식으로 된 정보를 포함합니다. 변수는 규칙 세트, 규칙 블록 또는 규칙 레벨에서 정의할 수 있습니다.

세부사항

규칙 세트 변수

규칙 세트에 전체적으로 적용되고 해당 규칙 세트의 모든 표현식에 의해 참조될 수 있습니다.

규칙 블록 변수

규칙 블록 내에서 및 포함된 규칙 블록 내에서만 적용되고 해당 규칙 블록의 모든 표현식에 의해 참조될 수 있습니다.

규칙 변수

해당 규칙 내의 표현식에 적용됩니다.

변수는 규칙 계층 구조의 다른 레벨에는 동일한 이름이 있을 수 있습니다. 변수에 액세스할 때에는 변수의 대부분의 로컬 정의가 사용됩니다. 예를 들어, 변수가 규칙 세트 레벨, 규칙 블록 레벨 및 규칙 레벨에서 동일한 이름으로 정의된 경우에는 규칙 레벨의 변수 정의가 해당 규칙 내의 표현식에 의해 사용됩니다.

변수가 규칙 세트 또는 규칙 블록 레벨에서 정의되어 있으면 여러 규칙이 서로 다른 시간에 이러한 변수를 가져오고 설정합니다. 그러므로 변수값을 정확하게 유지하려면 규칙 세트에서 변수 간 상호작용을 코딩하는 방법을 알아야 합니다.

변수가 규칙 세트 또는 규칙 블록 레벨에서 정의되어 있으면 규칙 패턴이 일치한 후에 다시 초기화되지 않습니다.

다음 조건에서 규칙 세트 및 규칙 블록 변수의 가져오기 및 설정에 대해 잠금을 사용하면 변수값이 잘못 설정되는 것을 막을 수 있습니다.

- 타이머 규칙은 <onTimeOut> 조치 동안에 변수를 가져오거나 설정하는 경우
- ACT(Active Correlation Technology) 엔진이 임베드된 어플리케이션이 멀티스레드인 경우

규칙이 그룹화 키로 정의되고, <variable> 요소에 의해 정의된 규칙 변수가 수명 주기 조치 내에서 <activationInterval> 요소 내에 포함된 <activateOnEvent> 또는 <deactivateOnEvent> 요소 내에 포함된 <filteringPredicate> 요소 내에서 올바르지 않는 경우 이 경우 원인은 규칙 변수가 규칙 인스턴스에만 적용되고, 규칙 인스턴스가 이러한 표현식 실행 당시 존재하지 않기 때문입니다.

속성

<variable>에는 다음과 같은 속성이 있습니다.

표 72. <variable> 요소의 속성

이름	설명	데이터 유형	필수 여부
name	특정 변수를 식별합니다. 변수는 이름으로 참조됩니다.	xsd:NMTOKEN	예
dataType	변수가 포함하는 정보의 유형을 식별합니다. 이는 java.lang.String과 같은 완전한 데이터 유형이어야 합니다.	xsd:NMTOKEN	예

변수의 이름 제한사항

변수 이름에는 특정 제한사항이 있습니다. 그러므로 <variable> 요소에서 이름 속성값에는 다음과 같은 제한사항이 있습니다.

- 다음과 같은 문자만을 포함할 수 있습니다.
  - 대문자 ASCII 라틴 문자 A-Z. 유니코드 표시는 ₩u0041-₩u005a입니다.
  - 소문자 ASCII 라틴 문자 a-z. 유니코드 표시는 ₩u0061-₩u007a입니다.
  - ASCII 밑줄(_). 유니코드 표시는 ₩u005f입니다.
  - 달러 부호($). 유니코드 표시는 ₩u0024입니다.
  - ASCII 숫자 0 – 9. 유니코드 표시는 ₩u0030-₩u0039입니다.
- 이름은 널(null)일 수 없습니다.
- 이름은 빈 문자열일 수 없습니다.
- 이름에는 공백을 포함할 수 없습니다.
- 이름에는 마침표를 포함할 수 없습니다.
- 모든 양식에서 act_로 시작할 수 없습니다(대문자, 소문자 또는 대소문자 혼합은 제외).

포함 위치

<variable>는 다음 요소 내에 포함되어 있습니다.

- <ruleSet>
- <ruleBlock>
- <collectionRule>
- <computationRule>
- <duplicateRule>
- <filterRule>
- <sequenceRule>
- <thresholdRule>
- <timerRule>

포함 요소

<variable>는 다음 요소를 포함합니다.

요소는 표시된 순서대로 코딩되어야 합니다. 요소가 선택적이면 코딩할 필요가 없지만 코딩된 모든 요소는 올바른 순서를 따라야 합니다.

표 73. <variable> 요소 내에 포함된 요소

요소	필수 또는 선택적 여부
<comment>	선택적. 0 또는 1 발생이 허용됩니다.
<varInitializer>	필수. 1 발생만이 허용됩니다.

관련 개념

26 페이지의『변수』

규칙 언어에서 특정 변수는 서로 다른 이벤트 발생이나 규칙에 이벤트 관련 정보를 저장하는 데 사용됩니다. 그러면 이 이벤트 관련 정보는 규칙 내의 표현식에서 액세스할 수 있습니다. 일부 변수 유형은 규칙 작성자가 정의하고 나머지 유형은 ACT(Active Correlation Technology)가 제공합니다. 일부 유형은 표현식 내에서 직접 액세스할 수 있으며 나머지 유형은 ACT(Active Correlation Technology)가 제공한 메소드를 통해서만 액세스할 수 있습니다.

varInitializer 요소

<varInitializer> 요소에는 연관된 <variable> 요소에 정의된 변수의 초기값을 제공하는 표현식이 포함되어 있습니다.

세부사항

변수는 어떤 유형이나 가능하므로 표현식 코드는 ACT(Active Correlation Technology) 엔진이 저장할 배열 오브젝트 또는 다른 복잡한 구현 특정 오브젝트를 리턴할 수 있습니다.

표현식에서 사용할 수 있는 변수에 대한 정보는 26 페이지의『변수』의 내용을 참조하십시오. 특정 변수의 사용 여부는 표현식의 문맥에 따라 다릅니다.

속성

<varInitializer>에는 다음 속성이 있습니다.

표 74. <varInitializer> 요소의 속성

이름	설명	데이터 유형	필수 여부
expressionLanguage	표현식을 작성하는 데 사용하는 프로그래밍 언어를 식별합니다. Java 프로그래밍 언어는 유일하게 지원되는 표현식 언어이며 이 속성에 올바른 유일한 값은 `java`입니다.	xsd:NMTOKEN	예

포함 위치

<varInitializer>는 다음 요소 내에 포함되어 있습니다.

- <variable>

포함 요소

<varInitializer>에 포함된 요소가 없습니다.

관련 개념

21 페이지의『표현식』

표현식은 규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 표현식은 ACT(Active Correlation Technology) 엔진의 외부에 있는 코드에도 액세스할 수 있습니다. 규칙 언어에서 표현식은 특정 컨텍스트 또는 규칙 언어 요소 내에서만 올바릅니다.

whenLoaded 요소

<whenLoaded> 요소는 ACT(Active Correlation Technology) 엔진이 규칙을 로드할 때 규칙이 활성화되도록 지정합니다.

속성

<whenLoaded>에는 속성이 없습니다.

포함 위치

<whenLoaded>는 다음 요소 내에 포함되어 있습니다.

- <start>

포함 요소

<whenLoaded>에 포함된 요소가 없습니다.

제 6 장 용어집

이 용어집에는 ACT(Active Correlation Technology)의 중요 개념에 대한 용어 및 정의가 있습니다.

가져오기(import)
외부 코드를 표현식에 액세스 가능하도록 만드는 프로그래밍 언어 특정 방법입니다.

계산 패턴(computation pattern)
시간 간격 내에서 각 이벤트를 받을 때 수집된 이벤트에 표현식을 통해 계산을 적용하는 규칙을 정의하는 규칙 패턴입니다. 계산 패턴에 의해 정의된 규칙은 상태 규칙입니다.

규칙 블록(rule block)
규칙 세트 내에서 규칙을 기능별로 도메인으로 그룹화하기 위한 조직 단위입니다. 규칙 블록은 규칙뿐만 아니라 다른 규칙 블록도 포함할 수 있습니다.

규칙 세트(rule set)
ACT(Active Correlation Technology) 규칙 언어의 규칙 실행 단위입니다. 규칙 세트에는 규칙 블록으로 구성되어 있고 ACT(Active Correlation Technology) 엔진에 의해 실행될 규칙이 포함되어 있습니다. 엔진은 지정된 시간에 하나의 규칙 세트에 대해서만 작동합니다.

규칙 응답 조치(rule response action)
조치를 참조하십시오.

규칙 응답(rule response)
ACT(Active Correlation Technology) 엔진이 규칙 조건에 부합한 것으로 인식할 때 실행되는 표현식입니다. 규칙 응답은 하나 이상의 조치로 구성되어 있습니다.

규칙 인스턴스(rule instance)
그룹화 키의 컨텍스트에서 규칙의 사본입니다.

규칙 패턴(rule pattern)
이벤트 연관 상황의 표시입니다(예: 임계값 조건 또는 중복 이벤트 감지). ACT(Active Correlation Technology) 규칙 언어에는 콜렉션, 계산, 중복, 필터, 연속, 임계값 및 타이머와 같은 규칙 패턴이 포함됩니다. 규칙의 패턴은 규칙에 의해 정의된 상황이 발생할 때 일치합니다. 패턴이 일치하면 규칙은 적절한 규칙 응답 조치를 수행하여 처리를 종결합니다. 규칙이 활성이면 규칙 패턴은 여러 번 일치할 수 있습니다.

규칙(rule)

이벤트 간의 관계를 인식하고 적절한 규칙 응답을 실행하는 데 사용되는 연관 단위입니다. 규칙은 일곱 개의 규칙 패턴 중 하나의 구현이며 기능에 따라 규칙 세트의 일부인 규칙 블록으로 구성되어 있습니다. 규칙은 이벤트가 이벤트 선택 기준에 부합하는 경우 이벤트를 처리하기 위해 승인합니다.

그룹화 키(grouping key)

규칙에게 공통 특성을 공유하는 각 이벤트 그룹마다 별도의 규칙 인스턴스(또는 사본)를 작성하도록 지시하는 방법입니다.

내부 이벤트(internal event)

ACT(Active Correlation Technology) 엔진에서 실행 중인 규칙이 작성한 이벤트입니다. 이 이벤트는 다른 규칙으로 전달될 수 있습니다.

노드(node)

규칙 세트 내에서 개별적이고 독립적으로 추가하거나, 제거하거나, 대체할 수 있는 규칙 계층 구조 내의 오브젝트입니다. 특히 다음과 같은 오브젝트가 노드입니다.

- 규칙
- 규칙 블록
- 규칙 블록 변수
- 규칙 세트 변수

오브젝트는 규칙 레벨 아래에서는 개별적이고 독립적으로 작동할 수 없으므로 규칙 변수는 노드가 아닙니다.

도메인(domain)

규칙 그룹에서 기능 적용에 기반이 되는 범주입니다. 예를 들어, 도메인은 특정 지리적 지역, IT 관리 규율(예: 보안 감지 또는 네트워크 이벤트 연관) 또는 비즈니스 조직(예: 특정 회사 또는 회사의 부서)를 표시할 수 있습니다.

상태 규칙(stateful rule)

특정 기간 동안 이벤트 콜렉션에 대해서 작동하려는 목적으로 규칙 인스턴스의 특성에 대한 정보와 같은 상태 정보를 보관하는 규칙입니다. 콜렉션, 계산, 중복, 연속, 임계값 또는 타이머와 같은 규칙 패턴에 의해 정의된 규칙은 상태 규칙입니다.

상태가 없는 규칙(stateless rule)

상태 정보를 보관하지 않으므로 한 번에 한 이벤트에 대해서만 작동하는 규칙입니다. 필터 패턴에 의해 정의된 규칙은 상태가 없는 규칙입니다.

선언문(predicate)

필터링 선언문을 참조하십시오.

수명 주기 조치(life cycle action)

규칙이 로드되고, 로드 해제되고, 활성화되거나 비활성화될 때 실행되는 표현식입니다.

연속 패턴(sequence pattern)

시간 간격 내에 이벤트의 특정 순서의 존재나 부재를 감지하는 규칙을 정의하는 규칙 패턴입니다. 연속된 순서가 있거나 무작위일 수 있습니다. 연속 패턴에 의해 정의된 규칙은 상태 규칙입니다.

외부 오브젝트(external object)

어플리케이션과 표현식과의 통신을 위해 작성된 오브젝트입니다.

외부 이벤트(external event)

ACT(Active Correlation Technology) 엔진이 외부의 소스로부터 받는 이벤트입니다.

응답(response)

규칙 응답을 참조하십시오.

이벤트 선택자(event selector)

이벤트 선택 기준입니다. 규칙이 처리를 위해 어떤 이벤트를 승인할지를 판별하는 기준입니다. 이벤트 선택자에는 이벤트 유형과 필터링 선언문이 포함됩니다.

이벤트 제공자(event provider)

ACT(Active Correlation Technology)에서 처리하는 이벤트를 생성하는 모든 소프트웨어입니다.

임계값 패턴(threshold pattern)

시간 간격 내에서 선택한 이벤트 그룹을 수집하고, 각 이벤트를 받은 후에는 임계값 조건에 부합하는지 여부를 판별하는 규칙을 정의하는 규칙 패턴입니다. 임계값 패턴에 의해 정의된 규칙은 상태 규칙입니다.

조각(snippet)

소스 코드의 인용구입니다.

조치(action)

규칙 응답의 일부로서 실행되거나 규칙이 로드되고, 로드 해제되고, 활성화되거나 비활성화될 때 실행되는 표현식입니다.

중복 패턴(duplicate pattern)

지정된 시간 간격 내에서 승인되는 두 번째와 후속 이벤트의 수를 세어 이러한 이벤트의 규칙 세트 처리를 건너뛰는 규칙을 정의하는 규칙 패턴입니다. 중복 패턴에 의해 정의된 규칙은 상태 규칙입니다.

콜렉션 패턴

시간 간격 내에서 선택한 이벤트 그룹을 수집하는 규칙을 정의하는 규칙 패턴입니다. 콜렉션 패턴에 의해 정의된 규칙은 상태 규칙입니다.

타이머 패턴(timer pattern)

정기적으로 조치를 시작하는 규칙을 정의하는 규칙 패턴입니다. 타이머 패턴에 의해 정의된 규칙은 상태 규칙입니다. 타이머 규칙은 이벤트를 처리하지 않지만 이벤트에 의해 활성화되거나 비활성화될 수 있습니다.

표현식 언어(expression language)

표현식을 작성하는 데 사용하는 프로그래밍 언어입니다.

표현식(expression)

규칙에 추가할 수 있는 사용자 정의 논리가 포함되어 있는 코드입니다. 규칙 작성자는 변수의 초기화, 이벤트 선택 기준 정의 또는 규칙 응답 조치 및 수명 주기 조치의 스펙과 같은 여러 용도로 표현식을 사용할 수 있습니다.

필터 패턴(filter pattern)

규칙이 이벤트를 승인할 때 특정 조치를 취하는 규칙을 정의하는 규칙 패턴입니다. 필터 패턴에 의해 정의된 규칙은 단일 이벤트에 대해서만 작동하므로 상태가 없는 규칙입니다.

필터링 선언문(filtering predicate)

규칙이 이벤트를 처리하기 위해 승인하는 조건을 정의하는 표현식입니다. 필터링 선언문은 이벤트 선택자의 파트입니다. 필터링 선언문은 부울값을 리턴합니다.

ACT ACT(Active Correlation Technology)를 참조하십시오.

ACT(Active Correlation Technology)

규칙을 사용하여 이벤트 연관을 제공하는 IBM 기술입니다.

ACT(Active Correlation Technology) 규칙 빌더

ACT(Active Correlation Technology) 규칙 언어로 된 연관 규칙을 작성하기 위한 GUI입니다.

ACT(Active Correlation Technology) 규칙 언어

이벤트를 연관시키기 위해 규칙을 작성하는 XML 기반 언어입니다. 그런 다음 이러한 규칙을 ACT(Active Correlation Technology) 런타임 환경으로 전개할 수 있습니다.

ACT(Active Correlation Technology) 런타임 환경

ACT(Active Correlation Technology) 엔진이 임베드된 어플리케이션으로 이는 컴파일러를 포함하거나 포함하지 않습니다.

ACT(Active Correlation Technology) 엔진

ACT(Active Correlation Technology) 컴파일러의 출력에 따라 이벤트를 처리하는 ACT(Active Correlation Technology) 구성요소입니다.

ACT(Active Correlation Technology) 컴파일러

규칙 세트 및 ACT(Active Correlation Technology) 엔진이 필요로 하는 내

부 데이터 구조를 생성하기 위해 여기에 포함된 코드를 구문 분석하는 ACT(Active Correlation Technology) 구성요소입니다.

부록. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-270
135-270
군인공제회관빌딩
한국 아이.비.엠 주식회사

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다.

IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 명시적 또는 묵시적인 일체의 보증 없이 이 책을 ″현상태대로″ 제공합니다.

일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12
군인공제회관빌딩
한국 아이.비.엠 주식회사

이러한 정보는 해당 조건(예를 들어, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

상표

DB2, IBM, IBM 로고, Tivoli, Tivoli 로고, Tivoli Enterprise Console 및 WebSphere는 미국 또는 기타 국가에서 사용되는 International Business Machines Corporation의 등록상표입니다.

Java 및 모든 Java 기반 상표 및 로고는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

Printed in Korea