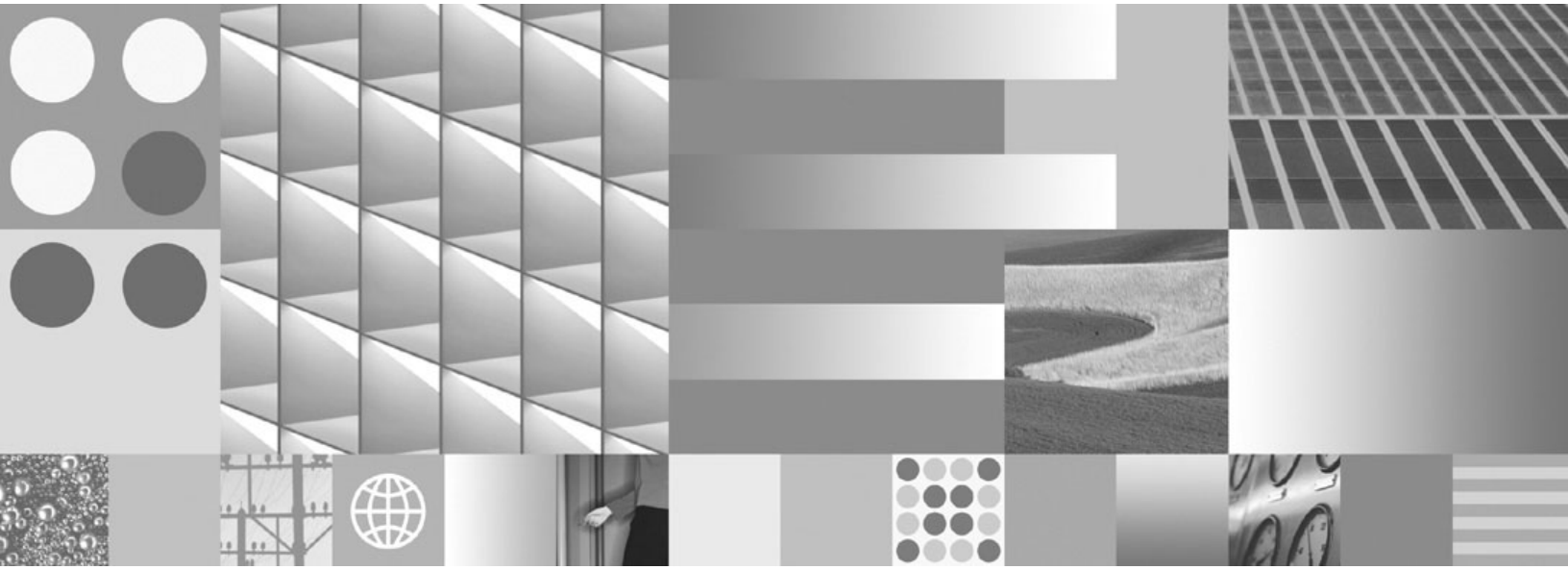
Active Correlation Technology

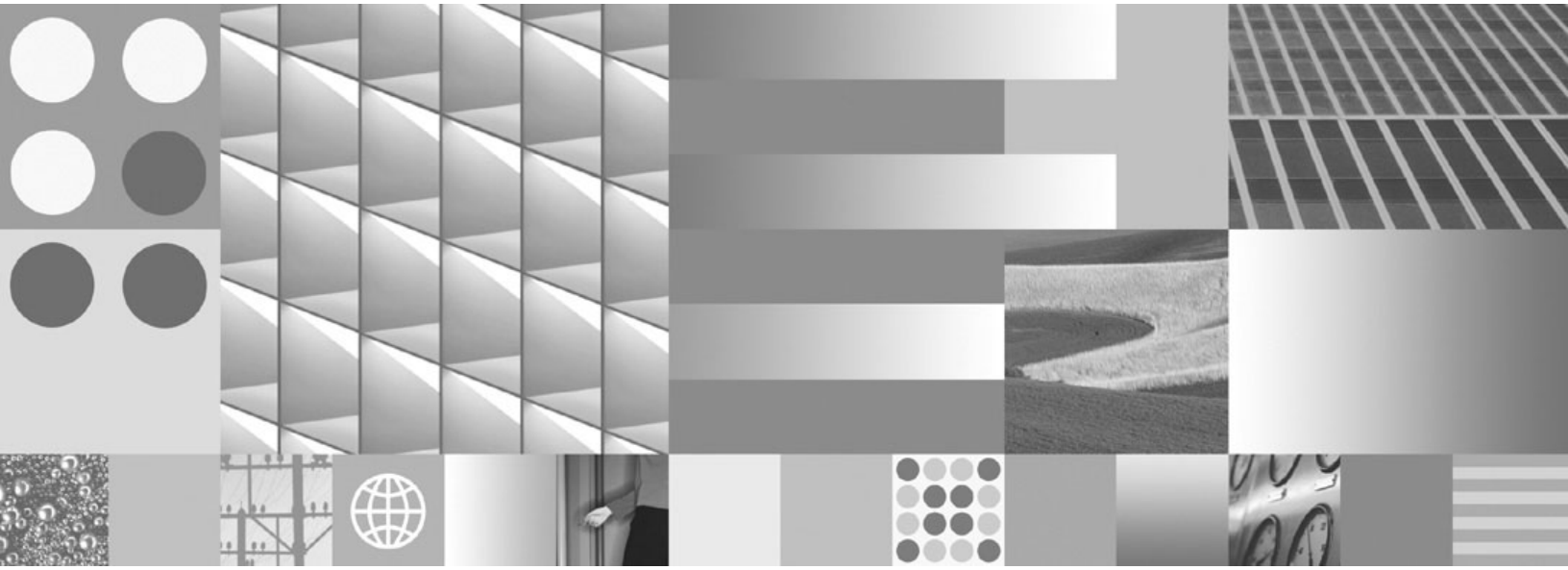
バージョン 6.0



ルール・ライターのガイドとリファレンス

Active Correlation Technology

バージョン 6.0



ルール・ライターのガイドとリファレンス

注

本書および本書で紹介する製品をご使用になる前に、 131 ページの『特記事項』に記載されている情報をお読みください。

第 1 刷 (2006.6)

この情報について

この情報では、IBM® Active Correlation Technology と複合イベント処理におけるその役割について概説します。Active Correlation Technology ルール言語は、イベントを相関させるルールを記述するための XML ベースの言語です。この情報は、それぞれのビジネス組織のために、イベントを相関させるルールの記述方法を理解する必要のあるルール・ライター向けに提供されています。

目次

この情報について iii

第 1 部 ルール・ライターのガイド . . 1

第 1 章 概要 3

第 2 章 ルール言語の概要 5
ルールの構造 5
ルールのライフ・サイクル 7
ルールの編成 9
ルール・パターン 11
コレクション・パターン 12
計算パターン 12
重複パターン 13
フィルター・パターン 14
シーケンス・パターン 14
しきい値パターン 17
タイマー・パターン 19
さまざまなルール・パターンの共通の性質および固有の性質 20
式 21
外部モジュールおよびオブジェクトのインポートとアクセス 22
変数の初期化とアクセス 23
イベント関連情報へのアクセス 24
式のコーディングのベスト・プラクティス . . . 25
変数 26
Active Correlation Technology 変数のデータ・タイプ 27
変数が有効となる式のコンテキスト 28
act_event 変数 28
act_eventCount 変数 29
act_eventList 変数 30
act_lib 変数 31
act_location 変数 32
act_nodeName 変数 33
act_threshold 変数 34
ルール・セットのイベント・フロー 35

第 3 章 ルール記述の概要 37
イベント相関の計画 37
イベントを相関させるルールの設計 40
ルール・ビルダー入門 40
Eclipse ワークベンチでのパースペクティブの設定 41
プリファレンスの設定 41
ルール・セット・ファイルを保管するためのプロジェクトの作成 42
ルール・セットの作成 42
ルール・ブロックの作成 43
ルールの作成 44
ルール・セットの検証 44
ルール・セットのコンパイル 44
ルール・セットの更新 45
ルール内の式にスニペットを組み込む 45

第 2 部 ルール・ライターのリファレンス 47

第 4 章 ルール・セット編成のサマリー 49
ルール・セットのサマリー 49
ルール・ブロックのサマリー 50
コレクション・ルールのサマリー 51
計算ルールのサマリー 52
重複ルールのサマリー 54
フィルター・ルールのサマリー 55
シーケンス・ルールのサマリー 56
しきい値ルールのサマリー 58
タイマー・ルールのサマリー 59

第 5 章 言語エレメントのリファレンス 61
action エレメント 62
activateOnEvent エレメント 63
activationByGroupingKey エレメント 64
activationInterval エレメント 71
activationTime エレメント 74
after エレメント 74
attributeAlias エレメント 76
attributeName エレメント 77
booleanThreshold エレメント 77
collectionRule エレメント 78
comment エレメント 79
computationRule エレメント 80
computedThreshold エレメント 81
computedValue エレメント 83
computeFunction エレメント 84
dateTime エレメント 85
deactivateOnEvent エレメント 86
duplicateRule エレメント 87
eventAttribute エレメント 88
eventCountThreshold エレメント 89
eventSelector エレメント 92
eventType エレメント 93
filteringPredicate エレメント 94
filterRule エレメント 95
groupingKey エレメント 96
import エレメント 99
inactiveWhenLoaded エレメント 100
lifeCycleActions エレメント 100
never エレメント 101
onActivation エレメント 101

onDeactivation エレメント 102
onDetection エレメント 103
onLoad エレメント 104
onNextEvent エレメント 104
onTimeOut エレメント 105
onTimeWindowComplete エレメント 106
onUnload エレメント 107
ruleBlock エレメント 107
ruleSet エレメント 108
runUntilDeactivated エレメント 109
sequenceRule エレメント 111
start エレメント 113
stop エレメント 114
stopAfter エレメント 115
thresholdRule エレメント 116
timeInterval エレメント 117
timerRule エレメント 119
timeWindow エレメント 120
variable エレメント 121
varInitializer エレメント 123
whenLoaded エレメント 124

第 6 章 用語集 125

付録. 特記事項 131
商標 132

第 1 部 ルール・ライターのガイド

第 1 章 概要

この概要では、複合イベント処理 (CEP とも呼ばれます) について簡単に説明し、Active Correlation Technology と複合イベント処理におけるその役割について概説します。

今日のビジネス環境

今日、商業組織および政府組織は、コンピューター・ネットワーク、とりわけインターネットを介した電子情報処理に依存しています。これらの組織は、グリッド・コンピューティングなどの付加的なテクノロジーを使用して、主幹業務アプリケーションを世界中のいたるところで休むことなく稼働させています。ビジネス・プロセス、アクティビティー、およびインフラストラクチャー、さらにその結果として我々のグローバル社会全体が、組織の情報技術 (IT) 層に依存しています。

組織は、そのビジネスにおいて何が起こっているかを常に把握している必要があります。例えば、主幹業務アプリケーションが使用可能であり、正しく機能しているかどうか、そしてビジネス・プロセス、アクティビティー、またはインフラストラクチャーにおける潜在的な危機を検出および回避する方法を把握している必要があります。危機が発生した場合は、その問題自体に加え、その修正方法、およびその原因を即時に理解する必要があります。

ビジネス・プロセス、アクティビティー、およびインフラストラクチャーに関連するほとんどのイベントの重要性は、情報量があまりに膨大であり、それが個別の関連性のない部分に分かれて存在しており、要約も非常に難しいため、認識または理解されることがありません。しかし、イベントは、その関係が容易に理解できるように集約され、相関されれば、豊富な情報を生み出す可能性があります。

複合イベント処理の目的は、イベントに関するよりよい情報をリアルタイムで入手することです。

複合イベント処理

イベントは、発生した事柄についての単なる通知です。

*複合イベント処理*は、イベント・ドリブン・システムにおける低レベル・イベントの分析、相関、および要約から高レベル・イベントを導出することです。これらの高レベル・イベント (複合イベントと呼びます) は、ユーザーに対して、理解しやすい用語でビジネス機会や問題について通知したり、自動化処理をトリガーしたりするのに適しています。これにより、組織は潜在的な機会または問題について早期に警告を受け、またビジネス・プロセス、アクティビティー、あるいはインフラストラクチャーの状況の変化の根本原因をよりよく理解できるようになるため、より効率的に活動することができます。

*イベント相関*は、イベント・ストリームにおけるパターンをリアルタイムで定義および検出し、関連するイベントに応答してアクションをインプリメントするプロセスです。検出された症状に基づいて問題を識別するために使用されます。イベントは、原因、時間、メンバーシップ、またはこれらの組み合わせによって相関させる

ことができます。イベント相関は、複合イベント処理に不可欠な部分です。

Active Correlation Technology

Active Correlation Technology は、ルールを使用して、イベント・ストリームにおけるパターンをリアルタイムで検出します。このテクノロジーは、多くの場合応答アクションは、単一の低レベル・イベントによってトリガーされるべきではなく、異なる時間に異なるコンテキストで発生しているイベントの複合的な構成によってトリガーされるべきであるという理解に基づいています。Active Correlation Technology は、イベント間の関係を利用して、ビジネス機会や問題の認識を可能にします。例えば、イベントの相関を通してリアルタイムで得たビジネス認識に基づいて、組織は以下のようなアクションを実行することができます。

- 休日特別セール期間中に、一部またはすべてのカスタマーに対して配送料の割引サービスを提供する。
- 次の 30 日間について、運送会社、注文金額、および注文量に基づいて配送コストを計算する。
- 2005 年 7 月 1 日から 2005 年 12 月 31 日までの期間中に 500 ドルを超える商品を購入したカスタマーに、25 ドル分の商品券を送付する。
- 36 時間以内に完了しない注文処理があった場合に、管理者に通知する。
- 30 秒間以内に同じコンピューターに対する 4 回を超えるログイン試行が検出された場合に、管理者に通知する。

Active Correlation Technology は、以下の基本項目で構成されています。

Active Correlation Technology ルール言語 (Active Correlation Technology rule language)
: イベントを相関させるルールを記述するための XML ベースの言語。これらのルールは、作成後、Active Correlation Technology ランタイム環境にデプロイできます。

Active Correlation Technology エンジン (Active Correlation Technology engine)
: Active Correlation Technology コンパイラーの出力に従ってイベントを処理する Active Correlation Technology コンポーネント。

Active Correlation Technology ルール・ビルダー (Active Correlation Technology rule builder)
: Active Correlation Technology ルール言語で相関ルールを記述するための GUI。

Active Correlation Technology ランタイム環境は、 Active Correlation Technology エンジンが組み込まれるアプリケーションです。

第 2 章 ルール言語の概要

この概要では、Active Correlation Technology ルール言語の主要概念について説明します。

ルール・パターンは、イベント相関状態 (しきい値条件または重複イベントの検出など) を表現したものです。Active Correlation Technology ルール言語には、IBM のお客様が対処する必要のあるほとんどのイベント相関状態を表すことが確認されている 7 つのルール・パターンが含まれています。7 つのルール・パターンのうち 6 つはステートフル・ルールを定義し、 1 つはステートレス・ルールを定義します。

ステートフル・ルールは、特定の期間に発生する複数のイベントを相関させ、それらのイベントに対する応答を生成します。ステートレス・ルールは、特定の条件を満たす単一イベントのみを処理し、そのイベントに対する応答を生成します。

ステートフル・ルール (stateful rule)
状態情報を保持するルール。状態情報は、一定期間にわたりイベントのコレクションに対して動作するための、ルール・インスタンスの特性に関する情報です。コレクション、計算、重複、シーケンス、しきい値、またはタイマーの各ルール・パターンによって定義されるルールは、ステートフル・ルールです。

ステートレス・ルール (stateless rule)
状態情報を保持せず、そのため一度に 1 つのイベントに対してのみ動作できるルール。フィルター・パターンによって定義されるルールは、ステートレス・ルールです。

関連するリファレンス

49 ページの『第 4 章 ルール・セット編成のサマリー』
このリファレンスでは、ルール・セット、ルール・ブロック、および各タイプのルールのすべての言語エレメントをリストします。 ルール・セットをコーディングするためのクイック・リファレンスとして利用できます。

61 ページの『第 5 章 言語エレメントのリファレンス』
このリファレンスでは、Active Correlation Technology ルール言語の XML スキーマにおける言語エレメントの詳細を説明します。言語エレメントはアルファベット順にリストされています。各エレメントで使用可能な属性については、そのエレメントに関するトピック内で説明されています。

125 ページの『第 6 章 用語集』
この用語集には、Active Correlation Technology の重要な概念に関する 用語および定義が含まれています。

ルールの構造

ルールの最も基本的な部分は、イベント選択、グループ化キー、ステートフル・ルールの時間枠、ルール応答、アクティブ化間隔、およびライフ・サイクル・アクションです。ルールには、式および変数も含まれます。式とは、ルールに追加できるカスタム・ロジックを含むコードです。

イベント選択

イベント選択基準は、ルールで処理するために受け入れるイベントを判別します。<eventSelector> エレメントは、ルールのイベント選択基準を定義します。イベント選択は、タイマー・パターンで定義されるルールを除くすべてのルールに適用されます。タイマー・ルールではイベントを処理しないため、イベント選択基準が組み込まれていません。

グループ化キー

通常、各アクティブ・ルールには、Active Correlation Technology エンジン内で実行中の 1 つのルール・インスタンス (またはコピー) が存在します。ただし、多くの場合異なるリソースのグループに関連している異なるイベントのグループに、同じルールが必要な場合があります。グループ化キーは、ルールに対して、共通の特性を共有する各イベント・グループに別個のルール・インスタンス (またはルール自体のコピー) を作成させるメソッドです。

グループ化キーは、イベント選択の付加的な形式として機能します。ルールがグループ化キーを使用して定義され、そのルールがグループ化キーによって定義される特性を持つイベントを受信した場合、そのイベントはその特性を共有するイベントを処理しているルール・インスタンスに送信されます。例えば、`Audit Failure` タイプのすべてのセキュリティー・イベントを収集するルールを定義し、イベントのホスト名属性をグループ化キーとして定義できます。これにより、ホスト名属性の固有値ごとにルールの個別のコピーを実行させることで、ルールを複数回使用できるようになります。また、`Audit Failure` イベントを受信するすべてのシステムをモニターして、各ホスト名について、2 分という期間内にそのイベントが 10 個より多く発生するかどうか判別することができます。

<groupingKey> エレメントは、ルールのグループ化キーを定義します。また、コレクション、計算、重複、シーケンス、およびしきい値の各パターンによって定義されるルールに対して有効です。

ステートフル・ルールの時間枠

ステートフル・ルールは特定の期間に発生する複数のイベントを相関させるため、ステートフル・ルールの基本的な部分は <timeWindow> エレメントで定義される時間枠です。時間枠は、ステートフル・ルールがそのパターンと突き合わせる処理を行う期間を指定します。

ルール応答

ルール応答アクションは、ルールがその処理を完了したときに実行されるアクションを定義します。以下の各言語エレメントは、それぞれ異なるタイプのルール応答アクションを定義します。

- <onDetection> 内の <action>
- <onNextEvent> 内の <action>
- <onTimeOut> 内の <action>
- <onTimeWindowComplete> 内の <action>

ルールで使用可能なルール応答アクションのタイプは、ルール・パターンによって異なります。

アクティブ化間隔

アクティブ化間隔は、ルールがアクティブ化および非アクティブ化されるタイミングを定義します。<activationInterval> エレメントはルールのアクティブ化間隔を定義します。

ルールは、離散的な時点で、または特定のイベントによってアクティブ化または非アクティブ化できます。

ルールが離散的な時点および 特定のイベントでアクティブ化または非アクティブ化されるよう指定した場合、離散的な時点またはイベントを受信した時点のいずれか早い時点でルールがアクティブ化または非アクティブ化されます。ただしこの場合、ルールはそのライフ・サイクルをとおして、多くのイベントによってアクティブ化または非アクティブ化される可能性があります。例えば、ルールがイベントによってアクティブ化され、非アクティブ化され、定義された時点でアクティブ化され、再度非アクティブ化され、別のイベントによってアクティブ化されることもあります。

<activationByGroupingKey> エレメントは、<activationInterval> エレメントに含まれるエレメントの 1 つです。<activationByGroupingKey> エレメントには、<groupingKey> エレメントで定義されているルール・インスタンスをアクティブ化および非アクティブ化できるイベントを指定するエレメントが含まれます。

ライフ・サイクル・アクション

ライフ・サイクル・アクションは、ルールのライフ・サイクルの 4 つの基本ステージ (ロード、アクティブ化、非アクティブ化、およびアンロード) で実行するアクションを定義します。

<lifeCycleActions> エレメントには、これらのアクションを定義する以下のエレメントが含まれます。

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

ルールのライフ・サイクル

ルールのライフ・サイクル内の各ステージには、複数の原因と結果がある可能性があります。式を記述し、(<lifeCycleActions> エレメントによって定義されるように) ライフ・サイクル・アクションにその式を含めることにより、ルール・ライターは、各ステージで実行するアクションを定義できます。

ルールのライフ・サイクル内のステージ

以下に、ルールのライフ・サイクル内の 4 つの基本ステージを示します。

ロード 実行中の Active Correlation Technology エンジンへのルールのロード。これにより、<onLoad> エレメント内のアクションがトリガーされます。

アクティブ化
ルールのアクティブ化。これにより、<onActivation> エレメント内のアクションがトリガーされます。

非アクティブ化
ルールの非アクティブ化。これにより、<onDeactivation> エレメント内のアクションがトリガーされます。

アンロード
実行中の Active Correlation Technology エンジンからのルールのアンロード。これにより、<onUnload> エレメント内のアクションがトリガーされます。

アクティブ化および非アクティブ化ステージは、1 つのルールのライフ・サイクルにおいて複数回発生する可能性がありますが、ロードおよびアンロード・ステージは 1 回のみ発生します。

通常、ライフ・サイクル・アクションを定義する必要はありません。特定のライフ・サイクル・アクションを定義する必要がある場合の例を、以下に示します。

- 特定のルールのロード時に、そのルール内でアクセスする必要のある外部システム (データベース・マネージャーなど) への接続を作成する必要がある場合があります。この同じルールのアンロード時に、接続を除去し、必要なクリーンアップ処理を実行する必要がある場合があります。
- 特定のルールのアクティブ化時に、そのルールで特定のリソースが使用可能であることを検証する必要がある場合があります。
- しきい値ルールが、しきい値にまだ達していない状態で非アクティブ化され、指定された期間がまだ終了していない場合、特定のユーザーにこの情報を含むメッセージを転送する必要があるかもしれません。

ルールのアクティブ化および非アクティブ化はライフ・サイクルにおいて多数回発生する可能性があるため、これらのステージについてコーディングするアクションは頻繁に実行される可能性があります。

各ライフ・サイクル・ステージの原因と結果

表 1 に、各ライフ・サイクル・ステージの原因と結果をリストします。

表 1. 各ライフ・サイクル・ステージの原因と結果

ライフ・サイクル・ステージ	原因	結果
ロード	以下のいずれかの状況: • ルールまたはルール・ブロックが追加または置き換えられ、その結果新規ルール (1 つ以上) がロードされた。 • Active Correlation Technology エンジンでルール・セットが置き換えられ、その結果新規ルール・セット内のルールがロードされた。	<onLoad> エレメント内のアクションが実行される。

表 1. 各ライフ・サイクル・ステージの原因と結果 (続き)

ライフ・サイクル・ステージ	原因	結果
アクティブ化	ルールはアクティブ化されます。ルールは、以下のいずれかの方法でアクティブ化できます。 • <activationInterval> エレメント内の定義に従う • act_lib 変数を介して使用可能な activate() メソッドを使用する • Active Correlation Technology エンジン内の activate() メソッドのアプリケーション呼び出しを使用する	ルールが非アクティブである場合、<onActivation> エレメント内のアクションが実行されます。
非アクティブ化	ルールは非アクティブ化されます。ルールは、以下のいずれかの方法で非アクティブ化できます。 • <activationInterval> エレメント内の定義に従う。例外として、<activationByGroupingKey> エレメント内で <deactivateOnEvent> エレメントが指定されている場合、ルールの非アクティブ化は発生しません。 • act_lib 変数を介して使用可能な deactivate() メソッドを使用する • Active Correlation Technology エンジン内の deactivate() メソッドのアプリケーション呼び出しを使用する	ルールがアクティブな場合、<onDeactivation> エレメント内のアクションが実行されます。
アンロード	以下のいずれかの状況: • Active Correlation Technology エンジンがシャットダウンし、その結果ルールがアンロードされた。 • ルールまたはルール・ブロックが除去または置き換えられ、その結果古いルール (1 つ以上) がアンロードされた。 • Active Correlation Technology エンジンでルール・セットが除去または置き換えられ、その結果古いルール・セット内のルールがアンロードされた。	ルールがアクティブな場合、<onDeactivation> エレメント内のアクションが実行され、その後 <onUnload> エレメント内のアクションが実行されます。そうでない場合、<onUnload> エレメント内のアクションのみが実行されます。

ルールの編成

Active Correlation Technology ルール言語は、ルールをルール・セットの一部であるルール・ブロックに編成します。

ルール・セット

ルール・セットには、Active Correlation Technology エンジンによって実行される、ルール・ブロックに編成されたルールが含まれます。これはルールの実行単位です。各 Active Correlation Technology エンジンは、一度に 1 つのルール・セットについてのみ動作します。

ルール・セットに含まれるルールは、 Active Correlation Technology エンジンに送信されるイベントによってトリガーされます。イベントは、各ルールのイベント選択基準に基づいて適切なルールに順次配送され、一度に 1 つのルールが実行されます。同じイベントを複数のルールに適用 (その結果それらのルールをトリガー) することができます。これらのルールは必ずしも関連しているとは限りませんが、関連している場合もあります。

ルール・セット内のルール・ブロックおよびルールの順序により、ルール・セットにおいてイベントがどのようにフローするかが決まります。

ルール・セットの有効範囲内の式 (カスタム・ロジックを含むコード) で使用するために、ルール・セット・レベルで変数およびインポートを定義できます。インポートは、外部コードにアクセスするためのプログラミング言語特有の方法です。ルール・ライターは、インポートを定義して、ルール内の式で使用するために外部モジュール (Java™ クラスなど) をインポートすることができます。

ルール・ブロック

ルール・ブロックは、ルールを機能に応じてルール・セット内のドメインにグループ化するための編成単位です。ドメインは、ルールのグループをその機能に基づいて適用するカテゴリーです。例えば、ドメインは、特定の地理上の区域、IT 管理規律 (セキュリティー検出またはネットワーク・イベント相関など)、またはビジネス組織 (特定の会社または会社の部門) を表すことができます。

ルール・ブロックには、ルールおよび他のルール・ブロックを含めることができます。ルール・ブロックはネストできるため、ルールの階層を構成できます。例えば、ルール・セットにネットワーク・イベント相関のためのルール・ブロックを含めることができ、ネットワーク・イベント相関のためのルール・ブロックには、他の 2 つのルール・ブロック (1 つは第 2 層の相関、もう 1 つは IP の相関用) を含めることができます。

したがって、ルール・セットは、さまざまなドメインにイベント相関機能を提供し、ルール・ブロックは、類似したイベントのセットにアクセスする必要がある可能性のある、これらのさまざまなドメインの編成を提供します。

ルール・ブロックの有効範囲内の式で使用するために、ルール・ブロック・レベルで変数およびインポートを定義できます。ルール・ブロックの有効範囲には、ルール・ブロックに含まれるルールおよびルール・ブロックが含まれます。

ルール

ルールは、イベント間の関係を認識し、適切なルール応答を実行するために使用される相関単位です。ルールは、以下の 7 つのルール・パターンのうち 1 つのインプリメンテーションで、その機能に応じて、ルール・セットの一部であるルール・ブロックに編成されます。

- コレクション・パターン
- 計算パターン
- 重複パターン
- フィルター・パターン
- シーケンス・パターン
- しきい値パターン
- タイマー・パターン

各ルールは、そのパターンに従って固有のイベント相関機能を提供でき、ルールはイベント転送を介してチェーニングできます。このルールのチェーニングによって、さまざまなパターンのイベント相関機能を結合またはネストできます。

ルールの有効範囲内の式で使用するために、ルール・レベルで変数を定義できます。

サマリー

要約すると、ルール・セットは実行単位、ルール・ブロックは編成単位、そしてルールは相関単位です。ルール・セットには 1 つ以上のルール・ブロックが含まれ、各ルール・ブロックには追加のルール・ブロックを含めることができます。各ルール・ブロックには、特定のドメインについてのルールが含まれます。ルール・ブロックをネストさせて、ルールの階層を構成することができます。ルール・セット内のルール・ブロックおよびルールの順序により、ルール・セットにおいてイベントがどのようにフローするかが決まります。

ルール内の式で使用するために、ルール・セット・レベルまたはルール・ブロック・レベルで変数およびインポートを定義できます。変数またはインポートの有効範囲は、それぞれのルール・セットまたはルール・ブロックです。変数はルール・レベルでも定義できますが、この場合その有効範囲はルールのみに制限されます。

ルール・パターン

ルール・パターンは、イベント相関状態 (しきい値条件または重複イベントの検出など) を表現したものです。Active Correlation Technology ルール言語では、コレクション、計算、重複、フィルター、シーケンス、しきい値、およびタイマーというルール・パターンが定義されています。

ルールのパターンは、ルールによって定義された状態が発生したときに一致したことになります。パターンが一致した場合、ルールは適切なルール応答アクションを実行することによりその処理を完了します。ルールがアクティブである間、ルール・パターンは複数回一致することがあります。

フィルター・パターンによって定義されるルールは、ルール言語の中で唯一のステートレス・ルールです。その他のすべてのルールはステートフルです。

コレクション・パターン

コレクション・ルールは、コレクション・パターンによって定義されます。このルールは時間間隔内に選択されたイベントのグループを収集します。これはステートフル・ルールです。

概説

コレクション・パターンは、一定期間にわたり類似したイベントを収集するために使用します。期間は、ルール言語の <timeWindow> エレメントによって定義された、必須の時間枠で指定されます。

ルール応答が実行される条件

コレクション・パターンを使用すると、<onTimeWindowComplete> エレメントで定義されているように、時間枠が完了したときにルール応答が実行されます。

このルール・パターンの使用例

コレクション・パターンの使用例は、以下を行うルールです。

期間内に特定のイベント・セレクターの基準に一致するイベントを収集します。期間が終了すると、収集されたイベントは単一イベントに要約されます。このイベントには、要約されたイベントの総数と特性情報が含まれます。

関連資料

51 ページの『コレクション・ルールのサマリー』
このサマリーには、コレクション・ルールの言語エレメントをすべてリストします。

計算パターン

計算ルールは、計算パターンによって定義されます。このルールは、時間間隔内で各イベントを受信するごとに、収集したイベントに対して式を使用して計算を適用します。これはステートフル・ルールです。

概説

計算パターンは、ルール言語の <computeFunction> エレメントで定義された計算関数を、一定の期間にわたって受け入れられた各イベントに対して実行します。期間は、<timeWindow> エレメントによって定義される必須の時間枠によって指定されます。

ルール応答が実行される条件

計算パターンを使用すると、<onTimeWindowComplete> エレメントで定義されているように、時間枠が完了したときにルール応答が実行されます。計算値は、<onTimeWindowComplete> アクションの実行中に使用可能です。

このルール・パターンの使用例

アプリケーションがカスタマー・オーダー・イベントを処理していると想定します。計算パターンの使用例は、以下を行うルールです。

イベントが受信されるたびに、オーダーの合計値が、指定された期間に発生したすべてのオーダーの合計値に追加され、すべてのオーダーの更新された合計値がユーザー・インターフェースを介して公開されます。

関連資料

52 ページの『計算ルールのサマリー』
このサマリーには、計算ルールの言語エレメントをすべてリストします。

重複パターン

重複ルールは、重複パターンによって定義されます。重複ルールでは、指定された時間間隔内に受け入れられた 2 番目以降のイベントをカウントしますが、これらのイベントのルール・セット処理をスキップします。これはステートフル・ルールです。

概説

重複パターンは、通常、一定期間にわたり類似 (重複) するイベントを分離するために使用されます。重複イベントとは、前のイベントと何らかの点で類似しているイベントを指しますが、そのイベントの正確なコピーであるとは限りません。単に、ルールのイベント選択基準を満たしたイベントが、重複していると見なされます。期間は、ルール言語の <timeWindow> エレメントによって定義された、必須の時間枠で指定されます。

ルール応答が実行される条件

重複パターンを使用すると、ルール応答は以下の時点で実行されます。

- 最初のイベントが検出されたとき (<onDetection> エレメントで定義)。
- 各重複イベントが処理されたとき (<onNextEvent> エレメントで定義)。
- 時間枠が完了したとき (<onTimeWindowComplete> エレメントで定義)。

重複イベントを受信していない場合でも、最初のイベントにより <onDetection> アクションがトリガーされます。この振る舞いの理由は、最初のイベントを転送して、重複イベントのルール・セット処理をスキップする必要がある場合があるためです。この場合、ルールに対して、<onDetection> アクションがトリガーされたときに最初のイベントを転送するルール応答アクションを追加できます。

重複イベント (2 番目以降のイベント) に対するデフォルトの処理では、重複イベントはカウントされますが、重複イベントのルール・セット処理はスキップされます。重複イベントに対して追加のアクションを実行する場合は、<onNextEvent> アクションを明示的に定義できます。例えば、重複イベントが、データベースまたはその他のリポジトリーに既に記録されているイベントである場合も考えられます。このため、<onNextEvent> アクションを、このような他のロケーションから重複イベントを除去するようコーディングする必要があります。

<onTimeWindowComplete> アクションは、処理された重複の数を含む、すべての重複イベントのサマリー・レコードを作成するのに使用できます。

このルール・パターンの使用例

「サービス妨害」メッセージが、同じリソース・タイプ (セキュリティー・モニター) から繰り返し発生すると想定します。これは、セキュリティー・ブリーチ (抜け穴) の可能性を示します。重複パターンの使用例は、以下を行うルールです。

> 「サービス妨害」メッセージがセキュリティー・モニターで発生すると、その後 30 秒間に発生するそのイベントの重複はすべてカウントされますが、オペレーター・コンソールには送信されません。また、30 秒間が経過したときに、ルールは、その期間に発生した「サービス妨害」メッセージの数を示すイベントを生成します。

関連資料

54 ページの『重複ルールのサマリー』
このサマリーには、重複ルールの言語エレメントをすべてリストします。

フィルター・パターン

フィルター・ルールは、フィルター・パターンによって定義されます。このルールは、イベントを受け入れるときに特定のアクションを実行します。このルールは単一イベントに対してのみ機能するため、ステートレス・ルールです。

概説

フィルター・パターンは、イベント選択基準と一致する個々のイベントに対して作用します。他のルール・パターンとは異なり、関連する状態情報 (過去のイベントのヒストリーなど) を保持しません。

ルール応答が実行される条件

フィルター・パターンを使用すると、<onDetection> エレメントで定義されているように、イベント選択基準と一致するイベントを受信したときにルール応答が実行されます。

このルール・パターンの使用例

フィルター・パターンの使用例は、以下を行うルールです。

> `ServerStatus` イベントが 95% を超える `serverLoad` を示す場合、ルールは管理者に連絡するアクションを実行します。

関連資料

55 ページの『フィルター・ルールのサマリー』
このサマリーには、フィルター・ルールの言語エレメントをすべてリストします。

シーケンス・パターン

シーケンス・ルールは、シーケンス・パターンによって定義されます。これは、特定の時間間隔内にイベントの特定のシーケンスが到着するかどうかを検出します。

シーケンスは、順序付けられていても、ランダムであってもかまいません。シーケンス・ルールはステートフル・ルールです。

概説

シーケンス・パターンは、一定の期間のイベントのシーケンスをチェックし、シーケンスが完全であるか、または不完全であるかを検出します。不完全なシーケンスには、1 つ以上のイベントが指定されたシーケンスで含まれますが、すべてのイベントは含まれません。

期間は、ルール言語の <timeWindow> エレメントによって定義された、必須の時間枠で指定されます。シーケンス内の各イベントは、ルール内の別個の <eventSelector> エレメントによって定義されます。イベントのシーケンスは、以下のいずれかの順序で検出できます。

- ルールの <eventSelector> エレメントがコーディングされている順序。この場合、ルールが最初の <eventSelector> エレメントによって定義されているイベントを検出すると、シーケンスの検出が開始されます。その後ルールは、2 番目の <eventSelector> エレメントによって定義されるイベントを待機します。
- ランダムな順序。この場合、ルールが <eventSelector> エレメントによって定義されているイベントのうちの任意の 1 つを検出すると、シーケンスの検出が開始されます。続いてルールは、<eventSelector> エレメントによって定義されている別のイベントの 1 つを待機します。

シーケンス・パターンは、主に以下の点でその他のルール・パターンと異なっています。

- ルールによって受け入れられるイベントを定義するための <eventSelector> エレメントが複数個含まれます。少なくとも 2 つの <eventSelector> エレメントが必要です。
- イベントが <eventSelector> エレメントの 1 つによって定義されている基準を満たすと、その <eventSelector> エレメントは、そのルール・インスタンス内の以降のイベント処理から除外されます。
- <eventSelector> エレメントの alias 属性は、シーケンス・ルール内でのみ有効であり、シーケンス・ルール内の特定のイベント・セレクターで選択されたイベントに固有の名前を付けます。フィルター述部またはアクション内の式において、act_eventList 変数を使用して、シーケンス・ルール内のイベントに別名を使用してアクセスできます。

ルール応答が実行される条件

シーケンス・パターンを使用すると、ルール応答は以下の時点で実行されます。

- 時間枠内にイベントの完全なシーケンスが検出されたとき (<onDetection> エレメントで定義)。
- 時間枠内に 1 つ以上のイベントが到着したものの、完全なシーケンスが到着しなかったとき (<onTimeOut> エレメントで定義)。

  シーケンス・パターンは、指定されたシーケンスが不完全であることを検出するのに役立ちます。例えば、「システム停止」イベントが発生し、その後「システ

ム始動」イベントが続かない場合、ルール・ライターは、<onTimeOut> アクションをコーディングして、このようなタイプの欠落したイベントを処理することができます。

このルール・パターンの使用例

完全なシーケンスの検出を説明するシナリオ:
ある IT 環境において、管理者が、DB2® のヒープ・サイズの値が WebSphere® Application Server に影響を与えているかどうかを把握し、影響を与えている場合はこの問題を修正したいと考えているとします。このため、指定された期間内に以下のイベントが以下の順序で発生した場合、管理者は DB2 ヒープ・サイズの値を増やし、データベース・マネージャーを再始動したいと考えます。

1. WebSphere Application Server のリソース割り振り例外。このイベントが `WASResourceAllocationException` タイプであると想定します。
2. DB2 エラー・メッセージ:「Not enough heap to process statement」このイベントが `DB2NotEnoughHeap` タイプであると想定します。

このシナリオの場合、2 つの <eventSelector> エレメントがシーケンス・ルールで定義されており、イベントが (ランダムな順序ではなく) <eventSelector> エレメントがコーディングされている順序で到着する必要があります。最初の <eventSelector> エレメントはイベント `WASResourceAllocationException` をチェックし、 2 番目の <eventSelector> エレメントはイベント `DB2NotEnoughHeap` をチェックしています。指定された時間枠内に、システムに対して以下のイベントが提示されると想定します。

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

ルールの振る舞いは以下のようになります。

1. 最初のイベント `WASResourceAllocationException` は受け入れられます。最初の <eventSelector> エレメントの基準が満たされたため、最初の <eventSelector> エレメントは、このルール内の以降のイベント処理から除外されます。
2. 2 番目のイベント `DB2BackupStarted` は無視されます。
3. 3 番目のイベント `WASResourceAllocationException` は無視されます。
4. 4 番目のイベント `WASResourceAllocationException` は無視されます。
5. 5 番目のイベント `DB2NotEnoughHeap` は受け入れられ、これによりシーケンスが完了します。<onDetection> ルール応答アクションが実行されます。このアクションは DB2 ヒープ・サイズの値を増やし、データベース・マネージャーを再始動するよう定義されています。ルールは初期状態に戻ります。

   これで、最初の <eventSelector> エレメントは、このルールによる以降のイベント処理に組み込まれます。

不完全なシーケンスの検出を説明するシナリオ:
あるビジネス組織が、すべてのカスタマー・オーダーに対して、そのオーダーを受

けてから 1 時間以内に配送の準備を整えたいと考えており、それを実現できない場合にそれを把握したいと考えているとします。

このシナリオの場合、2 つの <eventSelector> エレメントがシーケンス・ルールで定義されており、イベントが (ランダムな順序ではなく) <eventSelector> エレメントがコーディングされている順序で到着する必要があります。最初の <eventSelector> エレメントは `operationType=Order` が指定されたイベント `Netsales` をチェックし、2 番目の <eventSelector> エレメントは `operationType=Delivery` が指定されたイベント `Netsales` をチェックしています。指定された時間枠である 1 時間以内に、システムに対して以下のイベントが発行されると想定します。

1. `operationType=Order` が指定された `Netsales` イベント
2. `operationType=Order` が指定された `Netsales` イベント

ルールの振る舞いは以下のようになります。

1. 最初のイベントは受け入れられます。最初の <eventSelector> エレメントの基準が満たされたため、最初の <eventSelector> エレメントは、このルール内の以降のイベント処理から除外されます。
2. 2 番目のイベントは無視されます。
3. `operationType=Delivery` が指定された `Netsales` イベントが指定された時間枠内に受信されないため、<onTimeOut> ルール応答アクションが実行されます。このアクションは、カスタマー・オーダーの配送準備がオーダーを受けてから 1 時間以内に整っていないことを企業の幹部に通知するよう定義されています。ルールは初期状態に戻ります。

   これで、最初の <eventSelector> エレメントは、このルールによる以降のイベント処理に組み込まれます。

   関連概念

   24 ページの『イベント関連情報へのアクセス』
   以下の例は、Active Correlation Technology によって提供される変数を使用してイベント関連情報にアクセスする方法を示します。

   関連資料

   56 ページの『シーケンス・ルールのサマリー』
   このサマリーには、シーケンス・ルールの言語エレメントをすべてリストします。

しきい値パターン

しきい値ルールは、しきい値パターンによって定義されます。これは、特定の時間間隔内に選択されたイベントのグループを収集し、各イベントが受信された後、しきい値条件が満たされたかどうかを判別します。これはステートフル・ルールです。

概説

しきい値パターンは、特定の期間内に、しきい値に達するまでイベントを収集します。期間は、ルール言語の <timeWindow> エレメントによって定義された、必須の時間枠で指定されます。

しきい値パターンでは、しきい値タイプについて以下の 3 つのオプションが提供されています。

イベント・カウントしきい値

このタイプのしきい値を使用して、特定の期間内でイベント選択基準と一致する必要のあるイベント数を定義できます。指定されたしきい値は、受け入れられたイベントの数と比較されます。イベント・カウントが時間枠内で定義された制限値と等しくなると、しきい値に到達したことになります。

このタイプのしきい値は、非常に単純なイベント・カウント・チェックに便利です。例えば、次のような質問の回答に利用できます。「1 分以内に 5 回のログイン失敗が発生したか。」

このしきい値は、 <eventCountThreshold> エレメントによって定義されます。また <eventCountThreshold> エレメントでは、時間枠に対して指定可能な以下の 2 つの時間間隔モードのうち 1 つを指定します。

固定間隔

固定間隔は、イベント選択基準に一致する最初のイベントを受信したときに始まり、以下のいずれかが発生した時点で終了します。

- 指定された継続時間内にルールがそのしきい値に達した。
- 指定された継続時間が経過した。

スライディング間隔

スライディング間隔は、イベント選択基準と一致する最初のイベントを受信したときに始まります。ただし、ルールがしきい値に達することなく指定された継続時間が経過した場合、時間枠の開始時刻は、新規の「最初の」イベント (通常は次に受け入れられるイベント) の受信時刻に調整 (スライド) されます。このようにして、スライディング間隔は、以下のいずれかが発生する時点まで継続して調整されます。

- 指定された継続時間内にルールがそのしきい値に達した。
- 時間枠を開始するイベントの受信後、指定された継続時間内に後続のイベントを受信しなかった。

時間枠を開始するイベント (新規の「最初の」イベントとなる) は、「受信時刻をルールの時間間隔の継続時間に追加すると、現行時刻よりも後になる」という基準に一致する受信時刻を持つイベントです。以下に、この基準を式の形式で示します。

イベントの受信時刻 + ルールの時間間隔の継続時間 > 現行時刻

このようなイベントが存在しない場合、スライディング間隔はそれ以上の時間調整が不可能になり、間隔は終了します。

計算しきい値

このタイプのしきい値を使用して、受け入れられた各イベントについて計算を実行し、事前に定義された変数に保持される計算されたしきい値を戻すコードを記述 (または他のユーザーによって記述されたコードを使用) することができます。この計算しきい値は、定義済みしきい値と比較され、しきい値に達したかどうかが判別されます。

このため、例えば前のイベントから保管されたデータを使用するなど、複雑な計算を適用して計算されたしきい値を作成 (または更新) することがで

き、ルール・ライターは、計算されたしきい値を計算するロジックとは関係なく定義済みしきい値を設定できます。

このタイプのしきい値は、値の集約および定義済みしきい値との比較に便利です。例えば、特定の期間内の特定のカスタマーに対する売上の合計ドル額を計算し、その合計額を定義済みしきい値と比較するために使用できます。

このしきい値は、<computedThreshold> エレメントによって定義されます。

ブールしきい値

このタイプのしきい値を使用して、受け入れられた各イベントについて `true` または `false` の値を戻すコードを記述 (または他のユーザーによって記述されたコードを使用) することができます。値が `true` の場合、しきい値に達したことを示します。値が `false` の場合、しきい値ルールは、期間が終了するまで、または別のイベントを受け入れるまで処理を継続します。

このタイプのしきい値は、値の範囲のチェックに便利です。例えば、 CPU 使用率を常に 30 % から 80 % の間にする必要がある場合、このしきい値により、使用率がこの範囲内にあることを常に検証できます。

このしきい値は、<booleanThreshold> エレメントによって定義されます。

ルール応答が実行される条件

しきい値パターンを使用すると、ルール応答は以下の時点で実行されます。

- しきい値に達したとき (<onDetection> エレメントで定義)。
- 1 つ以上のイベントが受け入れられたものの、時間枠内でしきい値に達しなかったとき (<onTimeOut> エレメントで定義)。

このルール・パターンの使用例

イベント・カウントしきい値を使用したしきい値パターンの使用例として、以下を実行するルールが考えられます。

30 秒のスライディング時間間隔内に同じサブネットワークから `Server unreachable` イベントが 4 回より多く発行された場合、ルーターの状況をチェックするためのアクションを実行するルール。

関連資料

58 ページの『しきい値ルールのサマリー』
このサマリーには、しきい値ルールの言語エレメントをすべてリストします。

タイマー・パターン

タイマー・ルールは、タイマー・パターンによって定義されます。これにより、一定の間隔でアクションが開始されます。これはステートフル・ルールです。タイマー・ルールはイベントを処理しませんが、イベントによってアクティブ化または非アクティブ化できます。

概説

タイマー・パターンは、期間の最初に開始し、期間の最後に停止するタイマーに類似しています。期間は、ルール言語の <timeWindow> エレメントによって定義された、必須の時間枠で指定されます。

繰り返さないよう指定されていない限り、タイマー・パターンは、タイマー・ルールが非アクティブ化されるまで繰り返されます。このため、タイマー・ルールは、開始されるとアクションを開始する前に指定された期間待機し、ルールが非アクティブ化されるか、Active Correlation Technology エンジンがシャットダウンするまでこの振る舞いを繰り返します。

タイマー・ルールは、イベント選択基準を含まない点で独特です。タイマー・ルールは、<activationInterval> エレメントで定義されるように、ルールのアクティブ化間隔に従って処理を開始します。デフォルトの <activationInterval> エレメントを使用し、タイマー・パターンを repeat に設定した場合、タイマー・ルールは、 Active Correlation Technology エンジンによってロードされたときに開始され、Active Correlation Technology エンジンがシャットダウンしたときに停止します。イベントを使用してタイマー・ルールをアクティブ化するには、ルールの <activationInterval> エレメント内の <activateOnEvent> エレメントにイベントを指定する必要があります。

ルール応答が実行される条件

タイマー・パターンを使用すると、<onTimeWindowComplete> エレメントによって定義されるように、時間枠が完了したときにルール応答が実行されます。

このルール・パターンの使用例

タイマー・パターンは、クリーンアップ・ルールのインプリメントに便利です。タイマー・パターンの使用例として、以下を実行するルールが考えられます。

48 時間より長くオープン状態になっている無害な通知イベントをクリアするアクションを、30 分ごとに実行するルールです。

関連資料

59 ページの『タイマー・ルールのサマリー』
このサマリーには、タイマー・ルールの言語エレメントをすべてリストします。

さまざまなルール・パターンの共通の性質および固有の性質

このマトリックスでは、さまざまなルール・パターンの共通の性質と固有の性質について概要を示します。

表 2 に、ルールの 1 次言語エレメントをリストし、そのエレメントが有効となる各ルール・タイプの列には X を示します。1 次言語エレメントは、さまざまなルール・タイプの直接の子エレメントです。リストには、これらの直接の子エレメントに含まれるエレメントは含まれていません (これらもルール・タイプによって異なることがあります)。また、特定のエレメント属性の妥当性が、ルール・タイプによって異なる場合があります。

表 2. さまざまなルール・パターンの共通の性質および固有の性質を示すマトリックス

エレメント	コレクション	計算	重複	フィルター	シーケンス	しきい値	タイマー
<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	

表 2. さまざまなルール・パターンの共通の性質および固有の性質を示すマトリックス (続き)

エレメント	コレクション	計算	重複	フィルター	シーケンス	しきい値	タイマー
<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

式

式とは、ルールに追加できるカスタム・ロジックを含むコードです。式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

ルール・ライターは、コンテキスト、および取得したい結果に応じて、異なる目的のために式をコーディングできます。式は、変数の初期化、イベント選択基準の定義、およびルール応答アクションおよびライフ・サイクル・アクションの指定に頻繁に使用されます。

式を含む言語エレメント

式を含む各言語エレメントには、式が作成されたプログラミング言語を識別する expressionLanguage 属性が指定されています。サポートされている式言語は、Java プログラミング言語のみです。

式は、以下のルール言語エレメントに含めることができます。

- ルール・セット、ルール・ブロック、またはルール変数の <varInitializer>
- <eventSelector> の <filteringPredicate>
- <groupingKey> の <computedValue>
- 計算ルールの <computeFunction>
- しきい値ルールの <booleanThreshold>
- しきい値ルールの <computedThreshold>
- ルールのルール応答アクション:
  - <onDetection> 内の <action>。このアクションは、重複ルール、フィルター・ルール、シーケンス・ルール、およびしきい値ルールに対してのみ有効です。
  - <onNextEvent> 内の <action>。このアクションは、重複ルールに対してのみ有効です。
  - <onTimeOut> 内の <action>。このアクションは、シーケンス・ルールおよびしきい値ルールに対してのみ有効です。

- <onTimeWindowComplete> 内の <action>。このアクションは、コレクション・ルール、計算ルール、重複ルール、およびタイマー・ルールに対してのみ有効です。

- ルールのライフ・サイクル・アクション:
  - <onLoad> 内の <action>
  - <onActivation> 内の <action>
  - <onDeactivation> 内の <action>
  - <onUnload> 内の <action>

式のコーディングを支援する Active Correlation Technology の機能

ルール・ライターが式をコーディングするのを支援するため、Active Correlation Technology では、以下を行う機能が提供されています。

- 式で使用する外部モジュール (Java クラスなど) およびオブジェクトのインポート。
- ルール・セット、ルール・ブロック、またはルール変数の初期化とアクセス。
- act_event 変数を使用した、ルールが処理中の現行イベントへのアクセス。
- act_eventCount 変数を使用した、ルールで受け入れられたイベント数へのアクセス。
- act_eventList 変数を使用した、ルールで受け入れられたイベント・リストへのアクセス。これには、イベントのさまざまな属性にアクセスする機能や、別名を使用してシーケンス・ルール内の各イベントにアクセスする機能も含まれます。
- act_lib 変数を使用した、メソッド (変数を取得および設定し、ルール・セットにおけるイベント・フローを制御する機能を含む) へのアクセス。
- act_location 変数を使用した、ルール階層における式のロケーションへのアクセス。
- act_nodeName 変数を使用した、ノードの完全修飾名へのアクセス。
- act_threshold 変数を使用した、しきい値ルールの定義されたしきい値へのアクセス。

外部モジュールおよびオブジェクトのインポートとアクセス

この例は、式から外部コード (Java クラスなど) および外部オブジェクトにアクセス可能にする方法を示します。外部オブジェクトとは、アプリケーションが式と通信するために作成するオブジェクトのことです。

式から外部コードにアクセスする前に、その外部コードを式からアクセス可能にしなければなりません。

インポートは、式で外部コードにアクセス可能にするための、プログラム言語固有の方法です。<import> エレメントには、ルール内の他の式で使用できるようインポートする外部モジュール (Java クラスなど) を指定する、特別な種類の式が含まれています。インポートは、ルール・セット・レベルまたはルール・ブロック・レベルで定義できます。

以下の <import> エレメントには、他の式から参照できる StaticHelper クラスおよび Queue クラスをインポートする、Java プログラミング言語で記述された式が含まれています。

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

import ステートメントではクラスの絶対パス名を使用する必要はありませんが、コンパイル時間を短縮するには絶対パス名を指定してください。例えば、 Java クラスを `com.ibm.act.sample.*` や `com.ibm.act.*` ではなく、`com.ibm.act.sample.StaticHelper` と指定します。

静的メソッドへのアクセス

以下の例では、StaticHelper クラスのインポート後に、ルール応答アクション内の式でこのクラスがどのように参照されるかを示します。

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

オブジェクトのインスタンス・メソッドへのアクセス

以下の例では、 Queue クラスのインポート後に、ルール応答アクション内の式でこのクラスがどのように参照されるかを示します。 この例では、`OutputQueueOne` という名前を持つ `Queue` タイプの外部オブジェクトが取得され、特定のキューにイベントを書き込むために使用されます。

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

変数の初期化とアクセス

この例は、ルール・セット、ルール・ブロック、またはルール変数を初期化し、アクセスする方法を示します。

変数は、ルール・セット、ルール・ブロック、またはルールの各レベルで定義できます。変数にアクセスするには、初期化式を使用して変数を初期化する必要があります。以下の式は、hostsList および hostsString という 2 つの変数を初期化します。

```
<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>
```

すべての変数には式を使用してアクセスします。以下の例は、上記の例で初期化された hostsList 変数および hostsString 変数に、ルール応答アクション内の式を使用してアクセスする方法を示します。この例では、hostsList を変更し、hostsString に新しい値を指定しています。

```
<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>
```

イベント関連情報へのアクセス

以下の例は、Active Correlation Technology によって提供される変数を使用してイベント関連情報にアクセスする方法を示します。

現行イベントにアクセスする例:
以下のコードは、 act_event 変数を使用してイベントの hostname 属性を取得する方法を示します。

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

索引を使用してイベント・リストからイベントにアクセスする例:
以下のコードは、act_eventList 変数を使用して、イベント・リストの 1 番目のイベントを取得する方法を示します。

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

別名を使用してイベント・リストからイベントにアクセスする例:
他のルール・タイプとは異なり、シーケンス・ルールでは複数のイベント・セレクターを使用でき、実際に最低 2 つのイベント・セレクターを使用する必要があります。<eventSelector> エレメントの alias 属性は、シーケンス・ルール内でのみ有効であり、シーケンス・ルール内の特定のイベント・セレクターで選択されたイベントに固有の名前を付けます。フィルター述部またはアクション内の式において、act_eventList 変数を使用して、シーケンス・ルール内のイベントに別名を使用してアクセスできます。

以下のコードは、シーケンス・ルールの 2 つのイベント・セレクターを示します。別名は、TECevent および WASevent です。

```
<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>
```

以下のコードは、act_eventList 変数を使用して TECevent という名前の最初のイベント・セレクターで受け入れられたイベントを取得する方法を示します。

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

式のコーディングのベスト・プラクティス

ここでは、式を効率的かつ効果的にコーディングするためのいくつかのベスト・プラクティスとヒントを記載します。

- 理解しやすくするため、ここで示す式の例の大半には、Java コードが XML 構成内に直接含まれています。しかし、ルールを作成する際は、 Java コードを含む外部モジュールを使用し、これらの外部モジュールを式の一部として呼び出すのがベスト・プラクティスと言えます。

  ルール・ビルダーで既存のスニペットを使用または編集するか、新規スニペットを作成して、外部モジュールを呼び出すコードとして使用することもできます。スニペットは、式の中で使用できるソース・コードの抜粋です。ルール・ビルダーにおいて、スニペットは「断片」ビューを介して使用できます。

  この方法により、Java コードの設計、開発、編集、テスト、およびデバッグの作業を任意の統合開発環境 (IDE) で実行でき、開発プロセスの標準工程として制御できます。

- 式のコードが XML として構文解析されないようにするために、コードを CDATA セクション内に含めます。このセクションでは、以下の例に示すように、 `<![CDATA[` をコードの直前に配置し、`]]>` をコードの直後に配置します。

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

  XML パーサーは、CDATA セクション内にあるものはすべて無視します。

- 式内で act_lib.getExternalContext() メソッドを使用する場合は、このメソッドから戻されるオブジェクトをルール・セットまたはルール・ブロック変数内に保管しないでください。理由としては、アプリケーションがオブジェクト・インスタンスへの参照を変更することがあり、関連するルール・セットまたはルール・ブロック変数が更新されないためです。

- <action> エレメント内の式で return ステートメント (`return;`) が使用され、それぞれのルール応答アクションまたはライフ・サイクル・アクションについて追加の <action> エレメントがコーディングされている場合、コードの実行は、 return ステートメントがコーディングされている箇所で終了し、次の <action> エレメント内の式で再度開始されます。

- ルール管理およびその他の Active Correlation Technology エンジン・メソッドは、ルール応答アクションまたはライフ・サイクル・アクション内から呼び出せません。

関連情報

22 ページの『外部モジュールおよびオブジェクトのインポートとアクセス』
この例は、式から外部コード (Java クラスなど) および外部オブジェクトにアク

セス可能にする方法を示します。外部オブジェクトとは、アプリケーションが式と通信するために作成するオブジェクトのことです。

45 ページの『ルール内の式にスニペットを組み込む』
ルール内の式に、Eclipse ワークベンチ内の「断片」ビューからスニペットを組み込むことができます。

変数

ルール言語では、特定の変数を使用して、さまざまなイベントの発生やルール間でイベント関連の情報を保管します。そして、このイベント関連の情報に、ルール内の式からアクセスできます。ルール・ライターが定義するタイプの変数と、Active Correlation Technology によって提供されるタイプの変数があります。式の中で直接アクセスできるタイプと、Active Correlation Technology によって提供されるメソッドを介してのみアクセスできるタイプがあります。

<variable> エレメント内で定義され、メソッドを介してアクセスされる変数

ルール、ルール・ブロック、またはルール・セットの <variable> エレメントで変数を定義できます。これにより、以下のいずれかのメソッドを使用して、式の中でこの変数にアクセスできます。

- getVariable() メソッドまたは get*javatype*Variable() メソッドの 1 つ
- setVariable() メソッドまたは set*javatype*Variable() メソッドの 1 つ

例えば、ルールの <variable> エレメントで *rule_writer_variable* 変数を定義した場合、以下のコードを使用してこの変数にアクセスできます。

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

Active Correlation Technology によって提供され、式の中で直接アクセスされる変数

以下の変数は、Active Correlation Technology によって提供されています。これらの変数は式の中でインラインで使用できます。

- act_event
- act_eventList
- act_lib

例えば、以下のコードを使用して act_event 変数にアクセスし、イベントの hostname 属性を取得できます。

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

Active Correlation Technology によって提供され、メソッドを介してアクセスされる変数

以下の変数は、Active Correlation Technology によって提供されています。getVariable() メソッドまたは get*javatype*Variable() メソッドの 1 つを使用して、式の中でこれらの変数にアクセスできます。

- act_eventCount

- act_location
- act_nodeName
- act_threshold

例えば、以下のコードを使用して、act_eventCount 変数にアクセスできます。

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

表 3 に、IACTLibrary インターフェースがこれらの変数に対して提供する定数を示します。コードにミススペルやタイプミスがある場合、実行時ではなくコンパイル時に検出されるようにするには、常に変数自体ではなく、それらの変数を表す定数を使用するようにしてください。例えば、
`act_lib.getIntVariable("act_eventCount");` ではなく
`act_lib.getIntVariable(IACTLibrary.EVENTCOUNT);` を使用してください。

表 3. 変数および関連する定数

変数	関連する定数
act_eventCount	EVENTCOUNT
act_location	LOCATION
act_nodeName	NODENAME
act_threshold	THRESHOLD

関連資料

121 ページの『variable エレメント』
<variable> エレメントは、変数を定義し、式によって参照可能な 形式で情報を保持します。変数は、ルール・セット、ルール・ブロック、または ルールの各レベルで定義できます。

Active Correlation Technology 変数のデータ・タイプ

Active Correlation Technology によって提供される変数のデータ・タイプはさまざまです。

表 4 では、これらの変数のデータ・タイプを示します。

表 4. Active Correlation Technology 変数のデータ・タイプ

変数	データ・タイプ
act_ event	com.ibm.correlation.IEvent インターフェースで定義されるタイプ
act_ eventCount	int
act_ eventList	com.ibm.correlation.IEventList インターフェースで定義されるタイプ
act_ lib	com.ibm.correlation.IACTLibrary インターフェースで定義されるタイプ
act_ location	java.lang.String
act_ nodeName	java.lang.String
act_ threshold	この変数は、しきい値ルールの <computedThreshold> または <eventCountThreshold> エレメントの threshold 属性値であるため、データ・タイプは threshold 属性値のデータ・タイプと同じでなくてはなりません。

変数が有効となる式のコンテキスト

Active Correlation Technology によって提供される変数は、特定の式コンテキスト内でのみ有効です。

表 5 に、これらの変数が有効となる式コンテキストをリストします。この表では、それぞれの式のコンテキストにおいて有効な各変数の列に X が示されています。これらの変数に適用されるその他の使用制限事項については、それぞれの変数についてのトピックで説明しています。

表 5. 変数が有効となる式のコンテキスト

式のコンテキスト	act_ event	act_ eventCount	act_ eventList	act_ lib	act_ location	act_ nodeName	act_ threshold
<onActivation> 内の <action>	X			X	X	X	X
<onDeactivation> 内の <action>	X	X	X	X	X	X	X
<onDetection> 内の <action>	X	X	X	X	X	X	X
<onLoad> 内の <action>				X	X	X	X
<onNextEvent> 内の <action>	X	X	X	X	X	X	
<onTimeOut> 内の <action>		X	X	X	X	X	X
<onTimeWindowComplete> 内の <action>		X	X	X	X	X	
<onUnload> 内の <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
ルールの <varInitializer>				X	X	X	X

act_event 変数

act_event 変数は、現行イベントに適用されるメソッドへのアクセスを提供します。

詳細

タイマー・ルールはイベントを処理しないため、タイマー・ルール内の act_event 変数は、ルールをアクティブ化または非アクティブ化するイベントに対してのみ適用されます。

act_event 変数は、イベントがルールをアクティブ化または非アクティブ化した場合のみ <onActivation> および <onDeactivation> アクション内で適用されます。それ以外の場合、この変数は NULL です。

コーディングの例

以下のコードは、act_event 変数にアクセスして、イベントの hostname 属性を取得します。

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

アクセスできるメソッド

act_event 変数がアクセスを提供するメソッドは、 29 ページの表 6 で示すように IEvent インターフェースで定義されます。

表 6. IEvent インターフェースと対応するメソッド、および Javadoc メソッド説明のロケーション

インターフェース	メソッド	Javadoc メソッド説明のロケーション
IEvent	• get • getAttribute • getBooleanAttribute • getByteAttribute • getShortAttribute • getIntAttribute • getLongAttribute • getFloatAttribute • getDoubleAttribute • getStringAttribute • set • getTimeStamp • setTimeStamp • getType • getOriginal	com.ibm.correlation.IEvent

act_eventCount 変数

act_eventCount 変数は、ルールで受け入れられたイベント数と同じ整数です。

詳細

重複ルールでは、act_eventCount 変数の値は受け入れられたイベントの総数です。これには、元のイベントと重複したすべてのイベントの両方が含まれます。それ以外のすべてのルール・タイプの場合、値はイベント・リストのサイズと同じです。これは、act_eventList.size() メソッドを使用して act_eventList 変数を介して取得できます。

タイマー・ルールではイベントを処理しないため、act_eventCount 変数と act_eventList 変数はタイマー・ルール内では無効です。

ルールがグループ化キーを使用して定義されている場合、act_eventCount 変数、act_eventList 変数、および act_threshold 変数は以下の式コンテキストでは無効です。

- ライフ・サイクル・アクション
- <activateOnEvent> に含まれる <filteringPredicate> または <activationInterval> に含まれる <deactivateOnEvent>
- <computedValue>

これは、この場合ルール変数がルール・インスタンスにのみ適用され、これらの式の実行時にルール・インスタンスが存在しないためです。

コーディングの例

以下のコードは act_lib 変数にアクセスして、ルールで受け入れられたイベント数を取得します。

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

act_eventList 変数

act_eventList 変数は、ルールで受け入れられたイベントのリストに適用されるメソッドへのアクセスを提供します。

詳細

フィルター・ルールと重複ルールは常に、1 つ以下のイベントのリストを持ちます。フィルター・ルールはステートレス・ルールであり、重複ルールは最初に分析されたイベントのみを保持するためです。

タイマー・ルールではイベントを処理しないため、act_eventCount 変数と act_eventList 変数はタイマー・ルール内では無効です。

ルールがグループ化キーを使用して定義されている場合、act_eventCount 変数、act_eventList 変数、および act_threshold 変数は以下の式コンテキストでは無効です。

- ライフ・サイクル・アクション
- <activateOnEvent> に含まれる <filteringPredicate> または <activationInterval> に含まれる <deactivateOnEvent>
- <computedValue>

これは、この場合ルール変数がルール・インスタンスにのみ適用され、これらの式の実行時にルール・インスタンスが存在しないためです。

コーディングの例

以下のコードは、act_eventList 変数にアクセスして、イベント・リストの 2 番目のイベントを取得します。

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

アクセスできるメソッド

act_eventList 変数がアクセスを提供するメソッドは、表 7 で示すように IEventList インターフェースで定義されます。

表 7. IEventList インターフェースと対応するメソッド、および Javadoc メソッド説明のロケーション

インターフェース	メソッド	Javadoc メソッド説明のロケーション
IEventList	• get • size • isEmpty • listIterator	com.ibm.correlation.IEventList

act_lib 変数

act_lib 変数は、Active Correlation Technology のライブラリー・メソッドへのアクセスを提供します。

詳細

act_lib 変数がアクセスできるメソッドは、この変数が使用される式を含むルール言語エレメントによって異なります。表 8 を参照してください。

表 8. act_lib 変数がアクセスできるメソッド (act_lib 変数を含む式のコンテキストに基づく)

式のコンテキスト	IACTLibrary メソッド	IExitableActionLibrary メソッド	IActionLibrary メソッド
<onActivation> 内の <action>	X		
<onDeactivation> 内の <action>	X		
<onDetection> 内の <action>	X	X	X
<onLoad> 内の <action>	X		
<onNextEvent> 内の <action>	X	X	X
<onTimeOut> 内の <action>	X		X
<onTimeWindowComplete> 内の <action>	X		X
<onUnload> 内の <action>	X		
<booleanThreshold>	X		
<computedThreshold>	X		
<computedValue>	X		
<computeFunction>	X		
<filteringPredicate>	X		
<varInitializer>	X		

コーディングの例

以下のコードは act_lib 変数にアクセスして、ルール・セットを終了させるメソッドを呼び出します。

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

アクセスできるメソッド

act_lib 変数がアクセスを提供できるメソッドは、 32 ページの表 9 に示すように、IACTLibrary、IExitableActionLibrary、および IActionLibrary インターフェースで定義されます。

表 9. インターフェースと対応するメソッド、および *Javadoc* メソッド説明のロケーション

インターフェース	メソッド	Javadoc メソッド説明のロケーション
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 インターフェースで定義されるメソッドは、 IActionLibrary インターフェースでも使用可能です。	com.ibm.correlation.IActionLibrary
IExitableActionLibrary	• exitRuleSet • exitRuleBlock IACTLibrary インターフェースおよび IActionLibrary インターフェースで定義されるメソッドは、 IExitableActionLibrary インターフェースでも使用可能です。	com.ibm.correlation.IExitableActionLibrary

act_location 変数

act_location 変数は、ルール階層における式のロケーションを示すストリングです。

詳細

ロケーションは、ルール階層における式の位置を示す完全修飾名です。これは、*identifier.identifier....* という形式で、各 *identifier* には、以下のうち 1 つが指定されます。

- それぞれの階層にある XML エレメントの name 属性の値。
- ルール・ブロックまたはルール内で複数回発生し、name 属性を持たないエレメントの場合は、式を含む XML エレメントとその後に続く大括弧で囲まれた索引番号。この索引番号は、式を含むエレメント内におけるその式の位置を示します。索引番号を割り当てるためのカウンターは、 1 ではなく 0 から開始されます。このため、例えばエレメントが 3 番目の <action> エレメントに含まれる場合、索引番号は `action[2]` のように示されます。

これらの ID は、最も高いレベルのルール・ブロックから、式を含む最も低いレベルのエレメントへの降順になります。

コーディングの例

以下のコードは act_lib 変数にアクセスして、式のロケーションを取得します。

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

変数から戻されるロケーションの例

以下の値は、 act_location 変数から戻されるロケーションの例です。

`ruleBlockA.ruleA.eventSelector[3].filteringPredicate`
この式は、以下の中に含まれます。
- name 属性値 `ruleBlockA` を持つルール・ブロック
- name 属性値 `ruleA` を持つルール
- 4 番目の <eventSelector> エレメント
- <filteringPredicate> エレメント

`ruleBlockA.ruleA.onDetection.action[5]`
この式は、以下の中に含まれます。
- name 属性値 `ruleBlockA` を持つルール・ブロック
- name 属性値 `ruleA` を持つルール
- <onDetection> エレメント
- 6 番目の <action> エレメント

`ruleBlockA.ruleA.variableA.varInitializer`
この式は、以下の中に含まれます。
- name 属性値 `ruleBlockA` を持つルール・ブロック
- name 属性値 `ruleA` を持つルール
- name 属性値 `variableA` を持つ属性
- <varInitializer> エレメント

act_nodeName 変数

act_nodeName 変数は、ノードの完全修飾名を示すストリングです。

詳細

Active Correlation Technology において、ノードは、ルール・セットに対して、独立して個別に追加、除去、または置き換えることができるルール階層内のオブジェクトです。具体的には、ルール、ルール・ブロック、ルール・ブロック変数、およびルール・セット変数といったオブジェクトがノードです。ルール・レベルより下のレベルでは、オブジェクトを独立して個別に操作することはできないため、ルール変数はノードではありません。

name 属性値 `ruleBlockA` を持つルール・ブロック内にある、name 属性値 `rule1` を持つルールの完全修飾ノード名は、`ruleBlockA.rule1` です。ルールはルール・セット内で階層的に編成されているため、完全修飾ノード名では、より低いレベルのノードへの降下を示すためにピリオド (.) が使用されます。

コーディングの例

以下のコードは、act_nodeName 変数にアクセスして、ノードの完全修飾名を取得します。

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

act_threshold 変数

act_threshold 変数は、しきい値ルールの <computedThreshold> または <eventCountThreshold> エレメントの threshold 属性の値 (定義されたしきい値) です。

詳細

act_threshold 変数は、しきい値ルール内でのみ有効です。

ルールがグループ化キーを使用して定義されている場合、act_eventCount 変数、act_eventList 変数、および act_threshold 変数は以下の式コンテキストでは無効です。

- ライフ・サイクル・アクション
- <activateOnEvent> に含まれる <filteringPredicate> または <activationInterval> に含まれる <deactivateOnEvent>
- <computedValue>

これは、この場合ルール変数がルール・インスタンスにのみ適用され、これらの式の実行時にルール・インスタンスが存在しないためです。

コーディングの例

以下のコードは act_lib 変数にアクセスして、定義されたしきい値を取得します。

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

ルール・セットのイベント・フロー

イベントは、ルール・ブロックおよびルールがコーディングされた順序でルール・セット内をフローします。Active Correlation Technology エンジンがイベントを受信すると、エンジンはイベント・タイプを判別し、ルールのアクティブ化、イベントの処理、またはルールの非アクティブ化にこのイベント・タイプを使用するルールを識別します。

ルールによるイベントの使用方法

イベントを使用する各ルールでは、ルールのアクティブ化、イベントの処理、またはルールの非アクティブ化に対して指定されたすべての基準にイベントが一致するかどうか、最初に判別されます。一致する場合、ルールは以下のアクションを実行します。

ルールのアクティブ化の場合
ルールの <onActivation> エレメントに含まれるアクションが実行されます (コーディングされている場合)。

イベント処理の場合
ルールはイベントを処理します。ルール・パターンが一致する場合、ルール応答アクションが実行されます (コーディングされている場合)。状態によっては、ルール応答アクションでは以下を行うことができます。

- アクションにより、イベントにルール・ブロックまたはルール・セットの残りの部分での処理をスキップさせることができます。
- アクションにより、処理のために、別のルールまたはルール・ブロックに新規または既存のイベントを送信できます。

ルールの非アクティブ化の場合
ルールの <onDeactivation> エレメントに含まれるアクションが実行されます (コーディングされている場合)。

イベント・フローに影響するメソッド

Active Correlation Technology では、ルール・セットにおけるイベントのフローに影響を与えるために呼び出すことができる以下のメソッドを提供しています。これらのメソッドは、act_lib 変数を介して使用できます。

exitRuleSet
このメソッドは、現行イベントがルール・セットの以降のルールで処理されないことを指定します。

exitRuleBlock
このメソッドは、現行イベントが現行ルール・ブロック、またはこのルール・ブロックが含むすべてのルール・ブロックの以降のルールで処理されないことを指定します。ただし、現行ルール・ブロックの有効範囲外にある以降のルールでは処理されます。

forward
このメソッドは、現行ルールが処理を完了していない場合でも、イベントが他のルールおよびルール・ブロックに送信されることを指定します。その後、他の各ルールおよびルール・ブロックは、完全にイベントを処理してから、forward メソッドを呼び出したルールにイベントを戻します。

forwardOnCompletion

このメソッドは、現行ルールが処理を完了した後に、イベントが他のルールおよびルール・ブロックに送信されることを指定します。

第 3 章 ルール記述の概要

イベントを相関させるルールを記述する前に、イベント相関について理解および計画し、ルールを設計する必要があります。Active Correlation Technology ルール・ビルダーを使用して、ルールを記述し、ルール・セットをコンパイルすることができます。

Active Correlation Technology ルール・ビルダーは、ルールを記述するための GUI です。これにはオンライン・ヘルプが含まれています。ルール・ビルダーにおいて、結果のルール・セット・ファイルは、ファイル・タイプが actl の XML ファイルです。

Active Correlation Technology では、変更管理またはバージョン管理システムは提供されていません。

イベント相関の計画

イベント相関の計画には、イベント相関とは何であるか、およびご使用のアプリケーションにどのように適用できるかについての理解または学習が含まれます。

以下の概念について理解できていることを確認してください。

- 3 ページの『第 1 章 概要』 および 5 ページの『第 2 章 ルール言語の概要』 内の情報
- ご使用のアプリケーションが処理するイベント

各アプリケーションは、以下の例で説明しているように、さまざまなイベントのセットを処理することができます。

保険業の例

保険業においては、請求プロセス全体の作業フローを追跡するイベントを生成して相関させ、ビジネス・プロセスが適時に完了しているかどうかを判別することができます。

販売業の例

異なるタイプのビジネスにおいて、販売結果を定期的に要約、報告、および目標と比較して、特定の期間における販売目標の達成状況を示すことができます。

IT 環境の例

IT 環境においては、主幹業務のシステムによって 1 分ごとにイベントを生成し、データベース・サーバーが正常に稼働していることを示すことができます。相関ルールを記述して、これらのハートビート・イベントの受信をモニターし、予期されるハートビート・イベントが受信されない場合に特定のルール応答アクションを実行することができます。

また、ご使用のアプリケーションが処理するイベントのフォーマットも理解する必要があります。Active Correlation Technology では、Active Correlation Technology エンジンによって処理されているイベント内のデータにアクセスするために、Java

クラスおよびメソッドが提供されています。ただし、これらのクラスおよびメソッドを使用して処理中のイベントに対してアクセスまたは変更を行う場合は、基礎となるイベント・オブジェクトについての基本的な理解が重要です。

イベント相関について計画するには、以下のステップを実行します。

1. 相関させるご使用のアプリケーションからのイベントを決定する。
2. イベントを相関させるためのルール・パターンを決定する。

   ルール・パターンは特定のイベント相関状態を表し、これを使用して、何らかの形でその状態の原因となっているイベントを相関させることができます。アプリケーションが処理するイベントが、Active Correlation Technology ルール言語によって定義されるルール・パターンにどのように関連しているかを考慮してください。これにより、使用する必要があるルール・パターンを決定できます。

   常に、イベント相関状況に最も適したパターンを使用してください。例えば、イベントの特定のシーケンスを検出するルールが必要な場合は、フィルター・ルールに対して、ルール応答アクション内にシーケンス・パターンの振る舞いを含むコードを記述しないでください。シーケンス・パターンはむしろ、シーケンス・ルールの作成に使用してください。
3. 使用する各ルール・パターンの構成を確認する。

   以下の情報は、ルール言語の基本構成を要約したものですが、それぞれの詳細は、ルール・パターンごとに一意です。この情報は、ルール・ビルダー GUI での表示とほぼ同じになるように構成されています。

   特性　ルールの特性の定義。ルール名、説明、およびパターンが含まれます。詳しくは、以下のトピックを参照してください。
   - 78 ページの『collectionRule エレメント』
   - 80 ページの『computationRule エレメント』
   - 87 ページの『duplicateRule エレメント』
   - 95 ページの『filterRule エレメント』
   - 111 ページの『sequenceRule エレメント』
   - 116 ページの『thresholdRule エレメント』
   - 119 ページの『timerRule エレメント』

   変数　ルールの変数の定義。各変数の名前、タイプ、説明、および初期化式が含まれます。詳しくは、121 ページの『variable エレメント』 を参照してください。

   イベント選択
   ルールで処理するために受け入れるイベントを決定する基準の定義。詳しくは、92 ページの『eventSelector エレメント』 を参照してください。

   グループ化キー
   グループ化キーの定義。これは、共通の特性を共有するイベントの各グループについて、別個のルール・インスタンス (または自身のコピー) を作成するようルールに指示するための方法です。詳しくは、96 ページの『groupingKey エレメント』 を参照してください。

パターン特性

ステートフル・ルールがそのパターンの突き合わせ処理を行う期間の指定、および特定のステートフル・ルール・パターンの固有の特徴の定義。詳しくは、 120 ページの『timeWindow エレメント』 を参照してください。

計算ルールの場合、これには収集されたイベントに適用する計算の定義が含まれます。詳しくは、 84 ページの『computeFunction エレメント』を参照してください。

しきい値ルールの場合、これには、しきい値タイプの定義およびしきい値タイプに特有のその他の情報が含まれます。詳しくは、以下のトピックを参照してください。

- 77 ページの『booleanThreshold エレメント』
- 81 ページの『computedThreshold エレメント』
- 89 ページの『eventCountThreshold エレメント』

ルール応答

ルールがその処理を完了したときに実行するアクションの定義。

詳しくは、以下のトピックを参照してください。

- 重複、フィルター、シーケンス、またはしきい値ルールの場合: 103 ページの『onDetection エレメント』
- 重複ルールの場合: 104 ページの『onNextEvent エレメント』
- シーケンスまたはしきい値ルールの場合: 105 ページの『onTimeOut エレメント』
- コレクション、計算、重複、またはタイマー・ルールの場合: 106 ページの『onTimeWindowComplete エレメント』

アクティブ化間隔

ルールがいつアクティブおよび非アクティブになるかの定義。詳しくは、 71 ページの『activationInterval エレメント』 を参照してください。

ライフ・サイクル

ルールのライフ・サイクルにおける、ロード、アクティブ化、非アクティブ化、およびアンロードの 4 つの基本ステージで実行するアクションの定義 (必要な場合)。通常、これらのアクションを定義する必要はありません。詳しくは、 100 ページの『lifeCycleActions エレメント』 を参照してください。

4. ルール式内で呼び出される Java メソッドおよび関連するスニペットを確認する。ルール・ライターは、多くの Java コードをルール式内に記述するのではなく、Java メソッドを使用して外部モジュールを呼び出すようにしてください。これらの外部モジュールは、Active Correlation Technology を組み込むアプリケーションによって提供されるか、必要に応じてルール・ライターが作成できます。各 Java メソッドと関連するスニペットも確認してください。詳しくは、 25 ページの『式のコーディングのベスト・プラクティス』 を参照してください。

40 ページの『イベントを相関させるルールの設計』 に進みます。

イベントを相関させるルールの設計

イベント相関を計画したら、イベントを相関させるルールを設計する必要があります。

最初に、 37 ページの『イベント相関の計画』 を完了します。

ルールを設計するには、以下のステップを実行します。

1. ルール・セット内のルールとルール・ブロックの編成を設計します。
2. さまざまな変数を定義するレベル (ルール・セット・レベル、ルール・ブロック・レベル、またはルール・レベルなど) を設計します。
3. ルール・パターンに基づいて各ルールのエレメントを設計します。

   このステップでは、計画段階で指定した各ルール・パターンの構成を利用します。
4. ルール間の相互作用、特にイベントの転送と、重複イベントのルール・セット処理のスキップに関して設計します。

   詳しくは、 35 ページの『ルール・セットのイベント・フロー』 を参照してください。
5. 式内で呼び出される、作成したすべての Java メソッドおよび関連するスニペットを設計、作成、およびテストします。

『ルール・ビルダー入門』 に進みます。

ルール・ビルダー入門

イベントを相関させるためのルールの設計後、Active Correlation Technology のルール・ビルダーを使用してルールを記述することができます。

以下のステップは、ルール・ビルダーを使用してルールを記述する際の基本ステップです。

1. Eclipse ワークベンチを開きます。
2. Eclipse ワークベンチでパースペクティブを設定します。
3. Active Correlation Technology に対するプリファレンスを任意に設定するか、デフォルトの設定を受け入れます。
4. ルール・セット・ファイルを保管するプロジェクトを作成するか、既存のプロジェクトを使用します。
5. ルール・セット・ファイルを作成し、選択したプロジェクトに保管します。
6. ルール・セット内には少なくとも 1 つのルール・ブロックを作成します。さらに、ルール・セット内に別のルール・ブロックを作成したり、ルール・ブロック内にルール・ブロックを作成したりすることができます。
7. ルール・ブロック内にルールを作成します。
8. ルール・セットを検証します。
9. ルール・セットをコンパイルします。
10. ルール・セットを必要に応じて更新します。

Eclipse ワークベンチ内の「断片」ビューから、ルール内の式にスニペットを組み込むこともできます。

Eclipse ワークベンチでのパースペクティブの設定

いずれの作業を実行するよりも前に、Eclipse ワークベンチでパースペクティブを設定する必要があります。ここでは、ルール・ビルダー・パースペクティブを開く方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。

ルール・ビルダー・パースペクティブ以外のパースペクティブも使用できますが、選択するパースペクティブによって、さまざまな作業を実行する手順が若干異なります。

ルール・ビルダー・パースペクティブを開くには、以下のステップを実行します。

1. ワークベンチで、「**ウィンドウ**」 → 「**パースペクティブを開く**」 → 「**その他...**」をクリックします。
2. 「**ルール・ビルダー**」をクリックし、「**OK**」をクリックします。 これにより、ルール・ビルダー・パースペクティブが表示されます。

『プリファレンスの設定』 に進みます。

プリファレンスの設定

ルール・セット・ファイルを作成する前に、Active Correlation Technology のプリファレンスが Eclipse ワークベンチで正しく設定されていることを確認する必要があります。ここでは、これらのプリファレンスの設定方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。

Active Correlation Technology に対するプリファレンスを任意に設定する場合は、以下の手順を実行します。

1. ワークベンチで「**ウィンドウ**」 → 「**設定...**」をクリックします。
2. 「**Active Correlation Technology**」をクリックし、「Active Correlation Technology」ページで、ルール・ビルダーにおいて新規に作成されたルールおよびルール・ブロックの接頭部として 『act』 を付加するかどうかを指定します。デフォルトでは、『act』 は接頭部として付加されません。
3. 「**Active Correlation Technology**」を展開します。 ご使用のアプリケーションによっては、以下の追加項目が表示される可能性があります。これらの項目をクリックして、関連するプリファレンスを設定してください。

項目	関連するプリファレンス
コモン・ベース・イベント定義プロバイダー	コモン・ベース・イベント仕様に準拠する (所定のイベント・タイプに含めることができる属性の名前とタイプを含む) 1 つ以上のイベント・タイプの構造を規定する XML ファイルを指定できます。これにより、これらのイベント・タイプと属性をルール作成時に使用可能になります。

項目	関連するプリファレンス
コンパイラー	以下の基本項目を指定できます。 • コンパイルされたルール・セットを保管するかどうか • コンパイルされたルール・セットのファイル・タイプ • コンパイル時に使用するコードのクラス・パス デフォルトでは、コンパイルされたルール・セットが保管され、ファイル・タイプ `acts` として「ナビゲーター」ビューに表示されます。これにより、コンパイラー出力がシリアライズ化されたルール・セットであることがわかります。

『ルール・セット・ファイルを保管するためのプロジェクトの作成』 に進みます。

ルール・セット・ファイルを保管するためのプロジェクトの作成

Eclipse ワークベンチでルール・セット・ファイルを作成するときは、ファイルを保管するプロジェクトを指定する必要があります。このため、ルール・セット・ファイルを作成する前にプロジェクトを作成するか、既存のプロジェクトを使用する必要があります。ここでは、Eclipse ワークベンチでプロジェクトを作成する方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

ワークベンチで単純なプロジェクトを作成するには、以下のステップを実行します。

1. 「**ファイル**」 → 「**新規**」 → 「**プロジェクト...**」をクリックします。
2. 「新規プロジェクト」ウィザードで、「**シンプル**」 → 「**プロジェクト**」をクリックしてから、「**次へ**」をクリックします。
3. 「**プロジェクト名**」フィールドにプロジェクトの固有名を入力します。また、プロジェクトのデフォルトのロケーションを使用するか、別のロケーションを選択します。
4. 「**終了**」をクリックします。 これにより、新規プロジェクトがルール・ビルダー・パースペクティブの「ナビゲーター」ビューに表示されます。

『ルール・セットの作成』 に進みます。

ルール・セットの作成

ここでは、ルール・セットの作成方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

ルール・セット・ファイルを作成するには、以下のステップを実行します。

1. 「**ファイル**」 → 「**新規**」 → 「**ルール・セット・ファイル**」をクリックします。
2. 「新規ルール・セット・ファイル」ウィザードで、ルール・セット・ファイルを保管するプロジェクトと関連付けられているフォルダーの名前をクリックします。これは 42 ページの『ルール・セット・ファイルを保管するためのプロジェクトの作成』 で作成したプロジェクトである可能性があります。 フォルダー名が最初のフィールドに表示されます。
3. 「**ファイル名**」フィールドに、ルール・セット・ファイルの名前を入力します。ファイル・タイプは actl である必要があります。
4. 「**次へ**」をクリックします。
5. ルール・セットの固有名と、任意で説明を入力します。「**XML エンコード**」フィールドでデフォルト値を使用しない場合は、XML ファイルであるルール・セット・ファイルのエンコード・スタイルも指定します。
6. 「**終了**」をクリックします。 これにより、「ナビゲーター」ビューの該当するプロジェクト・フォルダー内にルール・セット・ファイルが表示されます。ファイルは保管されるときに検証されるため、「検証済み」という語がファイルの横に表示されます。ルール・セット・ファイルは「アウトライン」ビューにも表示されます。

『ルール・ブロックの作成』 に進みます。

ルール・ブロックの作成

ここでは、ルール・セットまたは別のルール・ブロック内にルール・ブロックを作成する方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

初めてルール・セットを作成する場合、ルールを作成するには、ルール・セット内に少なくとも 1 つのルール・ブロックを作成する必要があります。最初のルール・ブロックを作成した後は、ルール・ブロック内のルール・ブロックを含め、追加のルール・ブロックを作成できます。

ルール・ブロックを作成するには、以下のステップを実行します。

1. 「ナビゲーター」ビューで、更新するルール・セット・ファイルの名前をダブルクリックします。 これにより、「アウトライン」ビューでファイルが開きます。「アウトライン」ビューでルール・セットをクリックすると、そのルール・セットの現行情報がエディター領域に表示されます。
2. 「アウトライン」ビューで、ルール・セットを右クリックします。
3. 「**子の新規作成**」 → 「**ルール・ブロック**」をクリックします。 これにより、「アウトライン」ビューのルール・セット内にルール・ブロックが表示され、ルール・ブロックの現行情報がエディター領域に表示されます。

この後、以下の方法で追加のルール・ブロックを作成できます。

- 既存のルール・ブロックと対等のルール・ブロックを作成するには、該当する既存のルール・ブロックを右クリックし、「**兄弟の新規作成**」 → 「**ルール・ブロック**」をクリックします。

- 既存のルール・ブロック内にルール・ブロックを作成するには、該当する既存のルール・ブロックを右クリックし、「**子の新規作成**」 → 「**ルール・ブロック**」をクリックします。

また、エディターを使用して、ルール・セットおよび各ルール・ブロックの情報を更新できます。『ルールの作成』 に進みます。

ルールの作成

ここでは、ルールの作成方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

ルールは、ルール・ブロック内で作成する必要があります。ルールを作成するには、以下のステップを実行します。

1. 「アウトライン」ビューで、更新するルール・ブロックを右クリックします。
2. 「**子の新規作成**」をクリックし、作成するルールのタイプをクリックします。これにより、「アウトライン」ビューのルール・ブロック内にルールが表示され、ルールの現行情報がエディター領域に表示されます。

同じ方法で、ルールをさらにルール・ブロックに追加できます。また、エディターを使用して各ルールの情報を更新できます。『ルール・セットの検証』 に進みます。

ルール・セットの検証

ここでは、コンパイルする前にルール・セットを検証する方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

ルール・セットを検証するには、以下のステップを実行します。

1. 「アウトライン」ビューで、ルール、ルール・ブロック、またはルール・セットを右クリックします。
2. 「**ルール・セットの検証**」をクリックします。 検証が完了すると、問題が存在するかどうかを示すメッセージ・ウィンドウが表示されます。検証が正常に完了すると、「検証済み」という語が「ナビゲーター」ビュー内のルール・セットのファイル名の右側に表示されます。

検証が正常に完了した場合は、『ルール・セットのコンパイル』 に進みます。

ルール・セットのコンパイル

ここでは、ルール・セットのコンパイル方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

「ナビゲーター」または「アウトライン」ビューで、コンパイルするルール・セットを右クリックし、「**ルール・セットのコンパイル**」をクリックします。コンパイル・エラーがあれば、「問題」ビューに表示されます。

デフォルトでは、コンパイル・エラーがない場合、コンパイルされたルール・セットがデフォルトのファイル・タイプ acts として「ナビゲーター」ビューに表示されます。これにより、シリアライズ化されたルール・セットであることがわかります。また、「ナビゲーター」ビューでは、「コンパイル済み」という語がルール・セットのファイル名の右側に表示されます。

ルール・セットの更新

ここでは、ルール・セットの更新方法を説明します。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

「アウトライン」ビューで、更新する項目 (ルール、ルール・ブロック、またはルール・セット) をクリックします。エディター領域にその項目の現行情報が表示されます。また、エディターを使用してこの情報を更新できます。

既存のルール・ブロックまたはルールと対等のルール・ブロックまたはルールを作成するには、以下のステップを実行します。

1. 既存のルール・ブロックまたはルールを右クリックします。
2. 「**兄弟の新規作成**」および追加する項目 (ルール・ブロックまたはルール) をクリックします。

既存のルール・ブロック内にルール・ブロックまたはルールを作成するには、以下のステップを実行します。

1. 既存のルール・ブロックを右クリックします。
2. 「**子の新規作成**」および追加する項目 (ルール・ブロックまたはルール) をクリックします。

「アウトライン」ビューでその他の更新機能にアクセスするには、以下のステップを実行します。

1. 更新する項目 (ルール・ブロックまたはルール) を右クリックします。
2. 「**削除**」、「**コピー**」、または「**貼り付け**」などの機能のメニュー項目をクリックします。

ルール内の式にスニペットを組み込む

ルール内の式に、Eclipse ワークベンチ内の「断片」ビューからスニペットを組み込むことができます。

最初に、Eclipse ワークベンチを開きます (開いていない場合)。また、ルール・ビルダー・パースペクティブも開きます。

「断片」ビューがルール・ビルダー・パースペクティブに表示されます。スニペットは、その機能に基づいて複数のカテゴリーに分類されます。

「断片」ビューからスニペットを組み込むには、以下の手順を実行します。

1. スニペット・カテゴリーをクリックして、カテゴリー内のスニペットの名前を参照します。
2. 式に組み込むスニペットをクリックします。

3. スニペットをそれぞれの「**式**」フィールドにドラッグします。 コードが「**式**」フィールドのカーソル位置に配置されます。コードが、ルール・ライターからの入力 (特定のメッセージ・テキストまたは変数値など) を必要とする場合、コードが式に組み込まれる前にその入力を促すプロンプトが出されます。

44 ページの『ルール・セットの検証』 に進みます。

第 2 部 ルール・ライターのリファレンス

第 4 章 ルール・セット編成のサマリー

このリファレンスでは、ルール・セット、ルール・ブロック、および各タイプのルールのすべての言語エレメントをリストします。 ルール・セットをコーディングするためのクイック・リファレンスとして利用できます。

表 10 では、各言語エレメントの後に続く表記の意味を説明しています。*n* は、無制限の数値を表します。

表 10. 言語エレメントの指定回数を定義する表記の説明

表記	意味
(0, 1)	0 は、その言語エレメントがオプションであることを示します。1 は、コーディングされた場合、1 回のみ指定できることを示します。
(0, *n*)	0 は、その言語エレメントがオプションであることを示します。*n* は、コーディングされた場合、複数回指定できることを示します。
(1, 1)	最初の 1 は、その言語エレメントが必須であることを示します。 2 番目の 1 は、1 回のみ指定できることを示します。
(1, *n*)	1 は、その言語エレメントが必須であることを示します。*n* は、複数回指定できることを示します。
(2, *n*)	2 は、その言語エレメントを 2 回指定する必要があることを示します。*n* は、追加の指定ができることを示します。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

ルール・セットのサマリー

このサマリーでは、ルール・セットの言語エレメントをリストします。

ルール・セットのエレメント

<ruleSet> には、以下のエレメントが含まれます。

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

関連資料

50 ページの『ルール・ブロックのサマリー』
このサマリーでは、ルール・ブロックの言語エレメントをリストします。

ルール・ブロックのサマリー

このサマリーでは、ルール・ブロックの言語エレメントをリストします。

ルール・ブロックのエレメント

<ruleBlock> には、以下のエレメントが含まれます。

<comment> エレメント、<import> エレメント、および <variable> エレメントをコーディングする場合、これらはここに示した順序でコーディングする必要があります。その他のエレメントは、任意の順序でコーディングできます。

- <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)

関連資料

51 ページの『コレクション・ルールのサマリー』
このサマリーには、コレクション・ルールの言語エレメントをすべてリストします。

52 ページの『計算ルールのサマリー』
このサマリーには、計算ルールの言語エレメントをすべてリストします。

54 ページの『重複ルールのサマリー』
このサマリーには、重複ルールの言語エレメントをすべてリストします。

55 ページの『フィルター・ルールのサマリー』
このサマリーには、フィルター・ルールの言語エレメントをすべてリストします。

56 ページの『シーケンス・ルールのサマリー』
このサマリーには、シーケンス・ルールの言語エレメントをすべてリストします。

58 ページの『しきい値ルールのサマリー』
このサマリーには、しきい値ルールの言語エレメントをすべてリストします。

59 ページの『タイマー・ルールのサマリー』
このサマリーには、タイマー・ルールの言語エレメントをすべてリストします。

コレクション・ルールのサマリー

このサマリーには、コレクション・ルールの言語エレメントをすべてリストします。

ルールのエレメント

<collectionRule> には以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 3 つのエレメント (任意の順序) (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

計算ルールのサマリー

このサマリーには、計算ルールの言語エレメントをすべてリストします。

ルールのエレメント

<computationRule> には以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 3 つのエレメント (任意の順序) (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <computeFunction> (1, 1)
- <timeWindow> (1, 1)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)

– <action> (0, n)

重複ルールのサマリー

このサマリーには、重複ルールの言語エレメントをすべてリストします。

ルールのエレメント

<duplicateRule> には以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  – <comment> (0, 1)
  – <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  – <activationTime> (0, 1)
    - <start> (0, 1)
      • 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        – <dateTime>
        – <whenLoaded>
        – <inactiveWhenLoaded>
    - <stop> (0, 1)
      • 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 3 つのエレメント (任意の順序) (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- <timeWindow> (1, 1)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onNextEvent> (0, 1)
  - <action> (0, n)
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

フィルター・ルールのサマリー

このサマリーには、フィルター・ルールの言語エレメントをすべてリストします。

ルールのエレメント

<filterRule> には以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        - <dateTime>

        - <whenLoaded>
        - <inactiveWhenLoaded>
      - <stop> (0, 1)
        - 以下の 3 つのエレメントのいずれか 1 つ (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)

シーケンス・ルールのサマリー

このサマリーには、シーケンス・ルールの言語エレメントをすべてリストします。

ルールのエレメント

<sequenceRule> には、以下のエレメントが含まれます。

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

        - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
          - <dateTime>
          - <whenLoaded>
          - <inactiveWhenLoaded>
      - <stop> (0, 1)
        - 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 3 つのエレメント (任意の順序) (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>

- <computedValue>

- <timeWindow> (1, 1)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

しきい値ルールのサマリー

このサマリーには、しきい値ルールの言語エレメントをすべてリストします。

ルールのエレメント

<thresholdRule> には、以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 3 つのエレメント (任意の順序) (1, n):
    - <attributeAlias>
      - <eventAttribute> (2, n)
    - <attributeName>
    - <computedValue>
- 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
  - <booleanThreshold>
  - <computedThreshold>
  - <eventCountThreshold>
- <timeWindow> (1, 1)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onDetection> (0, 1)
  - <action> (0, n)
- <onTimeOut> (0, 1)
  - <action> (0, n)

タイマー・ルールのサマリー

このサマリーには、タイマー・ルールの言語エレメントをすべてリストします。

ルールのエレメント

<timerRule> には、以下のエレメントが含まれます。

- <comment> (0, 1)
- <variable> (0, n)
  - <comment> (0, 1)
  - <varInitializer> (1, 1)
- <activationInterval> (0, 1)
  - <activationTime> (0, 1)
    - <start> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (1, 1):
        - <dateTime>
        - <whenLoaded>
        - <inactiveWhenLoaded>
    - <stop> (0, 1)
      - 以下の 3 つのエレメントのいずれか 1 つ (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)
  - 以下の 2 つのエレメントのいずれか 1 つ (1, 1):
    - <timeInterval>
    - <runUntilDeactivated>
- <onTimeWindowComplete> (0, 1)
  - <action> (0, n)

第 5 章 言語エレメントのリファレンス

このリファレンスでは、Active Correlation Technology ルール言語の XML スキーマにおける言語エレメントの詳細を説明します。言語エレメントはアルファベット順にリストされています。各エレメントで使用可能な属性については、そのエレメントに関するトピック内で説明されています。

XML や、SGML および HTML などのその他のマークアップ言語において、エレメントは、開始タグ、終了タグ、関連する属性とその値、そして開始タグと終了タグの間に含まれる任意のテキストから構成される基本単位です。属性は、エレメントの特定の特性を定義するためにエレメントに対してコーディングされる、名前と値のペアです。属性には、その値として提供される情報 (数値情報、テキスト情報、ブール情報など) の種類を識別するデータ・タイプが指定されています。

XML では、ネーム・スペースは、固有の名前を提供してスキーマ内のエレメントおよびタイプ定義と関連付ける Uniform Resource ID (URI) です。 URI は、エレメントの定義を含む XML スキーマを示します。ネーム・スペースは、プレフィックス・ストリングとそれに続くコロンで指定されます。Active Correlation Technology ルール言語スキーマは、3 つの異なるファイルで定義され、以下の 3 つのネーム・スペースを使用します。

xsd: このネーム・スペースは、言語エレメントが標準の XML スキーマ (http://www.w3.org で説明されています) で定義されていることを示します。

br: このネーム・スペースは、言語エレメントが Active Correlation Technology のベース・ルール・セット・スキーマで定義されていることを示します。このスキーマは、com/ibm/correlation/ruleparser/xml/RuleSetBase.xsd サブディレクトリー内の ACTparser.jar ファイルにあります。例えば、`br:ruleSet` は RuleSetBase.xsd ファイルで定義されている ruleSet エレメントを指します。

act: このネーム・スペースは、言語エレメントが Active Correlation Technology 言語スキーマで定義されていることを示します。このスキーマは、com/ibm/correlation/ruleparser/xml/ACTL.xsd サブディレクトリー内の ACTparser.jar ファイルにあります。例えば、`act:ruleSet` は ACTL.xsd ファイルで定義されている ruleSet エレメントを指します。

ルール言語スキーマでは、言語エレメントは、以下の例のようにエレメントまたは複合タイプとして定義されます。

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

このスキーマでは、minOccurs および maxOccurs 属性が、言語エレメントの最小出現数と最大出現数をそれぞれ定義します。 62 ページの表 11 では、minOccurs および maxOccurs 属性に対するさまざまな値の意味を説明します。

表 11. 言語エレメントの出現数を定義するスキーマの属性

属性	属性値	意味
minOccurs	0	言語エレメントはオプションです。
minOccurs	1	言語エレメントは最低 1 回指定する必要があります。minOccurs 属性のデフォルト値は 1 です。
minOccurs	2	言語エレメントは最低 2 回指定する必要があります。
maxOccurs	1	言語エレメントを複数回指定することはできません。maxOccurs 属性のデフォルト値は 1 です。
maxOccurs	unbounded	言語エレメントは、何回でも指定できます。

action エレメント

<action> エレメントには、ルール応答アクションまたはライフ・サイクル・アクションのいずれかを定義する式が含まれます。

詳細

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<action> には以下の属性があります。

表 12. <action> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい
name	アクションを識別します。この ID は、トラブルシューティングを行う上で役立つことがあります。特定のルール応答アクションまたはライフ・サイクル・アクションに対して定義されたすべての <action> エレメントにおいて固有の名前である場合は、特に有用です。	xsd:NMTOKEN	いいえ

このエレメントを含むエレメント

<action> は以下のエレメントに含まれます。

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

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

このエレメントに含まれるエレメント

<action> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

activateOnEvent エレメント

<activateOnEvent> エレメントは、ルール、またはルール・インスタンス (<groupingKey> エレメントを使用して定義されるルールの場合) をアクティブ化できるイベントを定義します。

イベントは、以下の 3 つの方法で選択できます。

- <filteringPredicate> エレメントとともに 1 つ以上の <eventType> エレメントを使用する方法
- <filteringPredicate> エレメントを使用せずに 1 つ以上の <eventType> エレメントを使用する方法
- <eventType> エレメントを使用せずに <filteringPredicate> エレメントを使用する方法

ルールが非アクティブであり、<eventType> エレメントや <filteringPredicate> エレメントがコーディングされていない場合、発生するすべてのイベントが選択されます。

<eventType> エレメントをコーディングしないと、システム・パフォーマンスが低下する場合があります。

`Audit Failure` タイプのすべてのイベントを選択するとします。フィルター述部を使用して選択基準をさらに絞り込み、特定の値のイベント属性を持つイベントのみ選択されるようにすることができます。例えば、<eventType> エレメントをコーディングして `Audit Failure` タイプのすべてのイベントを選択し、<filteringPredicate> エレメントをコーディングして、値が `MyCriticalSystem` の hostname 属性を持つイベントのみを選択できます。

属性

<activateOnEvent> には属性がありません。

このエレメントを含むエレメント

<activateOnEvent> は以下のエレメントに含まれます。

- <activationInterval>
- <activationByGroupingKey>

このエレメントに含まれるエレメント

<activateOnEvent> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 13. <activateOnEvent> エレメントに含まれるエレメント

エレメント	必須/オプション
<eventType>	オプション。0 回以上指定できます。
<filteringPredicate>	オプション。指定できるのは 0 回または 1 回です。
<stopAfter>	このエレメントは、<activateOnEvent> エレメントが <activationByGroupingKey> エレメントに含まれる場合にのみ有効です。 オプション。指定できるのは 0 回または 1 回です。

activationByGroupingKey エレメント

<activationByGroupingKey> エレメントには、<groupingKey> エレメントで定義されているルール・インスタンスをアクティブ化および非アクティブ化できるイベントを指定するエレメントが含まれます。<groupingKey> エレメントは、フィルターおよびタイマー・ルールに対しては無効であるため、<activationByGroupingKey> エレメントはこれらのルールに適用されません。

詳細

<activationByGroupingKey> エレメントで提供される関数は、グループ化キーを定義するルールで使用できます。これを使用すると、グループ化キーに基づいてルール・インスタンスのアクティブ化および非アクティブ化を制御できます。<activationByGroupingKey> エレメントをコーディングすると、<activationByGroupingKey> 内の <activateOnEvent> の条件および <deactivateOnEvent> の条件に基づいて、各ルール・インスタンスを別個にアクティブ化または非アクティブ化できるようになります。

以下の例は、計算ルールにおける <activationByGroupingKey> エレメントの使用方法を示しています。

- 以下の計算ルールでは、`StockSharesTraded` タイプのイベントを受け入れます。これらのイベントは、多くの会社について取引されている株式数を示しています。
- グループ化キーは、イベントの stockSymbol 属性です。stockSymbol 属性の値は特定の会社の名前です。

- イベントの sharesTraded 属性の値は、個別の会社について取引された株式数です (会社名は stockSymbol 属性の値によって示されます)。
- 特定の会社に関して、10 分間に取引されたその会社の株式数を示すレポートが作成されます。ただし、計算ルールでこのレポートを作成可能にするには、事前に stockSymbol 属性の値として個別の会社の名前が指定された StartReporting タイプのイベントを受け取る必要があります。

```
<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>
```

属性

<activationByGroupingKey> には属性がありません。

このエレメントを含むエレメント

<activationByGroupingKey> は、以下のエレメントに含まれます。

- <activationInterval>

このエレメントに含まれるエレメント

<activationByGroupingKey> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 14. <activationByGroupingKey> エレメントに含まれるエレメント

エレメント	必須/オプション
<activateOnEvent>	オプション。指定できるのは 0 回または 1 回です。
<deactivateOnEvent>	オプション。指定できるのは 0 回または 1 回です。

<activationInterval> エレメントおよび <activationByGroupingKey> エレメント間の関係

<activateOnEvent> エレメントおよび <deactivateOnEvent> エレメントは、以下の両方のエレメントに含まれます。

- <activationInterval>
- <activationByGroupingKey> (<activationInterval> にも含まれる)

ルールの振る舞いは、現行のルール・アクティビティーと、<activationInterval> エレメントおよび <activationByGroupingKey> エレメントに含まれる <activateOnEvent> と <deactivateOnEvent> の定義間の相互作用に応じて異なる場合があります。以下の例は、これらの定義がどのように相互作用することがあるか示します。

この例では、保守モードのシステムのイベントを抑制し、保守期間の終了時点で、抑制されたイベント数のサマリー・レポートが提供されるよう重複ルールを定義しています。

デフォルトでは、グループ化キーを定義するルールは、すべてのグループ化キー値の処理を許可します。したがって、イベントがルールのイベント選択基準を満たすと、すべてのルール・インスタンスがアクティブになり、グループ化キーの任意の値に従ってこれらのイベントを受け入れられるようになります。ルールのアクティブ化間隔は、ルールにグループ化キーが含まれない場合と同じです。これは、ルールのイベント選択基準を満たすイベントは、基本的にすべて処理されるためです。

以下の例では、グループ化キーは `hostname` であり、<activationInterval> エレメント内の定義が以下のアクションを指定しています。

1. `StartMaintenanceModeAllHosts` タイプのイベントを受信したときは、すべてのルール・インスタンスをアクティブ化します。
2. 2 時間後、または `StopMaintenanceModeAllHosts` タイプのイベントを受信した時点で、すべてのルール・インスタンスを非アクティブ化します。

```
<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>
```

場合により、どのルール・インスタンスをいつアクティブにするか、制御する必要があるかもしれません。そのような場合は、<activationByGroupingKey> エレメントをコーディングします。

上記の例を拡張した以下の例では、グループ化キー値を使用して、処理可能なルール・インスタンスを選択する方法を示します。<activationByGroupingKey> エレメント内の定義は、以下のアクションを指定します。

1. 特定のホスト名について StartMaintenanceMode タイプのイベントが受信された場合、そのホスト名のルール・インスタンスの処理を許可します。
2. これらのルール・インスタンスを、アクティブ化の 2 時間後、またはそれぞれのホスト名について StopMaintenanceMode タイプのイベントが受信されたときに非アクティブ化します。

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

以下の文は、<activationByGroupingKey> エレメントをコーディングした場合に発生する状況を要約したものです。

- <activateOnEvent> エレメントが <activationByGroupingKey> エレメント内でコーディングされると、<activationByGroupingKey> <activateOnEvent> 条件と一致するイベントのグループ化キー値を共有するイベントのみが処理を許可されます。
- <deactivateOnEvent> エレメントが <activationByGroupingKey> エレメント内でコーディングされると、<activationByGroupingKey> <deactivateOnEvent> 条件と一致するイベントのグループ化キー値を共有するイベントの処理が許可されません。

さまざまなアクティブ化および非アクティブ化定義のルール状態に対する影響

表 15 および 70 ページの表 16 では、さまざまなアクティブ化および非アクティブ化定義がルールの状態にどのように影響するかを示しています。この表では、以下の規則が使用されます。

- *A* はアクティブ化イベントです。
- 『*A[x]*』 という表記では、*x* がグループ化キー値を示します。例えば、*A[1]* はグループ化キー値 1 を持つアクティブ化イベントです。
- *D* は非アクティブ化イベントです。
- 『*D[x]*』 という表記では、*x* がグループ化キー値を示します。例えば、*D[1]* はグループ化キー値 1 を持つ非アクティブ化イベントです。

表 15. アクティブ化定義の差異に基づくルール状態の変化

開始時のルール状態	場合によりルール状態が影響を受ける対象	終了時のルール状態
非アクティブ	<activationInterval> <activationTime> <start> で定義された時間	1. ルールはアクティブ化されます。 2. <onActivation> アクションが実行されます。 3. すべてのグループ化キー値が許可されます。
	activate() メソッド	
	<activationInterval> <activateOnEvent> で定義されたイベント *A*	
	<activationByGroupingKey> <activateOnEvent> で <stopAfter> *を使用せずに* 定義されたイベント *A[1]*	1. ルールはアクティブ化されます。 2. <onActivation> アクションが実行されます。 3. グループ化キー値 1 のみが許可されます。このルール・インスタンスについてルール・パターンが一致した場合、グループ化キー値 1 は許可されなくなります。
	<activateOnEvent> で <stopAfter> *を使用して* 定義されたイベント *A[2]*	1. ルールはアクティブ化されます。 2. <onActivation> アクションが実行されます。 3. グループ化キー値 2 のみが、<stopAfter> エレメントで指定された継続時間のみ許可されます。このルール・インスタンスのルール・パターンは、この継続時間中複数回一致することがあります。

表 15. アクティブ化定義の差異に基づくルール状態の変化 (続き)

開始時のルール状態	場合によりルール状態が影響を受ける対象	終了時のルール状態
• アクティブ • すべてのグループ化キー値の許可	<activationInterval> <activationTime> <start> で定義された時間	ルールの状態は変化しません。開始時のルール状態と同じです。
	activate() メソッド	
	<activationInterval> <activateOnEvent> で定義されたイベント *A*	
	<activationByGroupingKey> <activateOnEvent> で <stopAfter> を*使用せずに* 定義されたイベント *A[1]*	
	<activateOnEvent> で <stopAfter> を*使用して* 定義されたイベント *A[2]*	
• アクティブ • <activationByGroupingKey> <activateOnEvent> 定義に基づいてルール・インスタンスをトリガーしたグループ化キー値のみ許可	<activationInterval> <activationTime> <start> で定義された時間	ルールの状態は変化しません。開始時のルール状態と同じです。
	activate() メソッド	
	<activationInterval> <activateOnEvent> で定義されたイベント *A*	すべてのグループ化キー値が許可されます。
	<activationByGroupingKey> <activateOnEvent> で <stopAfter> を*使用せずに* 定義されたイベント *A[1]*	• 前に許可されていたグループ化キー値に加え、グループ化キー値 1 が許可されるようになります。 • このルール・インスタンスについてルール・パターンが一致した場合、グループ化キー値 1 は許可されなくなります。
	<activateOnEvent> で <stopAfter> を*使用して* 定義されたイベント *A[2]*	• 前に許可されていたグループ化キー値に加え、グループ化キー値 2 が許可されるようになります。 • この値は、<stopAfter> エレメントによって指定された継続時間のみ許可されます。 • このルール・インスタンスのルール・パターンは、この継続時間中複数回一致することがあります。
• アクティブ • <activationByGroupingKey> <deactivateOnEvent> 定義に基づいて許可されていない値を除きすべてのグループ化キー値を許可	<activationInterval> <activationTime> <start> で定義された時間	ルールの状態は変化しません。開始時のルール状態と同じです。
	activate() メソッド	
	<activationInterval> <activateOnEvent> で定義されたイベント *A*	すべてのグループ化キー値が許可されます。
	<activationByGroupingKey> <activateOnEvent> で <stopAfter> を*使用せずに* 定義されたイベント *A[1]*	前に許可されていたグループ化キー値に加え、グループ化キー値 1 が許可されるようになります。
	<activateOnEvent> で <stopAfter> を*使用して* 定義されたイベント *A[2]*	前に許可されていたグループ化キー値に加え、グループ化キー値 2 が許可されるようになります。

表 16. 非アクティブ化定義の差異に基づくルール状態の変化

開始時のルール状態	場合によりルール状態が影響を受ける対象	終了時のルール状態
非アクティブ	<activationInterval> <activationTime> <stop> で定義された時間	ルールの状態は変化しません。開始時のルール状態と同じです。
	deactivate() メソッド	
	<activationInterval> <deactivateOnEvent> で定義されたイベント *D*	
	<activationByGroupingKey> <deactivateOnEvent> で定義されたイベント *D[1]*	
	イベント *A[2]* について、<activationByGroupingKey> <activateOnEvent> <stopAfter> で定義された継続時間が終了します。	
• アクティブ • すべてのグループ化キー値の許可	<activationInterval> <activationTime> <stop> で定義された時間	1. すべてのルール・インスタンスが非アクティブ化されます。 2. <onDeactivation> アクションが実行されます。 3. ルールは非アクティブ化されます。
	deactivate() メソッド	
	<activationInterval> <deactivateOnEvent> で定義されたイベント *D*	
	<activationByGroupingKey> <deactivateOnEvent> で定義されたイベント *D[1]*	• グループ化キー値 1 は許可されなくなります。 • グループ化キー値 1 を持つルール・インスタンスがアクティブである場合、非アクティブ化されます。
	イベント *A[2]* について、<activationByGroupingKey> <activateOnEvent> <stopAfter> で定義された継続時間が終了します。	グループ化キー値 2 を持つルール・インスタンスが非アクティブ化されます。
• アクティブ • <activationByGroupingKey> <activateOnEvent> 定義に基づいてルール・インスタンスをトリガーしたグループ化キー値のみ許可	<activationInterval> <activationTime> <stop> で定義された時間	1. すべてのルール・インスタンスが非アクティブ化されます。 2. <onDeactivation> アクションが実行されます。 3. ルールは非アクティブ化されます。
	deactivate() メソッド	
	<activationInterval> <deactivateOnEvent> で定義されたイベント *D*	
	<activationByGroupingKey> <deactivateOnEvent> で定義されたイベント *D[1]*	• グループ化キー値 1 は許可されなくなります。 • グループ化キー値 1 を持つルール・インスタンスがアクティブである場合、非アクティブ化されます。
	イベント *A[2]* について、<activationByGroupingKey> <activateOnEvent> <stopAfter> で定義された継続時間が終了します。	• グループ化キー値 2 は許可されなくなります。 • グループ化キー値 2 を持つルール・インスタンスが非アクティブ化されます。

表 16. 非アクティブ化定義の差異に基づくルール状態の変化 (続き)

開始時のルール状態	場合によりルール状態が影響を受ける対象	終了時のルール状態
• アクティブ • <activationByGroupingKey> <deactivateOnEvent> 定義に基づいて許可されていない値を除きすべてのグループ化キー値を許可	<activationInterval> <activationTime> <stop> で定義された時間	1. すべてのルール・インスタンスが非アクティブ化されます。 2. <onDeactivation> アクションが実行されます。 3. ルールは非アクティブ化されます。
	deactivate() メソッド	
	<activationInterval> <deactivateOnEvent> で定義されたイベント *D*	
	<activationByGroupingKey> <deactivateOnEvent> で定義されたイベント *D[1]*	• グループ化キー値 1 は許可されなくなります。 • グループ化キー値 1 を持つルール・インスタンスがアクティブである場合、非アクティブ化されます。
	イベント *A[2]* について、<activationByGroupingKey> <activateOnEvent> <stopAfter> で定義された継続時間が終了します。	グループ化キー値 2 を持つルール・インスタンスが非アクティブ化されます。

activationInterval エレメント

<activationInterval> エレメントには、ルールがアクティブまたは非アクティブになるタイミングを定義するエレメントが含まれます。

詳細

ルールは、離散的な時点で、または特定のイベントによってアクティブ化または非アクティブ化できます。

ルールが離散的な時点および 特定のイベントでアクティブ化または非アクティブ化されるよう指定した場合、離散的な時点またはイベントを受信した時点のいずれか早い時点でルールがアクティブ化または非アクティブ化されます。ただしこの場合、ルールはそのライフ・サイクルをとおして、多くのイベントによってアクティブ化または非アクティブ化される可能性があります。例えば、ルールがイベントによってアクティブ化され、非アクティブ化され、定義された時点でアクティブ化され、再度非アクティブ化され、別のイベントによってアクティブ化されることもあります。

ビジネス環境においては、例えばビジネスに関する株式取引が開始されたことを示すイベントを受信したときにルールをアクティブ化する必要があるかもしれません。また、IT 環境においては、例えば 2005 年 10 月 29 日の 06:00 に保守期間を開始し、以下のいずれか早い時点で終了させる必要があるかもしれません。

- 2005 年 10 月 30 日 11:30
- 保守作業が完了したことを示すイベントの受信時

属性

<activationInterval> には属性がありません。

このエレメントを含むエレメント

<activationInterval> は以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<activationInterval> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 17. <activationInterval> エレメントに含まれるエレメント

エレメント	必須/オプション
<activationTime>	オプション。指定できるのは 0 回または 1 回です。
<activateOnEvent>	オプション。指定できるのは 0 回または 1 回です。
<deactivateOnEvent>	オプション。指定できるのは 0 回または 1 回です。
<activationByGroupingKey>	オプション。指定できるのは 0 回または 1 回です。

含まれるエレメント間の関係

<activationTime> エレメントに含まれる <start> エレメントおよび <stop> エレメントは、ルールをアクティブ化および非アクティブ化する静的メソッドです。これらのエレメントを使用することで、ルールを離散的な時点でアクティブ化または非アクティブ化できます。一方、<activateOnEvent> エレメントおよび <deactivateOnEvent> エレメントは、ルールをアクティブ化および非アクティブ化する動的メソッドです。これらのエレメントを使用することで、特定のイベントが発生した場合にルールをアクティブ化または非アクティブ化できます。例えば、ルールがまだアクティブ化されていない場合、<activateOnEvent> エレメントに定義された基準と一致するすべてのイベントによってルールがアクティブ化されます。また、ルールがまだ非アクティブ化されていない場合、<deactivateOnEvent> エレメントに定義された基準と一致するすべてのイベントによってルールが非アクティブ化されます。そのため、特定のイベントによって、ルールがアクティブ化または非アクティブ化されるタイミングの静的定義が変更されることがあります。

73 ページの表 18 では、以下のエレメントをさまざまに組み合わせてコーディングした場合に、ルールがいつどのようにアクティブ化または非アクティブ化されるかについて説明しています。

- <start>
- <stop>

- <activateOnEvent>
- <deactivateOnEvent>

表 18 では、*X* はルールをアクティブ化するイベントの名前、*Y* はルールを非アクティブ化するイベントの名前を示します。

<start> エレメントがまったくコーディングされていない場合、デフォルトの開始時刻は <whenLoaded> エレメントで定義されている時刻と同じです。

<stop> エレメントがまったくコーディングされていない場合、デフォルトの停止時刻は <never> エレメントで定義されている時刻と同じです。

表 18. <activationInterval> に含まれるエレメントをさまざまに組み合わせてコーディングした場合のルール・アクティビティー

<activationTime>		**<activateOnEvent>**	**<deactivateOnEvent>**	**ルール・アクティビティー**
<start>	**<stop>**			
<whenLoaded>	<never>			ルールはロード時にアクティブであり、Active Correlation Technology エンジンの実行中はアクティブな状態のままになります。
<whenLoaded>	<never>		*Y*	ルールはロード時にアクティブです。イベント *Y* がルールを非アクティブ化します。
<whenLoaded>	<never>	*X*	*Y*	ルールはロード時にアクティブです。イベント *Y* がルールを非アクティブ化し、イベント *X* がルールを再アクティブ化します。この非アクティブ化および再アクティブ化は、複数回発生することがあります。
<whenLoaded>	<after>			ルールはロード時にアクティブであり、指定された時間間隔の経過後に非アクティブ化されます。
<whenLoaded>	<dateTime>			ルールはロード時にアクティブであり、指定された日時に非アクティブ化されます。
<inactiveWhenLoaded>	<never>	*X*		ルールはロード時には非アクティブです。イベント *X* がルールをアクティブ化し、Active Correlation Technology エンジンの実行中はアクティブな状態のままになります。
<inactiveWhenLoaded>	<never>	*X*	*Y*	ルールはロード時には非アクティブです。イベント *X* がルールをアクティブ化し、イベント *Y* がルールを非アクティブ化します。このアクティブ化および非アクティブ化は、複数回発生することがあります。
<dateTime>	<dateTime>			ルールは指定された日時にアクティブ化され、指定された日時に非アクティブ化されます。
<dateTime>	<dateTime>	*X*	*Y*	ルールは指定された日時にアクティブ化され、指定された日時に非アクティブ化されます。イベント *X* がルールをアクティブ化し、イベント *Y* がルールを非アクティブ化します。イベント *X* および *Y* は、ルールを複数回アクティブ化および非アクティブ化できます。
<dateTime>	<never>			ルールは指定された日時にアクティブ化され、Active Correlation Technology エンジンの実行中はアクティブな状態のままになります。
<dateTime>	<never>		*Y*	ルールは指定された日時にアクティブ化されます。イベント *Y* がルールを非アクティブ化します。
<dateTime>	<never>	*X*	*Y*	ルールは指定された日時にアクティブ化されます。イベント *Y* がルールを非アクティブ化し、イベント *X* がルールを再アクティブ化します。この非アクティブ化および再アクティブ化は、複数回発生することがあります。
<dateTime>	<after>			ルールは指定された日時にアクティブ化され、指定された時間間隔の経過後に非アクティブ化されます。

表 18. <activationInterval> に含まれるエレメントをさまざまに組み合わせてコーディングした場合のルール・アクティビティー (続き)

<activationTime>		<activateOnEvent>	<deactivateOnEvent>	ルール・アクティビティー
<start>	<stop>			
<dateTime>	<after>	*X*	*Y*	ルールは指定された日時にアクティブ化され、指定された時間間隔の経過後に非アクティブ化されます。イベント *X* がルールをアクティブ化し、イベント *Y* がルールを非アクティブ化します。このアクティブ化および非アクティブ化は、複数回発生することがあります。

activationTime エレメント

<activationTime> エレメントは、ルールがアクティブ化または非アクティブ化される離散的な時点を定義します。

属性

<activationTime> には属性がありません。

このエレメントを含むエレメント

<activationTime> は以下のエレメントに含まれます。

- <activationInterval>

このエレメントに含まれるエレメント

<activationTime> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 19. <activationTime> エレメントに含まれるエレメント

エレメント	必須/オプション
<start>	オプション。指定できるのは 0 回または 1 回です。
<stop>	オプション。指定できるのは 0 回または 1 回です。

after エレメント

<after> エレメントは、ルールがアクティブ化されてからアクティブ状態のままになる継続期間を指定します。この継続時間の経過後、ルールは非アクティブ化されます。

属性

<after> には以下の属性があります。

表 20. <after> エレメントの属性

名前	説明	データ・タイプ	必須?
duration	継続時間の時間量を指定します。この属性のデータ・タイプは、unit 属性の値によって異なります。	• unit 属性の値が ISO-8601 の場合、データ・タイプは xsd:duration です。 • unit 属性の値が milliseconds の場合、データ・タイプは xsd:positiveInteger です。	はい
unit	使用する時間単位を指定します。この属性の有効値は以下のとおりです。 • ISO-8601 • milliseconds	xsd:string	はい

継続時間における ISO 8601 規格の使用

unit 属性の値として ISO-8601 がコーディングされている場合、duration 属性の値が ISO 8601 規格に従ってコーディングされ、継続時間を 1 つのストリングとして指定することを示します。標準の XML スキーマのデータ・タイプ仕様では、ISO 8601 を使用してデータ・タイプ duration を提供します。このデータ・タイプについては、http://www.w3.org/TR/xmlschema-2/#duration で詳しく説明されています。

標準の XML スキーマの duration データ・タイプのフォーマットは、以下のストリングです。

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

- ストリングは常に文字 P で始まります。
- *n*Y は、年数を示します。 1 年は、365 日と同じです。したがって、1Y とコーディングするのは、365D とコーディングするのと同じです。
- *n*M は月数を示します。 1 カ月は 30 日と同じです。したがって、1M とコーディングするのは、30D とコーディングするのと同じです。
- *n*D は日数を示します。
- T は時間の単位 (時間、分、および秒) と日の単位 (年、月、日) を分離する分離文字です。時間の単位は必ず T の後に続きます。
- *n*H は時間数を示します。
- *n*M は分数を示します。
- *n*S は秒数を示します。

以下はフォーマットの例です。

- P5DT12H は 5.5 日です。
- PT59M59S は 59 分 59 秒です。
- P1M は 1 カ月です。

このエレメントを含むエレメント

<after> は以下のエレメントに含まれます。

- <stop>

このエレメントに含まれるエレメント

<after> に含まれるエレメントはありません。

attributeAlias エレメント

<attributeAlias> エレメントは、同じ意味を持つものの、さまざまなイベントにおいて異なる名前を持つイベント属性を関連付けるための別名を提供します。例えば、3 つの異なるイベントにおいて、イベントの発生元となるシステムの名前を示すイベント属性に対して、3 つの異なる名前、host、hostname、および source が使用されることも考えられます。<attributeAlias> エレメントには、グループ化キーに対して、1 つのイベント属性として関連付ける必要のある個々のイベント属性を記述する <eventAttribute> エレメントが含まれます。

詳細

<attributeAlias> エレメントとその aliasName 属性は、グループ化キーというコンテキストにおいてのみ有効です。このエレメントとその属性は、<computedValue> エレメント内の式を含め、いずれの式でも参照できません。

属性

<attributeAlias> には以下の属性があります。

表 21. <attributeAlias> エレメントの属性

名前	説明	データ・タイプ	必須?
aliasName	<eventAttribute> エレメントに記述され、グループ化キーに対して 1 つのイベント属性として関連付けられるイベント属性の名前を定義します。この名前はルール内で固有である必要があります。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<attributeAlias> は以下のエレメントに含まれます。

- <groupingKey>

このエレメントに含まれるエレメント

<attributeAlias> には以下のエレメントが含まれます。

表 22. <attributeAlias> エレメントに含まれるエレメント

エレメント	必須/オプション
<eventAttribute>	このエレメントは 2 回指定する必要があります。追加の指定も可能です。

attributeName エレメント

<attributeName> エレメントには、グループ化キーの一部である特定のイベント属性の名前が含まれます。この名前は、getAttribute メソッド呼び出しにおいて act_event 変数に対して使用される名前と一致する必要があります。

属性

<attributeName> には属性がありません。

このエレメントを含むエレメント

<attributeName> は以下のエレメントに含まれます。

- <groupingKey>

このエレメントに含まれるエレメント

<attributeName> に含まれるエレメントはありません。

booleanThreshold エレメント

<booleanThreshold> エレメントは、しきい値ルールに対してのみ有効です。このエレメントには、各イベントを受信したときに呼び出される式が含まれます。式は、現行イベントと、ルールによって受け入れられたその他のイベントに基づいて、しきい値を計算または比較します。式は、しきい値に達したかどうかを示すブール値 `true` または `false` を戻します。

詳細

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<booleanThreshold> には以下の属性があります。

表 23. <booleanThreshold> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<booleanThreshold> は以下のエレメントに含まれます。

- <thresholdRule>

このエレメントに含まれるエレメント

<booleanThreshold> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

collectionRule エレメント

<collectionRule> エレメントは、コレクション・パターンに従ってルールを定義します。

属性

<collectionRule> には以下の属性があります。

表 24. <collectionRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は `false` で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

このエレメントを含むエレメント

<collectionRule> は以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<collectionRule> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 25. <collectionRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	オプション。指定できるのは 0 回または 1 回です。
<groupingKey>	オプション。指定できるのは 0 回または 1 回です。
<timeWindow>	必須。指定できるのは 1 回のみです。
<onTimeWindowComplete>	オプション。指定できるのは 0 回または 1 回です。

関連概念

12 ページの『コレクション・パターン』
コレクション・ルールは、コレクション・パターンによって定義されます。このルールは時間間隔内に選択されたイベントのグループを収集します。これはステートフル・ルールです。

comment エレメント

<comment> エレメントには、関数の説明の他、使用されているルール・セット、ルール・ブロック、ルール、または変数の目的を含めることができます。

属性

<comment> には属性がありません。

このエレメントを含むエレメント

<comment> は以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<comment> に含まれるエレメントはありません。

computationRule エレメント

<computationRule> エレメントは、計算パターンに従ってルールを定義します。

属性

<computationRule> には以下の属性があります。

表 26. <computationRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は `false` で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

このエレメントを含むエレメント

<computationRule> は以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<computationRule> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 27. <computationRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	オプション。指定できるのは 0 回または 1 回です。
<groupingKey>	オプション。指定できるのは 0 回または 1 回です。
<computeFunction>	必須。指定できるのは 1 回のみです。
<timeWindow>	必須。指定できるのは 1 回のみです。
<onTimeWindowComplete>	オプション。指定できるのは 0 回または 1 回です。

関連概念

12 ページの『計算パターン』
計算ルールは、計算パターンによって定義されます。このルールは、時間間隔内で各イベントを受信するごとに、収集したイベントに対して式を使用して計算を適用します。これはステートフル・ルールです。

computedThreshold エレメント

<computedThreshold> エレメントは、しきい値ルールに対してのみ有効です。このエレメントには、各イベントが受信されるごとに呼び出され、現行イベントと、ルールのイベント選択基準に一致したその他のイベントに基づいてしきい値を計算する式が含まれます。この式は、ルールに対して定義された変数に保管される、計算されたしきい値を戻します。ルールはその後、計算されたしきい値を使用して、定義済みのしきい値との比較を行います。

詳細

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<computedThreshold> には以下の属性があります。

表 28. <computedThreshold> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい
threshold	到達すべきしきい値を定義します。この定義済みしきい値は、ルール変数に対して有効なデータ・タイプに変換できる数値のストリング表記でなければなりません。	xsd:string	はい

表 28. <computedThreshold> エレメントの属性 (続き)

名前	説明	データ・タイプ	必須?
assignTo	この式から戻される計算されたしきい値を保持する変数名を示します。この変数は、<variable> エレメントを使用して、ルールに対して、ルール・セット、ルール・ブロック、またはルール・レベルで事前に定義されている必要があります。以下のいずれかの数値データ・タイプとして定義されている必要があります。 • java.lang.Double • java.lang.Float • java.lang.Integer • java.lang.Long • java.lang.String 変数がルール・セット・レベルまたはルール・ブロック・レベルで定義されている場合、ルール・パターンに一致した後は再初期化されません。	xsd:NMTOKEN	はい
thresholdComparison	計算されたしきい値と定義済みしきい値の比較に使用される演算子を定義します。この演算子の有効値は以下のとおりです。 • `lessThan` • `lessThanOrEqualTo` • `greaterThan` • `greaterThanOrEqualTo`	xsd:string	はい

このエレメントを含むエレメント

<computedThreshold> は以下のエレメントに含まれます。

- <thresholdRule>

このエレメントに含まれるエレメント

<computedThreshold> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

computedValue エレメント

<computedValue> エレメントには、ルールがイベントを受信したときに実行され、イベントの 1 つ以上の属性の値に基づくストリング値を作成する式が含まれます。このストリング値はその後、グループ化キーで使用できます。

詳細

場合により、ルール・ライターはグループ化キーで以下のような項目を使用する必要があるかもしれません。

- イベント属性値のサブストリング。例えば、イベント属性値に組み込み IP アドレスが含まれている場合、<computedValue> エレメントに含まれている式を使用して、グループ化キーで使用できるよう、その IP アドレスを固有値として抽出できます。
- 複数の異なるイベント属性の値のサブストリング。例えば、<computedValue> エレメント内の式を使用してサブストリングを抽出し、それらを組み合わせてグループ化キーで使用する固有値を作成できます。

<computedValue> エレメント内の式がヌル値を戻した場合、ルールは、このヌル値を欠落した属性値として取り扱います。

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<computedValue> には以下の属性があります。

表 29. <computedValue> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<computedValue> は以下のエレメントに含まれます。

- <groupingKey>

このエレメントに含まれるエレメント

<computedValue> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

computeFunction エレメント

<computeFunction> エレメントは、計算ルールに対してのみ有効です。このエレメントには、各イベントが受信されるごとに呼び出され、ルールに対して定義されている変数に保管される値を戻す式が含まれます。この式で戻される値は、<computeFunction> エレメントの assignTo 属性で指定されている変数のデータ・タイプと一致する必要があります。

詳細

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<computeFunction> には以下の属性があります。

表 30. <computeFunction> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい
assignTo	この式から戻される値を保持する変数名を示します。この変数は、<variable> エレメントを使用して、ルールに対して、ルール・セット、ルール・ブロック、またはルール・レベルで事前に定義されている必要があります。変数がルール・セット・レベルまたはルール・ブロック・レベルで定義されている場合、ルール・パターンに一致した後は再初期化されません。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<computeFunction> は以下のエレメントに含まれます。

- <computationRule>

このエレメントに含まれるエレメント

<computeFunction> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

dateTime エレメント

<dateTime> エレメントは、ルールがアクティブ化または非アクティブ化される日時を指定します。ただし、ルールがアクティブ化または非アクティブ化されるのは、指定された時刻より前に、実行中の Active Correlation Technology エンジンにルールがロードされている場合に限られます。

詳細

アクティブ化の指定時刻より前に、実行中の Active Correlation Technology エンジンにルールがロードされていない場合、ルールがアクティブ化されることはありません。非アクティブ化の指定時刻より前に、実行中の Active Correlation Technology エンジンにルールがロードされていない場合、ルールは <start>エレメントで定義されている状態に設定され、<stop> エレメントによって非アクティブ化されることはありません。

<dateTime> エレメントの内容は、標準 XML スキーマの `dateTime` データ・タイプのフォーマットに従ったストリングでなければなりません。例えば、`dateTime` は以下の形式の有限長文字シーケンスから構成されます。

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

- *yyyy* は年を表す 4 桁以上の数表示です。4 桁より多い場合は、先行ゼロが禁止され、`0000` が禁止されます。
- 以降の「-」は日付部分の各部分間の分離文字です。
- 最初の *mm* は、`01` で始まる、月を表す 2 桁の数表示です。
- *dd* は、`01` で始まる、日付を表す 2 桁の数表示です。
- `T` は、その後に時刻が続くことを示す分離文字です。
- *hh* は、`00` で始まり `23` で終わる、24 時間制の時刻を表す 2 桁の数表示です。
- `:` は、時刻部分の各部分間の分離文字です。
- 2 番目の *mm* は、`00` で始まり `59` で終わる、分を表す 2 桁の数表示です。
- *ss* は、`00` で始まり `59` で終わる、整数秒を表す 2 桁の数表示です。
- '.' *s*+ は、小数秒を表します (指定されている場合)。
- *zzzzzz* は、時間帯を表します (指定されている場合)。時間帯は、(('+' | '-') *hh* ':' *mm*) | 'Z' という形式の有限長文字シーケンスから構成されます。各部分について以下に示します。
  - '+' は、負でない継続時間を表します (指定されている場合)。'-' とともには指定できません。
  - '-' は、正でない継続時間を表します (指定されている場合)。'+' とともには指定できません。
  - *hh* は、`00` で始まり `14` で終わる、時間を表す 2 桁の数表示です。
  - *mm* は、`00` で始まり `59` で終わる、分を表す 2 桁の数表示です。ただし、時間の値が `14` である場合、分の値は `00` でなければなりません。
  - Z は UTC の省略表現 (`+00:00` または `-00:00`) です。それ以外の時間帯エレメントは指定できません。

以下に、<dateTime> エレメントの内容の例を 2 つ示します。

- `2005-06-01T13:05:06.07` は、2005 年 6 月 1 日の地方時の午後 1:05 を 6 秒と 100 分の 7 秒経過した時点を表します。
- `2005-06-01T13:05:06.07Z` は、2005 年 6 月 1 日の UTC 時刻の午後 1:05 を 6 秒と 100 分の 7 秒経過した時点を表します。これは、2005 年 6 月 1 日の EDT 時刻の午前 9:05 を 6 秒と 100 分の 7 秒経過した時点 (つまり `2005-06-01T09:05:06.07-04:00`) に相当します。

属性

<dateTime> には属性がありません。

このエレメントを含むエレメント

<dateTime> は以下のエレメントに含まれます。

- <start>
- <stop>

このエレメントに含まれるエレメント

<dateTime> に含まれるエレメントはありません。

deactivateOnEvent エレメント

<deactivateOnEvent> エレメントは、ルール、またはルール・インスタンス (<groupingKey> エレメントを使用して定義されるルールの場合) を非アクティブ化できるイベントを定義します。

イベントは、以下の 3 つの方法で選択できます。

- <filteringPredicate> エレメントとともに 1 つ以上の <eventType> エレメントを使用する方法
- <filteringPredicate> エレメントを使用せずに 1 つ以上の <eventType> エレメントを使用する方法
- <eventType> エレメントを使用せずに <filteringPredicate> エレメントを使用する方法

ルールがアクティブであり、<eventType> エレメントや <filteringPredicate> エレメントがコーディングされていない場合、発生するすべてのイベントが選択されます。

<eventType> エレメントをコーディングしないと、システム・パフォーマンスが低下する場合があります。

`Audit Failure` タイプのすべてのイベントを選択するとします。フィルター述部を使用して選択基準をさらに絞り込み、特定の値のイベント属性を持つイベントのみ選択されるようにすることができます。例えば、<eventType> エレメントをコーディングして `Audit Failure` タイプのすべてのイベントを選択し、<filteringPredicate> エレメントをコーディングして、値が `MyCriticalSystem` の hostname 属性を持つイベントのみを選択できます。

属性

<deactivateOnEvent> には属性がありません。

このエレメントを含むエレメント

<deactivateOnEvent> は以下のエレメントに含まれます。

- <activationInterval>
- <activationByGroupingKey>

このエレメントに含まれるエレメント

<deactivateOnEvent> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 31. <deactivateOnEvent> エレメントに含まれるエレメント

エレメント	必須/オプション
<eventType>	オプション。0 回以上指定できます。
<filteringPredicate>	オプション。指定できるのは 0 回または 1 回です。

duplicateRule エレメント

<duplicateRule> エレメントは、重複パターンに従ってルールを定義します。

属性

<duplicateRule> には以下の属性があります。

表 32. <duplicateRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は `false` で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

このエレメントを含むエレメント

<duplicateRule> は以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<duplicateRule> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 33. <duplicateRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	オプション。指定できるのは 0 回または 1 回です。
<groupingKey>	オプション。指定できるのは 0 回または 1 回です。
<timeWindow>	必須。指定できるのは 1 回のみです。
<onDetection>	オプション。指定できるのは 0 回または 1 回です。
<onNextEvent>	オプション。指定できるのは 0 回または 1 回です。
<onTimeWindowComplete>	オプション。指定できるのは 0 回または 1 回です。

関連概念

13 ページの『重複パターン』
重複ルールは、重複パターンによって定義されます。重複ルールでは、指定された時間間隔内に受け入れられた 2 番目以降のイベントをカウントしますが、これらのイベントのルール・セット処理をスキップします。これはステートフル・ルールです。

eventAttribute エレメント

<eventAttribute> エレメントは、イベント・タイプとイベント属性を、<attributeAlias> エレメントで定義される属性別名の一部として関連付ける方法を提供します。

属性

<eventAttribute> には以下の属性があります。

表 34. <eventAttribute> エレメントの属性

名前	説明	データ・タイプ	必須?
type	イベントのタイプ名を定義します。これは、<eventType> エレメントのタイプ属性に使用される名前と同じです。	xsd:NMTOKEN	はい

表 34. <eventAttribute> エレメントの属性 (続き)

名前	説明	データ・タイプ	必須?
attributeName	属性別名を使用して他のイベント属性と関連付けられているイベント属性の完全修飾名を指定します。この名前は、getAttribute メソッドを呼び出すために act_event 変数で使用される名前と一致する必要があります。	xsd:string	はい

このエレメントを含むエレメント

<eventAttribute> は以下のエレメントに含まれます。

- <attributeAlias>

このエレメントに含まれるエレメント

<eventAttribute> に含まれるエレメントはありません。

eventCountThreshold エレメント

<eventCountThreshold> エレメントはしきい値ルールに対してのみ有効です。このエレメントは、特定の期間内にイベント選択基準と一致する必要のあるイベント数を定義します。また、<eventCountThreshold> エレメントは、時間枠として指定可能な 2 つの時間間隔モード (固定またはスライディング) のいずれかも指定します。

詳細

固定間隔

固定間隔は、イベント選択基準に一致する最初のイベントを受信したときに始まり、以下のいずれかが発生した時点で終了します。

- 指定された継続時間内にルールがそのしきい値に達した。
- 指定された継続時間が経過した。

スライディング間隔

スライディング間隔は、イベント選択基準と一致する最初のイベントを受信したときに始まります。ただし、ルールがしきい値に達することなく指定された継続時間が経過した場合、時間枠の開始時刻は、新規の「最初の」イベント (通常は次に受け入れられるイベント) の受信時刻に調整 (スライド) されます。このようにして、スライディング間隔は、以下のいずれかが発生する時点まで継続して調整されます。

- 指定された継続時間内にルールがそのしきい値に達した。
- 時間枠を開始するイベントの受信後、指定された継続時間内に後続のイベントを受信しなかった。

時間枠を開始するイベント (新規の「最初の」イベントとなる) は、「受信時刻をルールの時間間隔の継続時間に追加すると、現行時刻よりも後になる」という基準に一致する受信時刻を持つイベントです。以下に、この基準を式の形式で示します。

イベントの受信時刻 + ルールの時間間隔の継続時間 > 現行時刻

このようなイベントが存在しない場合、スライディング間隔はそれ以上の時間調整が不可能になり、間隔は終了します。

しきい値ルールは、しきい値に達するか、期間が終了するまで、受け入れられた各イベントをカウントします。次に、ルールは必要に応じて <onDetection> エレメントまたは <onTimeOut> エレメント内で定義されているアクションを実行します。

<onDetection> アクション
これらのアクションは、イベント・カウントが、<eventCountThreshold> エレメントのしきい値属性で定義されている値と同じ場合 (しきい値に達したことを示します) に実行されます。

<onTimeOut> アクション
これらのアクションが実行されるタイミングは、時間間隔モードが固定かスライディングかによって異なります。

固定モード
固定モードの場合、これらのアクションは時間枠の有効期限が切れたときに実行されます。

スライディング・モード
スライディング・モードの場合、時間枠を開始するイベントの受信後、指定された継続時間内に後続のイベントが受信されなかった場合に、これらのアクションが実行されます。つまり、ルールの時間間隔の継続時間に追加したときに、現行時刻より後になる受信時刻を持つイベントが受信されなかったことになります。

時間枠の時間間隔モードは、<eventCountThreshold> エレメントの timeIntervalMode 属性で定義されます。以下のシナリオは、指定可能な 2 つの時間間隔モードの振る舞いと相違点を示します。

固定モードおよびスライディング・モードを示すシナリオ

ルールが、イベント選択基準に一致する 4 つのイベントをそれぞれ 8:00、8:04、8:06、および 8:07 という時間に受信すると想定します。イベント・カウントのしきい値は 3 であり、時間枠の継続時間は 5 分です。

`fixed` (固定) モードのルールの振る舞い
この時間間隔モードでは、しきい値ルールは 8:00 に処理を開始し、5 分間に 2 つのイベントのみを受信するため、8:05 に <onTimeOut> アクションを実行します。したがって、時間枠内ではしきい値に達しません。しきい値ルールは、3 番目のイベントを 8:06 に受信したときに再度処理を開始し、5 分間に 2 つのイベントのみを受信するため、8:11 に <onTimeOut> アクションを実行します。

固定モードは静的です。

`sliding` (スライディング) モードのルールの振る舞い
この時間間隔モードでは、しきい値ルールは 8:00 に処理を開始します。時間枠が完了するスケジュールとなっている 8:05 に、ルールは 2 つのイベントのみを受信していることを判別します。この場合、ルールは 8:00 に受信したイベントを破棄し、8:09 に終了する期間を再計算します (最初のイベントが 8:04 に受信したイベントになったため)。ルールが 8:07 にイベント

を受信したときに <onDetection> アクションが実行されます。最新の時間枠 (8:04 から 8:09) 内でしきい値 (8:04、8:06、および 8:07 の 3 つのイベント) に達したためです。

スライディング・モードは、時間枠内でしきい値に達することを試行して、開始時刻を継続的に調整 (スライド) するという点において動的です。

ここで、ルールがイベント選択基準に一致する 4 つのイベントをそれぞれ 8:00、8:04、8:06、および 8:10 という時間に受信すると想定します。イベント・カウントのしきい値は 3 であり、時間枠の継続時間は 5 分です。

sliding (スライディング) モードのルールの振る舞い

この場合、しきい値ルールは、8:00 に処理を開始します。時間枠が完了するスケジュールとなっている 8:05 に、ルールは 2 つのイベントのみを受信していることを判別します。この場合、ルールは 8:00 に受信したイベントを破棄し、8:09 に終了する期間を再計算します (最初のイベントが 8:04 に受信したイベントになったため)。

新たに時間枠が完了するスケジュールとなった 8:09 に、ルールは 2 つのイベントのみを受信していることを判別します。この場合、ルールは 8:04 に受信したイベントを破棄し、8:11 に終了する期間を再計算します (最初のイベントが 8:06 に受信したイベントになったため)。

新たに時間枠が完了するスケジュールとなった 8:11 に、ルールは 2 つのイベントのみを受信していることを判別します。この場合、ルールは 8:06 に受信したイベントを破棄し、8:15 に終了する期間を再計算します (最初のイベントが 8:10 に受信したイベントになったため)。

新たに時間枠が完了するスケジュールとなった 8:15 に、ルールは時間枠を開始した 8:10 のイベント以降イベントが受信されていないことを判別します。ルールはここで <onTimeOut> アクションを実行します。

属性

<eventCountThreshold> には以下の属性があります。

表 35. <eventCountThreshold> エレメントの属性

名前	説明	データ・タイプ	必須?
threshold	特定の時間内にイベント選択基準と一致する必要のあるイベント数を定義します。これは、到達すべきイベント・カウントのしきい値です。この値は正の整数でなければなりません。	xsd:positiveInteger	はい
timeIntervalMode	時間枠の時間間隔が固定かスライディングかを定義します。この属性の有効値は以下のとおりです。 • `fixed` (デフォルト値) • `sliding`	xsd:string	いいえ

このエレメントを含むエレメント

<eventCountThreshold> は以下のエレメントに含まれます。

- <thresholdRule>

このエレメントに含まれるエレメント

<eventCountThreshold> に含まれるエレメントはありません。

eventSelector エレメント

<eventSelector> エレメントは、ルールで処理するために選択されるイベントを定義します。

詳細

イベントは、以下の 3 つの方法で選択できます。

- <filteringPredicate> エレメントとともに 1 つ以上の <eventType> エレメントを使用する方法
- <filteringPredicate> エレメントを使用せずに 1 つ以上の <eventType> エレメントを使用する方法
- <eventType> エレメントを使用せずに <filteringPredicate> エレメントを使用する方法

ルールにすべてのイベントを処理させるような特別な場合は、以下のオプションがあります。

- <eventSelector> エレメントをコーディングしない。
- エレメントを含まない <eventSelector> エレメントをコーディングする。

<eventType> エレメントをコーディングしないと、システム・パフォーマンスが低下する場合があります。

`Audit Failure` タイプのすべてのイベントを選択するとします。フィルター述部を使用して選択基準をさらに絞り込み、特定の値のイベント属性を持つイベントのみ選択されるようにすることができます。例えば、<eventType> エレメントをコーディングして `Audit Failure` タイプのすべてのイベントを選択し、<filteringPredicate> エレメントをコーディングして、値が `MyCriticalSystem` の hostname 属性を持つイベントのみを選択できます。

属性

<eventSelector> には以下の属性があります。

表 36. <eventSelector> エレメントの属性

名前	説明	データ・タイプ	必須?
alias	この属性は、複数の <eventSelector> エレメントを持つ唯一のルールであるシーケンス・ルール内でのみ有効です。この属性は、シーケンス・ルール内の特定のイベント・セレクターで選択されたイベントに固有の名前を付けます。この別名は、フィルター述部およびアクションで、そのイベントへのアクセスに使用できます。	xsd:NMTOKEN	いいえ

このエレメントを含むエレメント

<eventSelector> は以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<eventSelector> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 37. <eventSelector> エレメントに含まれるエレメント

エレメント	必須/オプション
<eventType>	オプション。0 回以上指定できます。
<filteringPredicate>	オプション。指定できるのは 0 回または 1 回です。

eventType エレメント

<eventType> エレメントは、ルールで処理するために選択されるイベント・タイプ、またはルールをアクティブ化または非アクティブ化するイベント・タイプを定義します。

属性

<eventType> には以下の属性があります。

表 38. <eventType> エレメントの属性

名前	説明	データ・タイプ	必須?
type	イベント・タイプを定義します。 コモン・ベース・イベント仕様に準拠するイベントの場合、この名前は extensionName 属性の値です。 IBM Tivoli Enterprise Console® イベントの場合、この名前は、BAROC ファイルで定義されるイベント・クラス名です。 他のフォーマットに基づいたイベントは、別の属性を使用してイベント・タイプを指定する場合があります。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<eventType> は以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<eventType> に含まれるエレメントはありません。

filteringPredicate エレメント

<filteringPredicate> エレメントには、ルールによる処理のため、またはルールをアクティブ化または非アクティブ化するために選択されるイベントをさらに限定する式が含まれます。したがって、<eventType> エレメントを使用してイベント・タイプのみでフィルタリングする場合よりも、より包括的にイベントをフィルタリングできます。

詳細

式は条件を定義し、ブール値 `true` (条件と一致する場合) または `false` (条件と一致しない場合) のいずれかを戻します。

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<filteringPredicate> には以下の属性があります。

表 39. <filteringPredicate> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<filteringPredicate> は以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<filteringPredicate> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

filterRule エレメント

<filterRule> エレメントは、フィルター・パターンに従ってルールを定義します。

属性

<filterRule> には以下の属性があります。

表 40. <filterRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい

表 40. <filterRule> エレメントの属性 (続き)

名前	説明	データ・タイプ	必須?
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は false で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

このエレメントを含むエレメント

<filterRule> は以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<filterRule> には以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 41. <filterRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	オプション。指定できるのは 0 回または 1 回です。
<onDetection>	オプション。指定できるのは 0 回または 1 回です。

関連概念

14 ページの『フィルター・パターン』
フィルター・ルールは、フィルター・パターンによって定義されます。このルールは、イベントを受け入れるときに特定のアクションを実行します。このルールは単一イベントに対してのみ機能するため、ステートレス・ルールです。

groupingKey エレメント

通常、各アクティブ・ルールには、Active Correlation Technology エンジン内で実行中の 1 つのルール・インスタンス (またはコピー) が存在します。ただし、多くの場合異なるリソースのグループに関連している異なるイベントのグループに、同じルールが必要な場合があります。グループ化キーは、選択されたイベントを、グループとして固有の処理を行う目的で別々のグループに分離するために使用できる、1 つ以上のイベント属性またはイベント属性の一部です。<groupingKey> エレメン

トは、ルールに対してグループ化キーを定義します。<groupingKey> エレメントの目的は、ルールに対して、(グループ化キーを構成するイベント属性の値によって定義されているように) 共通の特性を共有するイベントの各グループについて、別個のルール・インスタンス (または自身のコピー) を作成するよう指示することです。

詳細

以下の 2 つのシナリオで、グループ化キーの重要性について説明します。

シナリオ 1:
2 つのイベント、DB2down イベントおよび DB2up イベントが発生します。DB2 プログラムは、A、B、および C という 3 つのコンピューター上で実行されています。DB2down イベントを DB2up イベントと相関させ、DB2 プログラムが停止し、再始動しないときにオペレーターにアラートを出すために、シーケンス・ルールが定義されています。

このシーケンス・ルールがグループ化キーなしで定義されており、コンピューター A から DB2down イベントが受信された場合、いずれかのコンピューターから DB2up イベントが受信されるとシーケンスは完了しますが、これにより意図した結果は得られません。しかし、グループ化キーが hostname 属性として定義されていた場合、ルールの固有のコピー (またはインスタンス) が、選択されたイベントの hostname 属性の各固有値に対して作成されます。Active Correlation Technology エンジンは、各イベントを、正しいルール・インスタンス (そのイベントの hostname 値に対するルール・インスタンス) に送信します。このため、コンピューター A から DB2down イベントを受信すると、Active Correlation Technology エンジンは、コンピューター A のルール・インスタンスを作成します。コンピューター B から DB2down イベントを受信すると、Active Correlation Technology エンジンは、コンピューター B に対して 2 番目のルール・インスタンスを作成します。コンピューター B から DB2up イベントを受信すると、Active Correlation Technology エンジンは、そのイベントを 2 番目のルール・インスタンス内で処理します。シーケンスは完了し、コンピューター B からの DB2down イベントおよび DB2up イベントが正しく相関されているため、オペレーターにアラートが出されます。

シナリオ 2:
「`Incorrect login attempted`」メッセージに対するイベントが、特定の環境内のすべてのコンピューター上で発生します。イベントには、ユーザー ID が含まれています。このイベントが 5 分間で 10 回より多く発生した場合にオペレーターに警告が発行されるよう、しきい値ルールが定義されています。

グループ化キーをユーザー ID として定義することもできます。その場合、各固有ユーザー ID に対して新規のルール・インスタンスが作成されます。各ユーザーのログイン試行は、固有のしきい値ルール・インスタンス内で追跡され、各インスタンスでは、そのユーザーによるログイン試行回数を個別にカウントします。 5 分間で不正ログイン試行が 10 回を超えるユーザー ID があれば、オペレーターは警告を受け取ります。

他にも、以下に示すようなさまざまな方法が考えられます。

- グループ化キーを、ユーザー ID ではなくホスト名として定義することもできます。このオプションでは、単一のコンピューターに対する多数の不正ログイン試行を検出できます。

- グループ化キーを、ホスト名とユーザー ID の組み合わせとして定義することもできます。このオプションでは、特定のユーザー ID による特定のコンピューターに対するハッキング試行の可能性を検出できます。

ルールに対して指定されたすべてのイベント・タイプで同じ属性が指定されている場合、グループ化キーを定義するための最も単純かつ一般的な方法として、<attributeName> エレメントを使用します。

属性

<groupingKey> には、以下の属性があります。

表 42. <groupingKey> エレメントの属性

名前	説明	データ・タイプ	必須?
missingAttributeHandling	以下の条件のいずれかに該当する場合に、ルールが実行する必要のあるアクションを定義します。 • 選択されたイベントに、グループ化キーを構成する属性が指定されているものの、その属性の値が欠落している場合。 • <computedValue> エレメント内の式が、ヌル値を戻す場合。ルールは、このヌル値を欠落した属性値として処理します。 missingAttributeHandling 属性の有効値は以下のとおりです。 • `ignoreEvent` (デフォルト値)。ルールがイベントを無視し、何のアクションも実行しないことを意味します。 • `ignoreAttribute`。ルールはイベントを受け入れるものの、値が欠落している属性を無視することを意味します。その後 Active Correlation Technology エンジンは、その属性に代替値を組み込みます。	xsd:string	いいえ

このエレメントを含むエレメント

<groupingKey> は、以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<groupingKey> には、以下のエレメントが含まれます。

表 43. <groupingKey> エレメントに含まれるエレメント

エレメント	必須/オプション
<attributeAlias>	これらのエレメントのうち 1 つは指定する必要があります。これらのエレメントを複数コーディングするかどうかはオプションです。 3 つすべてのエレメントを複数回指定することができます。これらのエレメントは、任意の順序でコーディングできます。
<attributeName>	
<computedValue>	

import エレメント

<import> エレメントには、ルール内の他の式で使用するためにインポートする外部モジュール (Java クラスなど) を指定する式が含まれます。

詳細

式のコードは、<import> エレメント内のストリングです。Active Correlation Technology コンパイラーは、<import> エレメントによって提供される import ステートメントを使用して、外部メソッドを呼び出すルール内の式のコードをコンパイルします。

属性

<import> には、以下の属性があります。

表 44. <import> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<import> は、以下のエレメントに含まれます。

- <ruleSet>
- <ruleBlock>

このエレメントに含まれるエレメント

<import> に含まれるエレメントはありません。

関連概念

22 ページの『外部モジュールおよびオブジェクトのインポートとアクセス』
この例は、式から外部コード (Java クラスなど) および外部オブジェクトにアク

セス可能にする方法を示します。外部オブジェクトとは、アプリケーションが式と通信するために作成するオブジェクトのことです。

inactiveWhenLoaded エレメント

<inactiveWhenLoaded> エレメントは、Active Correlation Technology エンジンによるロード時に、ルールが非アクティブであることを指定します。ルールは、別の手段によってアクティブ化されるまで非アクティブのままになります。

属性

<inactiveWhenLoaded> には、属性がありません。

このエレメントを含むエレメント

<inactiveWhenLoaded> は、以下のエレメントに含まれます。

- <start>

このエレメントに含まれるエレメント

<inactiveWhenLoaded> に含まれるエレメントはありません。

lifeCycleActions エレメント

<lifeCycleActions> エレメントには、ルールのライフ・サイクルの 4 つの基本ステージで実行するアクションを定義するエレメントが含まれます。

詳細

ロード・ステージおよびアクティブ化ステージについて定義されるアクションは、ルールが実際にロードまたはアクティブ化された後、ルールが処理を開始する前に呼び出されます。非アクティブ化ステージおよびアンロード・ステージについて定義されるアクションは、ルールが実際に非アクティブ化またはアンロードされる直前に呼び出されます。

属性

<lifeCycleActions> には、属性がありません。

このエレメントを含むエレメント

<lifeCycleActions> は、以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<lifeCycleActions> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 45. <lifeCycleActions> エレメントに含まれるエレメント

エレメント	必須/オプション
<onLoad>	オプション。指定できるのは 0 回または 1 回です。
<onActivation>	オプション。指定できるのは 0 回または 1 回です。
<onDeactivation>	オプション。指定できるのは 0 回または 1 回です。
<onUnload>	オプション。指定できるのは 0 回または 1 回です。

never エレメント

<never> エレメントは、ルールが特定の時点で非アクティブ化されることがないことを指定します。ただし、ルールはイベントまたはその他の手段で非アクティブ化することはできます。

属性

<never> には、属性がありません。

このエレメントを含むエレメント

<never> は、以下のエレメントに含まれます。

- <stop>

このエレメントに含まれるエレメント

<never> に含まれるエレメントはありません。

onActivation エレメント

<onActivation> エレメントは、ルールがアクティブ化されたときに実行するアクションまたはアクションのセットを指定します。<onActivation> アクションは、ルールがアクティブ化された後、ルールが処理を開始する前に呼び出されます。

詳細

ルール・セットに、同じ日時に、または同じイベントによってアクティブ化され、同じ時間枠を持つ複数のルールが含まれる場合、これらのルールの以下のアクションは正確に同時には実行されません。

- <onTimeOut> エレメントおよび <onTimeWindowComplete> エレメント内のルール応答アクション
- <onActivation> エレメントおよび <onDeactivation> エレメント内のライフ・サイクル・アクション

これらのアクションは、任意の順序で連続して実行されます。これらは、必ずしもルール・セット内でコーディングされている順序で実行されるわけではありません。各アクションは、シーケンス内の次のアクションが開始される前に完了する必要があるため、アクションは同時には実行されません。

属性

<onActivation> には、属性がありません。

このエレメントを含むエレメント

<onActivation> は、以下のエレメントに含まれます。

- <lifeCycleActions>

このエレメントに含まれるエレメント

<onActivation> には、以下のエレメントが含まれます。

表 46. <onActivation> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onDeactivation エレメント

<onDeactivation> エレメントは、ルールが非アクティブ化されるときに実行するアクションまたはアクションのセットを指定します。<onDeactivation> アクションは、ルールが非アクティブ化される直前に呼び出されます。

詳細

ルール・セットに、同じ日時に、または同じイベントによってアクティブ化され、同じ時間枠を持つ複数のルールが含まれる場合、これらのルールの以下のアクションは正確に同時には実行されません。

- <onTimeOut> エレメントおよび <onTimeWindowComplete> エレメント内のルール応答アクション
- <onActivation> エレメントおよび <onDeactivation> エレメント内のライフ・サイクル・アクション

これらのアクションは、任意の順序で連続して実行されます。これらは、必ずしもルール・セット内でコーディングされている順序で実行されるわけではありません。各アクションは、シーケンス内の次のアクションが開始される前に完了する必要があるため、アクションは同時には実行されません。

属性

<onDeactivation> には、属性がありません。

このエレメントを含むエレメント

<onDeactivation> は、以下のエレメントに含まれます。

- <lifeCycleActions>

このエレメントに含まれるエレメント

<onDeactivation> には、以下のエレメントが含まれます。

表 47. <onDeactivation> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onDetection エレメント

<onDetection> エレメントは、重複ルール、フィルター・ルール、シーケンス・ルール、およびしきい値ルールに対してのみ有効です。これは、ルール・パターンが検出されたときに実行するアクションまたはアクションのセットを指定します。

詳細

表 48 は、<onDetection> アクションが有効な各ルール・タイプについて、ルール・パターンがどのように検出されるかを説明しています。

表 48. ルール・タイプに基づくルール・パターンの検出方法

ルール・タイプ	ルール・パターンの検出方法
重複	このルール・パターンは、イベント選択基準を満たす最初のイベントの受信時に検出されます。
フィルター	このルール・パターンは、イベント選択基準を満たすすべてのイベントの受信時に検出されます。
シーケンス	このルール・パターンは、イベント選択基準を満たすイベントのシーケンスが、時間枠内に適切な順序で受信されたときに検出されます。
しきい値	このルール・パターンは、イベント選択基準を満たすイベントが時間枠内に受信され、しきい値に到達したときに検出されます。

属性

<onDetection> には、属性がありません。

このエレメントを含むエレメント

<onDetection> は、以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<onDetection> には、以下のエレメントが含まれます。

表 49. <onDetection> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onLoad エレメント

<onLoad> エレメントは、実行中の Active Correlation Technology エンジンにルールがロード (またはデプロイ) されたときに実行するアクションまたはアクションのセットを指定します。<onLoad> アクションは、ルールがロードされた後、ルールが処理を開始する前に呼び出されます。

属性

<onLoad> には、属性がありません。

このエレメントを含むエレメント

<onLoad> は、以下のエレメントに含まれます。

- <lifeCycleActions>

このエレメントに含まれるエレメント

<onLoad> には、以下のエレメントが含まれます。

表 50. <onLoad> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onNextEvent エレメント

<onNextEvent> エレメントは、重複ルールについてのみ有効です。これは、重複ルールが、指定された時間枠内でイベント選択基準を満たす 2 番目以降の各イベントを受信したときに実行するアクションまたはアクションのセットを指定します。

詳細

重複ルールの場合、Active Correlation Technology エンジンは、指定された時間枠内でイベント選択基準に一致する 2 番目以降の各イベントについて、ルール・セット処理をスキップします。このため、<onNextEvent> アクションをコーディングする唯一の理由は、 2 番目以降の各イベントについての代替処理を指定することにあります。

属性

<onNextEvent> には、属性がありません。

このエレメントを含むエレメント

<onNextEvent> は、以下のエレメントに含まれます。

- <duplicateRule>

このエレメントに含まれるエレメント

<onNextEvent> には、以下のエレメントが含まれます。

表 51. <onNextEvent> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onTimeOut エレメント

<onTimeOut> エレメントは、シーケンス・ルールおよびしきい値ルールについてのみ有効です。これは、ルールの時間枠の期限が切れた場合に実行するアクションまたはアクションのセットを指定します。

詳細

表 52 は、<onTimeOut> アクションが有効な各ルール・タイプについて、時間枠がどのように満了するかを説明しています。

表 52. ルール・タイプに基づく時間枠の満了方法

ルール・タイプ	時間枠の満了方法
シーケンス	1 つ以上のイベントが受け入れられたものの、時間枠内でイベントのすべてのシーケンスが受信されない場合に、時間枠が満了します。
しきい値	1 つ以上のイベントが受け入れられたものの、時間枠内でしきい値に到達しなかった場合に、時間枠が満了します。

ルール・セットに、同じ日時に、または同じイベントによってアクティブ化され、同じ時間枠を持つ複数のルールが含まれる場合、これらのルールの以下のアクションは正確に同時には実行されません。

- <onTimeOut> エレメントおよび <onTimeWindowComplete> エレメント内のルール応答アクション
- <onActivation> エレメントおよび <onDeactivation> エレメント内のライフ・サイクル・アクション

これらのアクションは、任意の順序で連続して実行されます。これらは、必ずしもルール・セット内でコーディングされている順序で実行されるわけではありません。各アクションは、シーケンス内の次のアクションが開始される前に完了する必要があるため、アクションは同時には実行されません。

属性

<onTimeOut> には、属性がありません。

このエレメントを含むエレメント

<onTimeOut> は、以下のエレメントに含まれます。

- <sequenceRule>
- <thresholdRule>

このエレメントに含まれるエレメント

<onTimeOut> には、以下のエレメントが含まれます。

表 53. <onTimeOut> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onTimeWindowComplete エレメント

<onTimeWindowComplete> エレメントは、コレクション、計算、重複、およびタイマーの各ルールについてのみ有効です。これは、ルールの時間枠が終了したときに実行するアクションまたはアクションのセットを指定します。

詳細

ルール・セットに、同じ日時に、または同じイベントによってアクティブ化され、同じ時間枠を持つ複数のルールが含まれる場合、これらのルールの以下のアクションは正確に同時には実行されません。

- <onTimeOut> エレメントおよび <onTimeWindowComplete> エレメント内のルール応答アクション
- <onActivation> エレメントおよび <onDeactivation> エレメント内のライフ・サイクル・アクション

これらのアクションは、任意の順序で連続して実行されます。これらは、必ずしもルール・セット内でコーディングされている順序で実行されるわけではありません。各アクションは、シーケンス内の次のアクションが開始される前に完了する必要があるため、アクションは同時には実行されません。

属性

<onTimeWindowComplete> には、属性がありません。

このエレメントを含むエレメント

<onTimeWindowComplete> は、以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<onTimeWindowComplete> には、以下のエレメントが含まれます。

表 54. <onTimeWindowComplete> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

onUnload エレメント

<onUnload> エレメントは、実行中の Active Correlation Technology エンジンからルールがアンロード (または除去) されるときに実行するアクションまたはアクションのセットを指定します。<onUnload> アクションは、ルールがアンロードされる直前に呼び出されます。

属性

<onUnload> には、属性がありません。

このエレメントを含むエレメント

<onUnload> は、以下のエレメントに含まれます。

- <lifeCycleActions>

このエレメントに含まれるエレメント

<onUnload> には、以下のエレメントが含まれます。

表 55. <onUnload> エレメントに含まれるエレメント

エレメント	必須/オプション
<action>	オプション。0 回以上指定できます。

ruleBlock エレメント

<ruleBlock> エレメントは、関連するルールをグループ化し、ルールを階層に編成する方法を提供します。

属性

<ruleBlock> には、以下の属性があります。

表 56. <ruleBlock> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルール・ブロックを識別します。この ID は、ルール・セットまたはこのルール・ブロックを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	Yes

このエレメントを含むエレメント

<ruleBlock> は、以下のエレメントに含まれます。

- <ruleSet>
- <ruleBlock>

このエレメントに含まれるエレメント

<ruleBlock> には、以下のエレメントが含まれます。

<comment> エレメント、<import> エレメント、および <variable> エレメントをコーディングする場合、これらはここに示した順序でコーディングする必要があります。その他のエレメントは、任意の順序でコーディングできます。

表 57. <ruleBlock> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<import>	オプション。0 回以上指定できます。
<variable>	オプション。0 回以上指定できます。
<ruleBlock>	オプション。0 回以上指定できます。
<collectionRule>	オプション。0 回以上指定できます。
<computationRule>	オプション。0 回以上指定できます。
<duplicateRule>	オプション。0 回以上指定できます。
<filterRule>	オプション。0 回以上指定できます。
<sequenceRule>	オプション。0 回以上指定できます。
<thresholdRule>	オプション。0 回以上指定できます。
<timerRule>	オプション。0 回以上指定できます。

ruleSet エレメント

act:ruleSet によって定義される <ruleSet> エレメントは、 Active Correlation Technology ルール言語のルート・エレメントです。他のすべてのエレメントは、この <ruleSet> エレメントに含まれます。

詳細

Active Correlation Technology 言語スキーマ (act:ruleSet) および Active Correlation Technology 基本ルール・セット・スキーマ (br:ruleSet) によって定義される <ruleSet> エレメントは重複しています。ただし、ルール・セットの作成時には、<ruleSet> エレメントでネーム・スペース act:ruleSet を指定する必要があります。

属性

<ruleSet> には、以下の属性があります。

表 58. <ruleSet> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルール・セットを識別します。この ID は固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<ruleSet> はルール言語のルート・エレメントであるため、どのエレメントにも含まれません。

このエレメントに含まれるエレメント

<ruleSet> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 59. <ruleSet> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<import>	オプション。0 回以上指定できます。
<variable>	オプション。0 回以上指定できます。
<ruleBlock>	オプション。0 回以上指定できます。

runUntilDeactivated エレメント

<runUntilDeactivated> エレメントは、ルールが非アクティブ化されるまで時間枠が継続されることを指定します。このため、このルールの時間枠は、ルールが処理を開始したときに開始され、ルールが非アクティブ化されるかルール・セットから除去されるまで、あるいは Active Correlation Technology エンジンがシャットダウンするまで終了しません。

詳細

<runUntilDeactivated> エレメントを含むルールの固有の振る舞いは、ルール・タイプによって異なります。 110 ページの表 60 では、 <timeWindow> エレメントが有効で、<runUntilDeactivated> エレメントが含まれる場合の各ルール・タイプにおけるルールの振る舞いを説明しています。

表 60. *<runUntilDeactivated> がコーディングされている場合のルールの振る舞い*

ルール・タイプ	**<runUntilDeactivated> がコーディングされている場合のルールの振る舞い**
コレクション	コレクション・ルールは、そのイベント選択基準を満たす最初のイベントを受け入れ、ルールが非アクティブ化されるまでイベントの受け入れおよび処理を継続します。ルールが非アクティブ化されると、<onTimeWindowComplete> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。
計算	計算ルールは、そのイベント選択基準を満たす最初のイベントを受け入れ、ルールが非アクティブ化されるまでイベントの受け入れおよび処理を継続します。ルールが非アクティブ化されると、<onTimeWindowComplete> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。
重複	重複ルールは、そのイベント選択基準を満たす最初のイベントを受け入れ、ルールが非アクティブ化されるまでイベントの受け入れおよび処理を継続します。ルールが非アクティブ化されると、<onTimeWindowComplete> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。
シーケンス	シーケンス・ルールは、そのイベント選択基準を満たす最初のイベントを受け入れ、以下のいずれかの状況が発生するまでイベントの受け入れおよび処理を継続します。 • シーケンス・パターンが検出された。これが発生すると、<onDetection> エレメントで定義されているアクションが実行され、ルールはその初期状態に戻ります。このルールによるイベント処理が再度開始され、このプロセスをルールが非アクティブ化されるまで何度も繰り返すことができます。 • イベントの処理中にルールが非アクティブ化された。これが発生すると、 <onTimeOut> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。
しきい値	しきい値ルールは、そのイベント選択基準を満たす最初のイベントを受け入れ、以下のいずれかの状況が発生するまでイベントの受け入れおよび処理を継続します。 • しきい値パターンが検出された。これが発生すると、<onDetection> エレメントで定義されているアクションが実行され、ルールはその初期状態に戻ります。このルールによるイベント処理が再度開始され、このプロセスをルールが非アクティブ化されるまで何度も繰り返すことができます。 • イベントの処理中にルールが非アクティブ化された。これが発生すると、 <onTimeOut> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。

表 60. <runUntilDeactivated> がコーディングされている場合のルールの振る舞い (続き)

ルール・タイプ	<runUntilDeactivated> がコーディングされている場合のルールの振る舞い
タイマー	タイマー・ルールは、アクティブ化された後、非アクティブ化されるまで何も実行しません。非アクティブ化されると、<onTimeWindowComplete> エレメントで定義されているアクションが実行され、その直後に <onDeactivation> エレメントで定義されているアクションが実行されます。<timerRule> エレメントの repeat 属性は無視されます。

属性

<runUntilDeactivated> には、属性がありません。

このエレメントを含むエレメント

<runUntilDeactivated> は、以下のエレメントに含まれます。

- <timeWindow>

このエレメントに含まれるエレメント

<runUntilDeactivated> に含まれるエレメントはありません。

sequenceRule エレメント

<sequenceRule> エレメントは、シーケンス・パターンに従ってルールを定義します。シーケンス・ルールは、複数のイベント・セレクターを使用可能な唯一のルールです。また、このルールには、少なくとも 2 つのイベント・セレクターが必要です。

属性

<sequenceRule> には、以下の属性があります。

表 61. <sequenceRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は `false` で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

表 61. <sequenceRule> エレメントの属性 (続き)

名前	説明	データ・タイプ	必須?
到着順序 (arrivalOrder)	ルールに対して <eventSelector> エレメントがコーディングされている順序でイベントが到着する必要があるかどうかを定義します。有効値は以下のとおりです。 • inOrder (デフォルト値) • randomOrder	xsd:string	いいえ

到着順序 (arrivalOrder) 属性の値が randomOrder の場合、 <eventSelector> エレメントのコーディング順序が重要です。最も具体的なイベント選択基準を指定する <eventSelector> エレメントを、それよりも具体的でないイベント選択基準を指定する <eventSelector> エレメントに先行してコーディングする必要があります。そのようにしないと、検出されるべきシーケンスが検出されなくなります。

例えば、以下のような場合を想定します。

- 3 つの <eventSelector> エレメントが定義されています。
- 最初の <eventSelector> エレメントは、イベント eventA をチェックしています。
- 2 番目の <eventSelector> エレメントは、すべてのイベントをチェックしています。
- 3 番目の <eventSelector> エレメントは、イベント eventB をチェックしています。
- 指定された時間枠内に、システムに対してイベント eventA、eventB、および eventC が発行されます。

ルールの振る舞いは以下のようになり、検出されるべきシーケンスが検出されなくなります。

1. 最初のイベント eventA は、最初の <eventSelector> エレメントによって受け入れられます。
2. 2 番目のイベント eventB は、2 番目の <eventSelector> エレメントによって受け入れられます。
3. 3 番目のイベント eventC は無視されます。

<eventSelector> エレメントが正しくコーディングされ、最も具体的なイベント選択基準がそれよりも具体的でないイベント選択基準よりも前にある場合を想定します。

- 最初の <eventSelector> エレメントは、イベント eventA をチェックしています。
- 2 番目の <eventSelector> エレメントは、イベント eventB をチェックしています。
- 3 番目の <eventSelector> エレメントは、すべてのイベントをチェックしています。

ルールの振る舞いは以下のようになり、シーケンスが検出されます。

1. 最初のイベント eventA は、最初の <eventSelector> エレメントによって受け入れられます。

2. 2 番目のイベント eventB は、2 番目の <eventSelector> エレメントによって受け入れられます。
3. 3 番目のイベント eventC は、3 番目の <eventSelector> エレメントによって受け入れられます。

このエレメントを含むエレメント

<sequenceRule> は、以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<sequenceRule> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 62. <sequenceRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	シーケンス・ルールの場合、このエレメントを 2 回指定する必要があります。追加の指定も可能です。
<groupingKey>	オプション。指定できるのは 0 回または 1 回です。
<timeWindow>	必須。指定できるのは 1 回のみです。
<onDetection>	オプション。指定できるのは 0 回または 1 回です。
<onTimeOut>	オプション。指定できるのは 0 回または 1 回です。

関連概念

14 ページの『シーケンス・パターン』
シーケンス・ルールは、シーケンス・パターンによって定義されます。これは、特定の時間間隔内にイベントの特定のシーケンスが到着するかどうかを検出します。シーケンスは、順序付けられていても、ランダムであってもかまいません。シーケンス・ルールはステートフル・ルールです。

start エレメント

<start> エレメントは、ルールが、特定の日付の特定の時刻にアクティブ化されるか、あるいは Active Correlation Technology エンジンによってルールがロードされたときにアクティブ化されるかを定義します。

詳細

<start> エレメントがまったくコーディングされていない場合、デフォルトの開始時刻は <whenLoaded> エレメントで定義されている時刻と同じです。

属性

<start> には、属性がありません。

このエレメントを含むエレメント

<start> は、以下のエレメントに含まれます。

- <activationTime>

このエレメントに含まれるエレメント

<start> には、以下のエレメントが含まれます。

表 63. <start> エレメントに含まれるエレメント

エレメント	必須/オプション
<dateTime>	これらのエレメントのうち 1 つを指定する必要があります。選択したエレメントは 1 回のみ指定できます。
<whenLoaded>	
<inactiveWhenLoaded>	

stop エレメント

<stop> エレメントは、ルールが、特定の日付の特定の時刻または特定の継続時間後に非アクティブ化されること、もしくは特定の時刻には非アクティブ化されないことを定義します。

詳細

<stop> エレメントがまったくコーディングされていない場合、デフォルトの停止時刻は <never> エレメントで定義されている時刻と同じです。

属性

<stop> エレメントには、属性がありません。

このエレメントを含むエレメント

<stop> エレメントは、以下のエレメントに含まれます。

- <activationTime>

このエレメントに含まれるエレメント

<stop> エレメントには、以下のエレメントが含まれます。

表 64. <stop> エレメントに含まれるエレメント

エレメント	必須/オプション
<dateTime>	これらのエレメントのうち 1 つを指定する必要があります。選択したエレメントは 1 回のみ指定できます。
<never>	
<after>	

stopAfter エレメント

<stopAfter> エレメントは、ルール・インスタンス (<groupingKey> エレメントで定義) がアクティブ化された後アクティブなままとなる継続時間を指定します。この継続時間が経過すると、ルール・インスタンスは非アクティブ化されます。

属性

<stopAfter> エレメントには、以下の属性があります。

表 65. <stopAfter> エレメントの属性

名前	説明	データ・タイプ	必須?
duration	継続時間の時間量を指定します。この属性のデータ・タイプは、unit 属性の値によって異なります。	• unit 属性の値が ISO-8601 の場合、データ・タイプは xsd:duration です。 • unit 属性の値が milliseconds の場合、データ・タイプは xsd:positiveInteger です。	はい
unit	使用する時間単位を指定します。この属性の有効値は以下のとおりです。 • ISO-8601 • milliseconds	xsd:string	はい

継続時間における ISO 8601 規格の使用

unit 属性の値として ISO-8601 がコーディングされている場合、duration 属性の値が ISO 8601 規格に従ってコーディングされ、継続時間を 1 つのストリングとして指定することを示します。標準の XML スキーマのデータ・タイプ仕様では、ISO 8601 を使用してデータ・タイプ duration を提供します。このデータ・タイプについては、http://www.w3.org/TR/xmlschema-2/#duration で詳しく説明されています。

標準の XML スキーマの duration データ・タイプのフォーマットは、以下のストリングです。

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

- ストリングは常に文字 P で始まります。
- *n*Y は、年数を示します。 1 年は、365 日と同じです。したがって、1Y とコーディングするのは、365D とコーディングするのと同じです。
- *n*M は月数を示します。 1 カ月は 30 日と同じです。したがって、1M とコーディングするのは、30D とコーディングするのと同じです。
- *n*D は日数を示します。
- T は時間の単位 (時間、分、および秒) と日の単位 (年、月、日) を分離する分離文字です。時間の単位は必ず T の後に続きます。
- *n*H は時間数を示します。
- *n*M は分数を示します。

- *n*S は秒数を示します。

以下はフォーマットの例です。

- P5DT12H は 5.5 日です。
- PT59M59S は 59 分 59 秒です。
- P1M は 1 カ月です。

このエレメントを含むエレメント

<stopAfter> は、 <activateOnEvent> エレメントに含まれますが、これは <activateOnEvent> が <activationByGroupingKey> エレメント内でコーディングされている場合のみです。

このエレメントに含まれるエレメント

<stopAfter> に含まれるエレメントはありません。

thresholdRule エレメント

<thresholdRule> エレメントは、しきい値パターンに従ってルールを定義します。

属性

<thresholdRule> には、以下の属性があります。

表 66. <thresholdRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	ルールが、すべてのイベントを受信するか、他のルールから転送されたイベントのみを受信するか決定します。デフォルト値は false で、ルールが他のルールから転送されたイベントを含むすべてのイベントを受信することを示します。	xsd:boolean	いいえ

このエレメントを含むエレメント

<thresholdRule> は、以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<thresholdRule> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 67. <thresholdRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<eventSelector>	オプション。指定できるのは 0 回または 1 回です。
<groupingKey>	オプション。指定できるのは 0 回または 1 回です。
<booleanThreshold>	これらのエレメントのうち 1 つを指定する必要があります。選択したエレメントは 1 回のみ指定できます。
<computedThreshold>	
<eventCountThreshold>	
<timeWindow>	必須。指定できるのは 1 回のみです。
<onDetection>	オプション。指定できるのは 0 回または 1 回です。
<onTimeOut>	オプション。指定できるのは 0 回または 1 回です。

関連概念

17 ページの『しきい値パターン』
しきい値ルールは、しきい値パターンによって定義されます。これは、 特定の時間間隔内に選択されたイベントのグループを収集し、各イベントが受信された後、 しきい値条件が満たされたかどうかを判別します。これはステートフル・ルール です。

timeInterval エレメント

<timeInterval> エレメントは、時間枠の継続時間を指定します。

属性

<timeInterval> には、以下の属性があります。

表 68. <timeInterval> エレメントの属性

名前	説明	データ・タイプ	必須?
duration	継続時間の時間量を指定します。この属性のデータ・タイプは、unit 属性の値によって異なります。	• unit 属性の値が ISO-8601 の場合、データ・タイプは xsd:duration です。 • unit 属性の値が milliseconds の場合、データ・タイプは xsd:positiveInteger です。	はい

表 68. <timeInterval> エレメントの属性 (続き)

名前	説明	データ・タイプ	必須?
unit	使用する時間単位を指定します。この属性の有効値は以下のとおりです。 • ISO-8601 • milliseconds	xsd:string	はい

継続時間における ISO 8601 規格の使用

unit 属性の値として ISO-8601 がコーディングされている場合、duration 属性の値が ISO 8601 規格に従ってコーディングされ、継続時間を 1 つのストリングとして指定することを示します。標準の XML スキーマのデータ・タイプ仕様では、ISO 8601 を使用してデータ・タイプ duration を提供します。このデータ・タイプについては、http://www.w3.org/TR/xmlschema-2/#duration で詳しく説明されています。

標準の XML スキーマの duration データ・タイプのフォーマットは、以下のストリングです。

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

- ストリングは常に文字 P で始まります。
- *n*Y は、年数を示します。 1 年は、365 日と同じです。したがって、1Y とコーディングするのは、365D とコーディングするのと同じです。
- *n*M は月数を示します。 1 カ月は 30 日と同じです。したがって、1M とコーディングするのは、30D とコーディングするのと同じです。
- *n*D は日数を示します。
- T は時間の単位 (時間、分、および秒) と日の単位 (年、月、日) を分離する分離文字です。時間の単位は必ず T の後に続きます。
- *n*H は時間数を示します。
- *n*M は分数を示します。
- *n*S は秒数を示します。

以下はフォーマットの例です。

- P5DT12H は 5.5 日です。
- PT59M59S は 59 分 59 秒です。
- P1M は 1 カ月です。

このエレメントを含むエレメント

<timeInterval> は、以下のエレメントに含まれます。

- <timeWindow>

このエレメントに含まれるエレメント

<timeInterval> に含まれるエレメントはありません。

timerRule エレメント

<timerRule> エレメントは、タイマー・パターンに従ってルールを定義します。

属性

<timerRule> には、以下の属性があります。

表 69. <timerRule> エレメントの属性

名前	説明	データ・タイプ	必須?
name	ルールを識別します。この ID は、このルールを含むルール・ブロック内で固有でなければなりません。ピリオドを含めることはできません。	xsd:NMTOKEN	はい
processOnlyForwardedEvents	タイマー・ルールはイベントを処理しないため、この属性は無視されます。	xsd:boolean	いいえ
repeat	タイマー・ルールが、非アクティブ化されるまで繰り返し実行されるかどうかを定義します。有効値は以下のとおりです。 • `true` (デフォルト値) • `false` 値が `false` に設定されている場合、ルールはその時間間隔をとおして 1 回のみ実行され、それぞれの時間枠の完了時にルール応答アクションを実行し、停止します。 タイマー・ルールの <timeWindow> エレメントに <runUntilDeactivated> エレメントが含まれる場合、repeat 属性は無視されます。	xsd:boolean	いいえ

このエレメントを含むエレメント

<timerRule> は、以下のエレメントに含まれます。

- <ruleBlock>

このエレメントに含まれるエレメント

<timerRule> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 70. <timerRule> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<variable>	オプション。0 回以上指定できます。
<activationInterval>	オプション。指定できるのは 0 回または 1 回です。
<lifeCycleActions>	オプション。指定できるのは 0 回または 1 回です。
<timeWindow>	必須。指定できるのは 1 回のみです。
<onTimeWindowComplete>	オプション。指定できるのは 0 回または 1 回です。

関連概念

19 ページの『タイマー・パターン』
タイマー・ルールは、タイマー・パターンによって定義されます。これにより、一定の間隔で アクションが開始されます。これはステートフル・ルールです。タイマー・ルールはイベントを 処理しませんが、イベントによってアクティブ化または非アクティブ化できます。

timeWindow エレメント

<timeWindow> エレメントには、ルールが処理を行う時間間隔を定義するエレメントが含まれます。

詳細

例えば、重複ルールの時間枠は、受け入れられた最初のイベントの重複であるイベントについて、ルールがどのくらいの期間チェックすべきかを定義します。時間枠が 30 秒の場合、重複ルールは、最初のイベントを受け入れてから 30 秒以内に発生するすべての重複イベントを処理します。

属性

<timeWindow> には、属性がありません。

このエレメントを含むエレメント

<timeWindow> は、以下のエレメントに含まれます。

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

このエレメントに含まれるエレメント

<timeWindow> には、以下のエレメントが含まれます。

表 71. <timeWindow> エレメントに含まれるエレメント

エレメント	必須/オプション
<timeInterval>	これらのエレメントのうち 1 つを指定する必要があります。選択したエレメントは 1 回のみ指定できます。
<runUntilDeactivated>	

variable エレメント

<variable> エレメントは、変数を定義し、式によって参照可能な形式で情報を保持します。変数は、ルール・セット、ルール・ブロック、またはルールの各レベルで定義できます。

詳細

ルール・セット変数
ルール・セット全体に適用され、そのルール・セット内の式による参照が可能です。

ルール・ブロック変数
ルール・ブロック内 (およびそのルール・ブロックに含まれるすべてのルール・ブロック内) にのみ適用され、そのルール・ブロック内の任意の式による参照が可能です。

ルール変数
そのルール内の式にのみ適用されます。

変数には、ルール階層の異なるレベルで同じ名前を指定することができます。変数がアクセスされると、その変数の最もローカルな定義が使用されます。例えば、ルール・セット・レベル、ルール・ブロック・レベル、およびルール・レベルで同じ名前を持つ変数が定義されている場合、ルール・レベルの変数定義がそのルール内の式によって使用されます。

変数がルール・セットまたはルール・ブロック・レベルで定義されている場合、複数のルールが別々の時点でこれらの変数を取得し、設定します。このため、正しい変数値が確実に保持されるようにするには、ルール・セット内で変数間の相互作用をどのようにコーディングするかについて認識しておく必要があります。

変数がルール・セット・レベルまたはルール・ブロック・レベルで定義されている場合、ルール・パターンに一致した後は再初期化されません。

以下のいずれかの条件下においては、不正な変数値の設定を回避するために、ルール・セット変数およびルール・ブロック変数の取得および設定時にロックを使用します。

- <onTimeOut> アクション中にタイマー・ルールが変数を取得または設定した場合
- Active Correlation Technology エンジンが組み込まれたアプリケーションがマルチスレッド化された場合

ルールがグループ化キーを使用して定義されている場合、 <variable> エレメントによって定義されたルール変数は、ライフ・サイクル・アクション内、または <activateOnEvent> エレメントに含まれる <filteringPredicate> エレメント内、あるいは <activationInterval> エレメントに含まれる <deactivateOnEvent> エレメントで無効です。これは、この場合ルール変数がルール・インスタンスにのみ適用され、これらの式の実行時にルール・インスタンスが存在しないためです。

属性

<variable> には、以下の属性があります。

表 72. <variable> エレメントの属性

名前	説明	データ・タイプ	必須?
name	特定の変数を識別します。変数はその名前によって参照されます。	xsd:NMTOKEN	はい
dataType	変数に含まれる情報のタイプを識別します。これは、java.lang.String のように、完全修飾されたデータ・タイプでなければなりません。	xsd:NMTOKEN	はい

変数の命名上の制約

変数名には特定の制約事項があります。このため、<variable> エレメントの name 属性の値には、以下の制約事項があります。

- 変数には、以下の文字のみを使用できます。
  - 大文字の ASCII ラテン文字 A から Z。Unicode 表記は、\u0041 から \u005a です。
  - 小文字の ASCII ラテン文字 a から z。Unicode 表記は、\u0061 から \u007a です。
  - ASCII のアンダースコアー (_)。Unicode 表記は、\u005f です。
  - ドル記号 ($)。Unicode 表記は、\u0024 です。
  - ASCII の数字 0 から 9。Unicode 表記は \u0030 から \u0039 です。
- NULL は使用できません。
- 空ストリングにすることはできません。
- スペースは使用できません。
- ピリオドは使用できません。
- 形式を問わず (大文字、小文字、大/小文字混合のいずれの形式であっても) act_ で開始することはできません。

このエレメントを含むエレメント

<variable> は、以下のエレメントに含まれます。

- <ruleSet>
- <ruleBlock>
- <collectionRule>
- <computationRule>

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

このエレメントに含まれるエレメント

<variable> には、以下のエレメントが含まれます。

以下のエレメントは、表示されている順序でコーディングする必要があります。エレメントがオプションである場合、コーディングする必要はありませんが、コーディングするエレメントはすべて正しい順序になっている必要があります。

表 73. <variable> エレメントに含まれるエレメント

エレメント	必須/オプション
<comment>	オプション。指定できるのは 0 回または 1 回です。
<varInitializer>	必須。指定できるのは 1 回のみです。

関連概念

26 ページの『変数』
ルール言語では、特定の変数を使用して、さまざまなイベントの発生や ルール間でイベント関連の情報を保管します。そして、このイベント関連の 情報に、ルール内の式からアクセスできます。ルール・ライターが 定義するタイプの変数と、Active Correlation Technology によって 提供されるタイプの変数があります。式の中で直接アクセスできる タイプと、Active Correlation Technology によって提供されるメソッドを介してのみ アクセスできるタイプがあります。

varInitializer エレメント

<varInitializer> エレメントには、関連付けられた <variable> エレメントで定義されている変数の初期値を提供する式が含まれます。

詳細

任意のタイプの変数を使用できるため、式コードは、配列オブジェクトまたはその他の複雑なインプリメンテーション固有のオブジェクトを戻すことができ、それらが Active Correlation Technology エンジンによって保管されます。

式で使用できる変数に関する情報については、26 ページの『変数』 を参照してください。特定の変数の使用方法は、式のコンテキストに依存します。

属性

<varInitializer> には、以下の属性があります。

表 74. <varInitializer> エレメントの属性

名前	説明	データ・タイプ	必須?
expressionLanguage	式の記述に使用されているプログラミング言語を示します。サポートされている式言語は Java プログラミング言語のみであるため、この属性の唯一の有効値は `java` です。	xsd:NMTOKEN	はい

このエレメントを含むエレメント

<varInitializer> は、以下のエレメントに含まれます。

- <variable>

このエレメントに含まれるエレメント

<varInitializer> に含まれるエレメントはありません。

関連概念

21 ページの『式』
式とは、ルールに追加できるカスタム・ロジックを含むコードです。 式では、Active Correlation Technology エンジンの外部にあるコードにアクセスすることもできます。ルール言語では、式は、特定のコンテキスト、またはルール言語エレメント内でのみ有効です。

whenLoaded エレメント

<whenLoaded> エレメントは、Active Correlation Technology エンジンによってロードされたときにルールがアクティブ化されることを指定します。

属性

<whenLoaded> には、属性がありません。

このエレメントを含むエレメント

<whenLoaded> は、以下のエレメントに含まれます。

- <start>

このエレメントに含まれるエレメント

<whenLoaded> に含まれるエレメントはありません。

第 6 章 用語集

この用語集には、Active Correlation Technology の重要な概念に関する用語および定義が含まれています。

ア

アクション (action)
ルール応答の一部として、またはルールがロード、アンロード、アクティブ化、または非アクティブ化されたときに実行される式。

イベント・セレクター (event selector)
イベント選択のための基準。これらの基準により、ルールで処理するために受け入れられるイベントが決定します。イベント・セレクターには、イベント・タイプおよびフィルター述部が含まれます。

イベント・プロバイダー (event provider)
Active Correlation Technology によって処理されるイベントを生成する任意のソフトウェア。

インポート (import)
式から外部コードへのアクセスを可能にする、プログラミング言語固有の方法。

応答 (response)
ルール応答を参照。

カ

外部イベント (external event)
Active Correlation Technology エンジンが、外部のソースから受信するイベント。

外部オブジェクト (external object)
アプリケーションが式と通信するために作成するオブジェクト。

グループ化キー (grouping key)
共通の特性を共有するイベントの各グループについて、別個のルール・インスタンス (または自身のコピー) を作成するようルールに指示するためのメソッド。

計算パターン (computation pattern)
特定の時間間隔内で、収集されたイベントに対して、各イベントの受信時に (式を介して) 計算を適用するためのルールを定義するルール・パターン。計算パターンによって定義されるルールは、ステートフル・ルールです。

コレクション・パターン (collection pattern)
特定の時間間隔内で、選択されたイベントのグループを収集するためのルールを定義するルール・パターン。コレクション・パターンによって定義されるルールは、ステートフル・ルールです。

サ

シーケンス・パターン (sequence pattern)
特定の時間間隔内で、特定のイベント・シーケンスが存在するか、存在しないかを検出するためのルールを定義するルール・パターン。シーケンスは、順序付けられていても、ランダムであってもかまいません。シーケンス・パターンによって定義されるルールは、ステートフル・ルールです。

式 (expression)
ルールに追加できるカスタム・ロジックを含むコード。ルール・ライターは、変数の初期化、イベント選択基準の定義、またはルール応答アクションやライフ・サイクル・アクションの指定など、さまざまな目的で式を使用できます。

しきい値パターン (threshold pattern)
特定の時間間隔内で、選択されたイベントのグループを収集し、各イベントが受信された後、しきい値条件が満たされたかどうかを判別するためのルールを定義するルール・パターン。しきい値パターンによって定義されるルールは、ステートフル・ルールです。

式言語 (expression language)
式の記述に使用されたプログラミング言語。

述部 (predicate)
フィルター述部を参照。

ステートフル・ルール (stateful rule)
状態情報を保持するルール。状態情報は、一定期間にわたりイベントのコレクションに対して動作するための、ルール・インスタンスの特性に関する情報です。コレクション、計算、重複、シーケンス、しきい値、またはタイマーの各ルール・パターンによって定義されるルールは、ステートフル・ルールです。

ステートレス・ルール (stateless rule)
状態情報を保持せず、そのため一度に 1 つのイベントに対してのみ動作できるルール。フィルター・パターンによって定義されるルールは、ステートレス・ルールです。

スニペット (snippet)
ソース・コードの抜粋。

タ

タイマー・パターン (timer pattern)
一定間隔でアクションを開始するためのルールを定義するルール・パターン。タイマー・パターンによって定義されるルールは、ステートフル・ルールです。タイマー・ルールはイベントを処理しませんが、イベントによってアクティブ化または非アクティブ化できます。

重複パターン (duplicate pattern)
指定された時間間隔内に受け入れられた 2 番目以降のイベントをカウントするが、これらのイベントについてのルール・セット処理をスキップするためのルールを定義するルール・パターン。重複パターンによって定義されるルールは、ステートフル・ルールです。

ドメイン (domain)
ルールのグループがその機能に基づいて適用されるカテゴリー。例えば、ド

メインは、特定の地理上の区域、IT 管理規律 (セキュリティー検出またはネットワーク・イベント相関など)、またはビジネス組織 (特定の会社または会社の部門) を表すことができます。

ナ

内部イベント (internal event)
Active Correlation Technology エンジンで実行中のルールによって作成されるイベント。このイベントは、他のルールに転送されることがあります。

ノード (node)
ルール・セットにおいて、独立して個別に追加、除去、または置換できるルール階層内のオブジェクト。具体的には、以下のオブジェクトがノードです。

- ルール
- ルール・ブロック
- ルール・ブロック変数
- ルール・セット変数

ルール・レベルより下のレベルでは、オブジェクトを独立して個別に操作することはできないため、ルール変数はノードではありません。

ハ

フィルター・パターン (filter pattern)
イベントの受け入れ時に特定のアクションを実行するためのルールを定義するルール・パターン。フィルター・パターンによって定義されるルールは、単一イベントに対してのみ作用します。つまり、このルールはステートレス・ルールです。

フィルター述部 (filtering predicate)
ルールで処理するためにイベントを受け入れる条件を定義する式。フィルター述部は、イベント・セレクターの一部です。フィルター述部は、ブール値を戻します。

ラ

ライフ・サイクル・アクション (life cycle action)
ルールがロード、アンロード、アクティブ化、または非アクティブ化されたときに実行される式。

ルール (rule)
イベント間の関係を認識し、適切なルール応答を実行するために使用される相関単位。ルールは、7 つのルール・パターンのうち 1 つのインプリメンテーションで、その機能に応じて、ルール・セットの一部であるルール・ブロックに編成されます。ルールは、イベントがイベント選択基準を満たす場合に、そのイベントを受け入れて処理します。

ルール・インスタンス (rule instance)
グループ化キーのコンテキストにおけるルールのコピー。

ルール・セット (rule set)
Active Correlation Technology ルール言語のルール実行単位。ルール・セットには、Active Correlation Technology エンジンによって実行される、ルー

ル・ブロックに編成されたルールが含まれます。エンジンは、一度に 1 つのルール・セットについてのみ動作します。

ルール・パターン (rule pattern)
イベント相関状態 (しきい値条件または重複イベントの検出など) を表現したもの。Active Correlation Technology ルール言語には、コレクション、計算、重複、フィルター、シーケンス、しきい値、およびタイマーというルール・パターンが含まれます。ルールのパターンは、ルールによって定義された状態が発生したときに一致したことになります。パターンが一致した場合、ルールは適切なルール応答アクションを実行することによりその処理を完了します。ルールがアクティブである間、ルール・パターンは複数回一致することがあります。

ルール・ブロック (rule block)
ルールを機能に応じてルール・セット内のドメインにグループ化するための編成単位。ルール・ブロックには、ルールのみでなく、他のルール・ブロックも含めることができます。

ルール応答 (rule response)
Active Correlation Technology エンジンがルール条件が満たされたことを認識したときに実行される式。ルール応答は、1 つ以上のアクションで構成されます。

ルール応答アクション (rule response action)
アクションを参照。

A

ACT Active Correlation Technology を参照。

Active Correlation Technology
ルールを介したイベント相関を提供する IBM のテクノロジー。

Active Correlation Technology エンジン (Active Correlation Technology engine)
Active Correlation Technology コンパイラーの出力に従ってイベントを処理する Active Correlation Technology コンポーネント。

Active Correlation Technology コンパイラー (Active Correlation Technology compiler)
ルール・セットおよびその中に含まれるコードを構文解析して、Active Correlation Technology エンジンに必要な内部データ構造を生成する Active Correlation Technology コンポーネント。

Active Correlation Technology ランタイム環境 (Active Correlation Technology runtime environment)
Active Correlation Technology エンジンが、コンパイラーとともにまたはコンパイラーなしで組み込まれたアプリケーション。

Active Correlation Technology ルール・ビルダー (Active Correlation Technology rule builder)
Active Correlation Technology ルール言語で相関ルールを記述するための GUI。

Active Correlation Technology ルール言語 (Active Correlation Technology rule language)

イベントを相関させるルールを記述するための XML ベースの言語。これらのルールは、作成後、Active Correlation Technology ランタイム環境にデプロイできます。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

商標

DB2、IBM、IBM ロゴ、Tivoli、Tivoli ロゴ、Tivoli Enterprise Console、WebSphere は、IBM Corporation の商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12